THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE

Master of Technology
in
Computer Science & Engineering

Random Walk based Search and Community Formation in Power Law P2P networks

---

Author:

Tathagata Das
Roll No: 03CS3022

Supervisor:

Prof. Niloy Ganguly
Assistant Professor

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY
KHARAGPUR

May 6, 2008

CERTIFICATE

This is to certify that this thesis entitled "**Random Walk based Search and Community Formation in Power Law P2P networks**" submitted by **Mr. Tathagata Das** to the **Department of Computer Science & Engineering, Indian Institute of Technology Kharagpur** in partial fulfillment of the requirement for the degree of **Masters of Technology in Computer Science & Engineering during** the period 2006-2008 is a record of authentic work carried by him under my supervision and guidance.

Dated: May 7, 2008
IIT Kharagpur

Prof. Niloy Ganguly
Dept. of Computer Science & Engineering
Indian Institute of Technology
Kharagpur

Acknowledgements

With great pleasure and deep sense of gratitude, I express my indebtedness to *Prof. Niloy Ganguly* for his invaluable guidance and constant encouragement at each and every step of my project work. He exposed me to the intricacies of relevant topics through proper counseling and discussions and always showed great interest in providing timely support and suitable suggestions. Along with him, I am very deeply grateful to *Subrata Nandi*, whose constant flow of ideas and inexhaustible enthusiasm have helped me overcome many hurdles that I have faced in my work.

I would like to express my gratitude to all my friends and colleagues in Computer Science Department and otherwise, especially *Sankalp Agarwal, Mukesh Agarwal and Sanchayan Chakraborty*, for their constant help and encouragement.

And finally, words are not enough to express my indebtedness and gratitude towards my parents to whom I owe every success and achievement of my life. Their constant support and encouragement under all odds has brought me where I stand today.

Contents

1 Introduction **1**

1.1 Peer-to-peer networks 1
1.2 Search in P2P networks 2
1.3 Community Formation 3
1.4 Search on P2P Community 3
1.5 Motivation and Objective 5
1.6 Thesis Layout 6

2 Literature Survey **7**

2.1 Community Formation 7
2.2 Community Evaluation 8

3 P2P Modeling **10**

3.1 Network Model 10
3.1.1 Topology & Network Load 10
3.1.2 Profile Distribution 11
3.1.3 Search & Matching 11
3.2 Evaluation Criteria 11
3.2.1 Search related metrics 11
3.2.2 Metrics related to community formation 12

4 Basic Exploration on Random Walk and Proliferation **13**

4.1 Basic Algorithm 13
4.1.1 Search 13
4.1.2 Community Formation 14
4.2 Simulation and Results 16
4.2.1 Simulation Plan 16
4.2.2 Results and Analysis 17
4.3 Probabilistic $\mathcal{RG}$ and $\mathcal{GG}$ 20

5 An Approach to Self-Adjusting Search ($\mathcal{SA}$) 22
5.1 The Final Algorithm 22
5.2 Simulation Results 25
5.3 Performance under Node Churn 25
5.4 Performance under Overlapping Interest Categories 28
5.4.1 Modification ($\mathcal{SA}'$) 29

6 Evaluation of P2P Communities: An intuitive approach 32
6.1 Definition 32
6.2 Evaluation of Explored Algorithms 35

7 Future Work 37

8 Conclusion 38

List of Figures

1.1 Random walk with proliferation – The initiator node produces few random walkers, which upon reaching the required destination nodes *proliferates* to produce new walkers. 4

4.1 Schematic representation of random walk and proliferation in the search algorithm – The node that wants to search produces few random walkers (i.e. the query packets, each of which randomly walk through non-matching/non-similar nodes until it reaches a similar node. The new found similar node gets connected to the initiator node by a *community edge*. Next, the query is *proliferated* into multiple new walkers which continue the search process. 15

4.2 Performance of $\mathcal{RR}$ search using Community Edge Addition (CEA) compared to Random Edge Addition (REA) 16

4.3 Performance of $\mathcal{RR}$, $\mathcal{RG}$ and $\mathcal{GG}$ wrt hit count and search efficiency . 18

4.4 Correspondence between LCC size and fraction of similar nodes returned in $\mathcal{RG}$ and $\mathcal{GG}$ 18

4.5 Average number of new similar nodes discovered (that is, nodes never found before) in each search by $\mathcal{RG}$ and $\mathcal{GG}$. Inset: $\mathcal{GG}$ continues to find more nodes than $\mathcal{RG}$. 19

4.6 Total node coverage and redundant node coverage in $\mathcal{RG}$ and $\mathcal{GG}$ . . . 20

4.7 Performance of different probabilities of random walk in $(\mathcal{RG})\mathcal{G}_P$ Search 21

5.1 Performance of $\mathcal{SA}$ wrt Hit count and efficiency 23

5.2 Performance of $\mathcal{SA}$ wrt LCC and similar node coverage 24

5.3 Total node coverage and redundant node coverage in $\mathcal{RG}$ and $\mathcal{GG}$ . . . 25

5.4 Performance of $\mathcal{SA}$ wrt LCC and search efficiency under different amount of node churn 26

5.5 Performance of $\mathcal{SA}$ wrt search efficiency under 5% churn with different P_{add} 27

5.6 Performance of $\mathcal{SA}$ wrt search efficiency under 20% churn with different P_{add} 28
5.7 Performance of $\mathcal{SA}$ wrt search efficiency under overlapping profiles . . 29
5.8 Performance of $\mathcal{SA}'$ wrt LCC and search efficiency under overlapping profiles 30

6.1 Schematic representation of the community evaluation scheme 32
6.2 Performance of $\mathcal{SA}$ wrt new community evaluation scheme 35

List of Tables

4.1 Neighbor selection strategies in different search algorithms 14

Abstract

In this work, an attempt was made to understand the dynamic of the community formation over P2P networks having power law topology and incrementally develop a novel and efficient algorithm that can search the network and simultaneously form communities to improve future searches. It is a completely decentralized algorithm where each peer searches by sending out random walkers to a limited number of neighbors. As it finds other peers having similar content, it restructures its own neighborhood with the objective of bringing them closer. This restructuring leads to clustering of nodes with similar content, thus forming P2P communities. Alongside, the search algorithm also adapts its walk strategy in order to take advantage of the community thus formed. This search strategy is more than twice as efficient as pure random walk on the same network. It has been shown to be efficient and robust under dynamic scenarios involving a continuous node churn. Furthermore, a new scheme has also been proposed to evaluate the community structure taking into consideration the inherently overlapping nature of such content/interest based communities.

Chapter 1

Introduction

1.1 Peer-to-peer networks

There has been a growing interest in peer-to-peer networks since the initial success of some very popular file-sharing applications such as Napster and Gnutella [20]. A peer-to-peer (P2P) network is a distributed system in which peers employ distributed resources to perform a critical function in a decentralized fashion. Nodes in a P2P network normally play equal roles, therefore, these nodes are also called peers. A typical P2P network often includes computers in unrelated administrative domains. These P2P participants join or leave the P2P system frequently, hence, P2P networks are dynamic in nature. P2P networks are overlay networks, where nodes are end systems in the Internet and maintain information about a set of other nodes (called neighbors) in the P2P layer. These nodes form a virtual overlay network on top of the Internet. Each link in a P2P overlay corresponds to a sequence of physical links in the underlying network. P2P networks offer the following major benefits

- They are self-organized and adaptive. Peers may come and go freely. P2P systems handle these events automatically.
- They can gather and harness the tremendous computation and storage resources on computers across the Internet.
- They are distributed and decentralized. Therefore, they are potentially fault-tolerant and load-balanced.

P2P networks can be classified based on the control over data location and network topology. There are three categories: *unstructured, loosely structured/semi-structured, and highly structured*. In an unstructured P2P network such as Gnutella [20], no rule

exists which defines where data is stored and the network topology is arbitrary. In a loosely structured or semi-structured network such as Freenet and Symphony, the overlay structure and the data location are not precisely determined. In Freenet, both the overlay topology and the data location are determined based on hints. The network topology eventually evolves into some intended structure. In Symphony, the overlay topology is determined probabilistically but the data location is defined precisely. In a highly structured P2P network such as Chord, Pastry and CAN, both the network architecture and the data placement are precisely specified. The neighbors of a node are well-defined. The data is stored in well defined locations.

1.2 Search in P2P networks

searching techniques in P2P networks is a very active research issue. The desired features of searching algorithms in P2P systems include high-quality query results, minimal routing state maintained per node, high routing efficiency, load balance, resilience to node failures i.e. node churn. Searching in highly structured systems follows the well-defined neighboring links. For this reason, highly structured P2P systems provide guarantees on finding existing data and bounded data lookup efficiency in terms of the number of overlay hops; however, the strict network structure imposes high overhead for handling frequent node join-leave. Unstructured P2P systems are extremely resilient to node join-leave, because no special network structure needs to be maintained. Searching in unstructured networks is often based on flooding or its variation because there is no control over data storage. The searching strategies in unstructured P2P systems are either blind search or informed search. In a blind search such as iterative deepening, no node has information about the location of the desired data. In an informed search such as routing indices, each node keeps some metadata about the data location. An unstructured P2P network can not offer bounded routing efficiency due to lack of structure. Searching in a loosely structured system depends on the overlay structure and how the data is stored. In Freenet, searching is directed by the hints used for the overlay construction and the data storage. In Symphony, the data location is precisely defined but the overlay structure is probabilistically formed. Searching in Symphony is guided by reducing the numerical distance from the querying source to the destination node where the desired data is located. The loosely structured systems can offer a balanced trade-off if they are properly designed.

1.3 Community Formation

Search efficiency can be improved upon by maintaining some information from previous search experiences locally in the peer nodes. Another alternative is to restructure the network such that the nodes containing similar content or data profiles are moved closer to each other. P2P network, being an appropriate example of socio-technological networks, inherently has the potential for developing a community structure. Therefore, the latter approach seems to be intuitively appealing and does not involve the overhead of maintaining search related information, as in the previous case. Formation of such peer communities based on the interest or content of the participating peers/nodes lies in the domain of semi-structured networks. The concept of peer communities is loosely based on the idea of *interest groups*, for example Yahoo Groups or Usenet Newsgroups. A node in the system claims to have some interests and depending upon the claims of all the peer nodes, the communities are implicitly formed (made up of peers with the same or similar interests). These nodes that claim to be having similar interests may reorganize their link structure in order to bring themselves closer in the network and make search more efficient. Such communities maybe either be *flat* or *hierarchial*. Formation and discovery of such communities is an active research issue.

The idea of interest/content based communities in P2P networks is closely tied with the categorization of interests/content. If we try to imagine an abstract idea of *interest* and categorize it into subcategories, then one of most intuitive property that we will find is the fact that such categories overlap with each other. In other words, there is no well defined boundary between two categories as some subjects of interest will have a certain degree of match with more than one of the defined categories. Due to this overlapping nature of the interest categories, communities formed based on them is bound to be overlapping by nature. Hence, to effectively evaluate P2P communities, the evaluation scheme should be able to identify overlapping communities.

1.4 Search on P2P Community

The search mechanisms may vary from flooding based searches to random walk based searches. Each different scheme has its own pros and cons. Formation of communities can boost the efficiency of all the search schemes, each to a different degree. There has been a lot of recent research work done in the area of community formation, but mostly using flooding based techniques. Random walk based searches have the advantage of having a very low bandwidth consumption compared to flooding. But its efficiency in

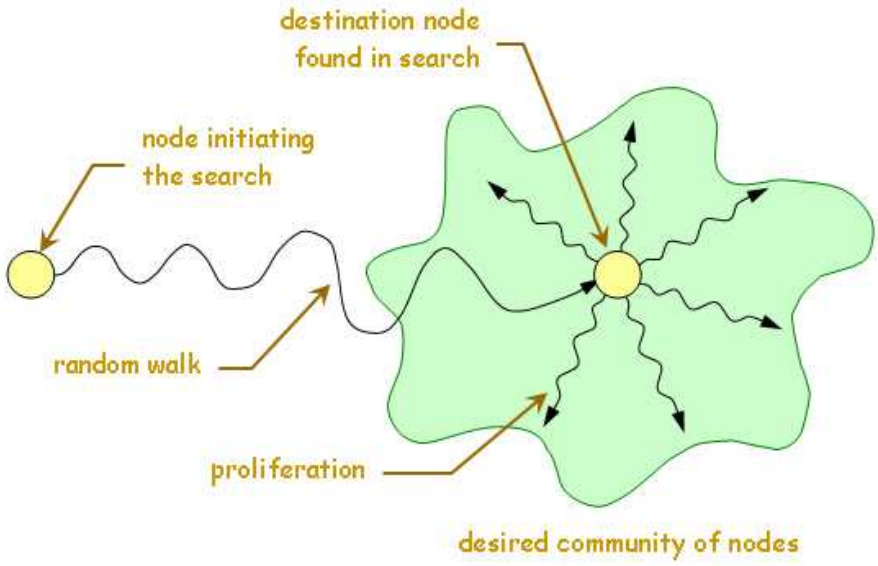


Figure 1.1: Random walk with proliferation – The initiator node produces few random walkers, which upon reaching the required destination nodes *proliferates* to produce new walkers.

finding search results in the network is also poor, and the challenge lies in attempting to significantly improve random walk based searches by forming interest based communities.

In this work, a variation of random walk based algorithm has been explored that is inherently very suitable for searching in communities. This variation is termed as ***Random Walk with Proliferation***. In this scheme, the node that wants to initiate the search, produces a few random walkers that walk through the network. When each of them reaches a destination node (i.e. a node in the desired community of nodes), then each of them generates a number new random walkers. These new random walkers will be able to make a more intensive search in the neighborhood of the destination nodes (that is within the community), and very easily find more desired nodes. This is intuitively very suitable for community searching because of the following reasons.

- Unlike flooding, it does not blindly produce too many walkers/packets right at the start of the search. Only when it finds a node belonging to the desired community, it *proliferates* to produces more walkers/packets in order to search the community. Hence its bandwidth usage is much lower compared to flooding based searches.
- Whenever it has found a desired node, it knows that a community might exist in

its vicinity. Hence, it intelligently generates more walkers since it is expected that there are a large number desired nodes in the neighborhood which can be found fast if searched with more walkers. Hence, unlike pure random walk, it is able to efficiently find desired nodes, the only constraint being the presence of a good community structure.

Such schemes that proliferates upon reaching desired positions in the network has been used in different works but none has formally explored the basic idea and attempted to understand its dynamics. In this work we try to understand the dynamics of the random walk with proliferation, with the aim of simultaneously building interest/content based P2P communities and also efficiently searching them.

1.5 Motivation and Objective

The idea of forming P2P communities to improve search efficiency is a very active research field. Many groups have explored this concept with very specific types of networks, while other have applied it to Erdos-Renyi networks. However, except some of the works by Prof. Ganguly et al [2],[1], there has hardly been work on algorithms where the search process itself triggers community formation. The idea is that the network as a whole gets trained / acquires memory as search progresses. This work derives its main inspiration from the previous work by Prof. Ganguly, which explored the concept of community formation by restructuring the neighborhood in a grid-like topology. The algorithms developed there can not be ported in a more realistic power law network.
Hence, the objective of this work was the following.

- Exploration and understanding of the dynamics of community formation and random walk with proliferation on power law networks
- Development of a novel decentralize and self-adjusting random walk based search algorithm that can simultaneously form communities as well as efficient search them
- Development of a new scheme for evaluation of overlapping peer-to-peer communities

1.6 Thesis Layout

The rest of the thesis is organized as follows. Chapter 2 gives a brief idea of the relevant work done in the field of community formation and evaluation in the existing literature. Chapter 3 gives a detail description of the model of P2P network that has been considered for this work. Chapter 4 explores the basic idea of random walk with proliferation and community formation, to understand the underlying dynamics. Based on this understanding, a final algorithm is proposed, whose details and performance under different conditions have been discussed in chapter 5. Chapter 6 states the details of the new intuitive scheme developed for evaluation of P2P communities, along with the performance of the explored algorithms with this metric. Finally, chapter 7 and 8 concludes the thesis with the extensions possible in this work in the future.

Chapter 2

Literature Survey

There have been quite a number of publications in the field of community formation as well as community detection in peer-to-peer networks. Here, we outline a brief idea of the existing literature in the fields of community formation and community detection / evaluation.

2.1 Community Formation

Almost all of the work in this area can be broadly outlined as follows. Each of the work has incorporated the following features in their community formation schemes -

P2P model Completely unstructured, or two-tiered with overlay, etc

Information categorization Subdivision of the information possessed by the peer nodes into abstract categories

Rewiring scheme Method to match the similarity between information or query patterns of two peers

Evaluation Addition and deletion of edges in order to bring the two similar nodes closer to each other, while considering factors like node degree, link weight, number of times the node has searched, etc

Each of the proposed community (also known as clubs) formation schemes have different flavors of the above items incorporated into it. To take a few examples, Asvanund and. Krishnan [3] have proposed a two-tier architecture based distributed digital library network. Matching of interests is done by Kullback-Leibler (K-L) divergence of the

past query pattern of one node with the content of other. Digital libraries (peers) connect to the matching regional directories (ultrapeers) if both sides match with each by the K-L divergence. Search is performed by the Gnutella 0.6 protocol. Another work by Khambatti, Ryu and Dasgupta [4],[5] brings in the concept of seers, which are the set of nodes which has contacts of a large number of similar nodes in the network. These seers are very valuable in terms of search because for a query, finding a corresponding seer will immediately give the contacts of other similar nodes and hence produce more results. Finally, flooding based search is performed in the network. Yet another work, done by Ganguly, Canright and Deutsch [2], [1], have put forward a very simple scheme for interest matching between peers and query matching by bit patterns. But they have considered a grid topology, which seems a bit unrealistic from real networks point of view.

In all, most of the work done in this field deals with flooding based search techniques (like Gnutella for example). Besides this, very few work have properly dealt with all the issues together, issues like possible network breakage, degree increase of nodes, evaluation of the search by proper search related metrics as used in [8]. Also another noticeable fact is that all the work has evaluated their schemes based on the search performance. None have dealt with the community structure in itself, that is, quantitatively evaluation of community structure and topology independent of the search algorithms. Our intention is design a simple community formation algorithm that is intuitive by nature and deal with these issues in a clear and coherent manner.

2.2 Community Evaluation

A significant amount of literature exists that deal with community detection and evaluation in networks. These detection schemes have been applied to all different kind of networks - theoretical networks as well as real-world networks like roadway networks, protein-protein interaction networks, social networks, disease networks, etc. [16] gives a very brief yet comprehensive overview of the schemes that exist. The question arises whether these scheme are applicable to interest based communities in peer-to-peer networks.

As discussed in section 1.3, proper evaluation of the P2P communities requires the evaluation scheme to consider the formation of the overlapping communities. In the existing literature, there seems to be only one well known algorithm that can detect overlapping communities - Clique Percolation Algorithm (CPA) [22], [23], [21]. This

algorithm uses the idea of *k-clique* communities, which is defined as the set of those cliques each of which overlap with each other by at least $k-1$ nodes. This method has been proved to be very successful in detecting communities in complex networks like protein-protein interaction networks and citation networks. But upon further analysis of the scheme, it seems hard to fit in our requirement for evaluating P2P communities. CPA does not take link weights into consideration. But in a P2P network, two nodes are said to be part of community if they are similar, if their inter-nodal distance is less, etc. In other words, both node properties (like differences in interest / content) and edge properties (like link distances) need to play a role in the evaluation of P2P communities. CPA could be adapted for evaluating such overlapping P2P communities by reducing the node and edge properties into a single edge weight, and then using that edge weight as a cut-off for the subset of edges that are to be considered for evaluation by CPA. But this reduction is tricky and the whole process becomes complicated. Hence, in order to evaluate P2P communities, a simple and intuitive approach has been suggested that properly takes into consideration node and edge properties.

Chapter 3

P2P Modeling

3.1 Network Model

Not only the internet as a whole [8], but also different existing P2P networks like Gnutella and KaZaA [7] exhibit a power-law topology. Inspired by this fact, this work assumes a realistic power-law topology for the modeled P2P network. Also, in order to form content base communities, we have classified the information content of the peers into abstract subcategories. The details are provided below.

3.1.1 Topology & Network Load

The initial P2P network is considered to be of a power law topology. According to the characteristic heavy tailed nature of power law networks, few nodes have high degrees while the majority of the nodes have low degrees. These initial connections are assumed to form a connectivity layer among the nodes and are hence termed as *Connectivity Edges*. New edges that are added to the network with the intention of forming community structures over the connectivity layer are called *Community Edges*.

For the purpose of our analysis, we consider the degree of a node as a measure of its continuous bandwidth usage, assuming that a low bandwidth consuming gossip protocol maintains the communication between the neighbors. Hence, there is a limit to the total number of edges it can have. In other words, each node can sustain only a limited number of new community edges. This increase in network load is measured relative to its initial degree (that is, the degree corresponding to its connectivity edges). This measure is termed as X where

$$X = \frac{New\ Degree\ -\ Initial\ Degree}{Initial\ Degree}$$

The maximum network load that each node can tolerate is assumed to be X_{max} times the initial network load (that is, the initial degree). Also, during the search protocol, there will be bursts of high bandwidth usage when a node needs to communicate with its neighbors. This is also limited by a maximum number of neighbors that a node can contact in a single burst of communication. Let this limit be known as Y_{max}.

3.1.2 Profile Distribution

In a file sharing P2P network, each node shares some data with other nodes in the network. This data is categorized into abstract categories called *Information Profiles*. This profile (P_I) therefore reflects the informational content as well as the informational interest of the user. It is represented in our system as a d-bit binary value, thus producing 2^d distinct categories. These profiles are distributed among the nodes following the Zipf's Law with the idea that some categories of data are highly popular whereas others are not.

3.1.3 Search & Matching

A search query is defined as a m-bit binary value, which is taken to be equal to the information profile P_I of the node that is initiating the search. This is based on the simple idea that the user of the node would like to search for items that fall in the same category as his own information content. In order to find nodes having similar content, the query packet is forwarded in the network according to the rules set by the search algorithm. Each node that encounters the query packet tries to match its own profile with the queried profile. When a node is found whose information profile exactly matches the query profile, it is said to be a *search hit* and the initiator node and matched node are said to be *similar* nodes.

3.2 Evaluation Criteria

We will like to evaluate the performance of these algorithms based on the following criteria. Besides the ones presented here, other measures will be used as and when required.

3.2.1 Search related metrics

Let the i^{th} search contact a total of c_i number of nodes of which h_i search hits were made, all using a total of p_i packets. Let the total number of nodes similar to the initiator node

(that is the maximum possible search hits) be H_i. Let the search be performed n times. Then the search-related metrics are defined as follows:

- *Total Hit Count*: Average number of hits (similar nodes) found in each search, i.e. $\frac{1}{n}\sum_i h_i$
- *Efficiency*: Average number of hits per search packet, i.e. $\frac{1}{n}\sum_i \frac{h_i}{p_i}$
- *Node Coverage*: Average number of nodes contacted for finding similar nodes in each search, i.e. $\frac{1}{n}\sum_i c_i$
- *Similar Node Coverage*: Average fraction of all the similar nodes present in the network that is returned in each search, i.e. $\frac{1}{n}\sum_i \frac{h_i}{H_i} \times 100$
- *Redundant Node Coverage*: Average number of nodes it unnecessarily contacts more than once in each search
- *Node Discovery*: Average number of new nodes found in a search (i.e. nodes that have not been found by the initiator node in earlier searches).

3.2.2 Metrics related to community formation

The community edges make connections between similar nodes only. If the nodes of a particular profile are considered, then these edges form a community overlay network over these nodes. The size of the largest connected component (LCC) in a network is generally considered as a measure of its connectedness. Since, it is desired that all the nodes of a profile are well connected by the community overlay network, the LCC of the network is taken to be a measure of the '*goodness*' of the community structure. It is expressed in terms of the percentage of nodes of each particular profile that lie within the LCC. This is averaged over all the profiles in the system, and is termed as *Average LCC* of the community structure.

Chapter 4

Basic Exploration on Random Walk and Proliferation

4.1 Basic Algorithm

As explained in section 1.4, random walk with proliferation is used as the basic algorithm with which we search the network. Four major variations of this algorithm is tested our in order to understand the dynamics under various conditions. These variations have been named as $\mathcal{RR}$, $\mathcal{RG}$, $\mathcal{G}R$ and $\mathcal{GG}$. Along with these, we also implement a fifth algorithm named $(RG)G$, which probabilistically performs random or greedy walk in order to achieve the best of both. In this section, we describe these algorithms in full detail. As mentioned earlier, there are two distinct processes in the algorithms – Search and Community Formation.

4.1.1 Search

Any node in the networks can start a search query. Let us say, it is initiated at a node U. It sends a search query message M to a few of its randomly selected (at most Y_{max}) neighbors, carrying the information profile (P_I) of U as the query profile to be searched. This message packet walks through the network until it comes across a node whose information profile matches with the queried profile. Then it is said to have made a *search hit*. Let that node be called node A. Following the search hit, A performs two operations - *Proliferation* and *Community Formation*. A proliferates (replicates) the query to a number of its neighbors (at most Y_{max} neighbors) with the aim of making a more intensified search in its vicinity. This is done to exploit the fact that due to community formation, nodes similar to A (hence similar to U) should be present in the neighborhood of A. Moreover, the general walk is further optimized by making each

query packet store the nodes it has traversed through, so that they are avoided while forwarding the packets.

The neighbor selection process for general walking and proliferation decides the randomness / greediness of the overall walk mechanism. The neighbors for the general query forwarding can be selected in two ways.

- *Random*: Any neighbor connected by any type of edge is chosen randomly
- *Greedy*: A neighbor connected by community edges is preferred over other neighbors

Similarly, during proliferation, the neighbors can be chosen in either of the following ways.

- *Random*: Any number of the connected neighbors are chosen without any bias
- *Greedy*: Neighbors connected by community edges are preferred over other neighbors

	Search algorithms			
Neighbor selection strategy	$\mathcal{RR}$	$\mathcal{GR}$	$\mathcal{RG}$	$\mathcal{GG}$
During query forwarding	Random	Greedy	Random	Greedy
During proliferation	Random	Random	Greedy	Greedy

Table 4.1: Neighbor selection strategies in different search algorithms

Various permutations of these general walk and proliferation schemes lead to four different types of searches. As shown in table 4.1, they have been named by two letters based on the $\mathcal{R}$andom or $\mathcal{G}$reedy scheme used. The first letter represents the scheme used for general walk and second letter for the proliferation scheme. We next explain the latter process, that is, Community Formation.

4.1.2 Community Formation

Whenever there is a search hit, we want to evolve the topology in order to increase the probability of the next query reaching the node A from U. This can be ensured simply by connecting the similar nodes U and A with a new community edge. This brings the similar nodes within one hop distance of each other, thus increasing the probability of reaching it by random walk from $(1 \,/\, \#k^{th}\, neighbors)$ (k = previous shortest distance of A from U)to $(1 \,/\, \#neighbors)$. On the other hand, due to the network load limit of X_{max},

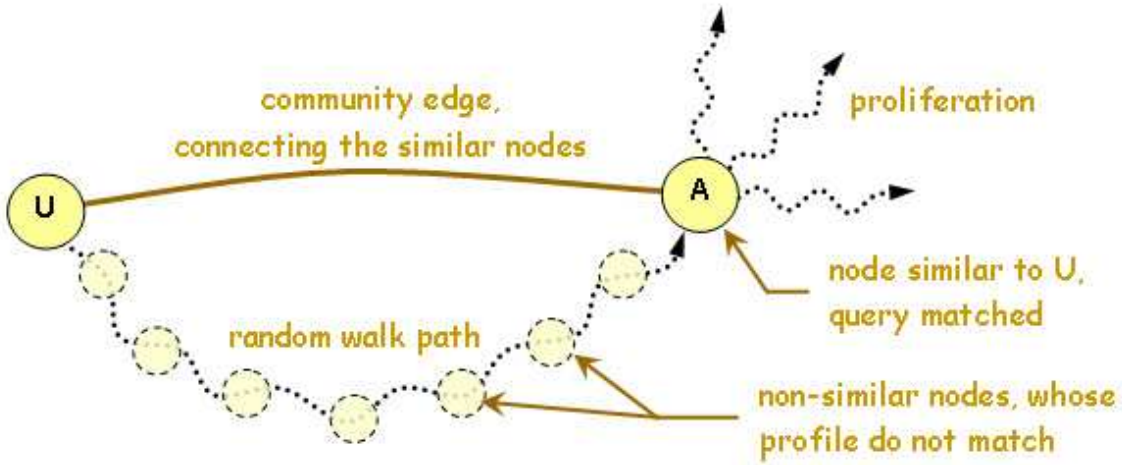


Figure 4.1: Schematic representation of random walk and proliferation in the search algorithm – The node that wants to search produces few random walkers (i.e. the query packets, each of which randomly walk through non-matching/non-similar nodes until it reaches a similar node. The new found similar node gets connected to the initiator node by a *community edge*. Next, the query is *proliferated* into multiple new walkers which continue the search process.

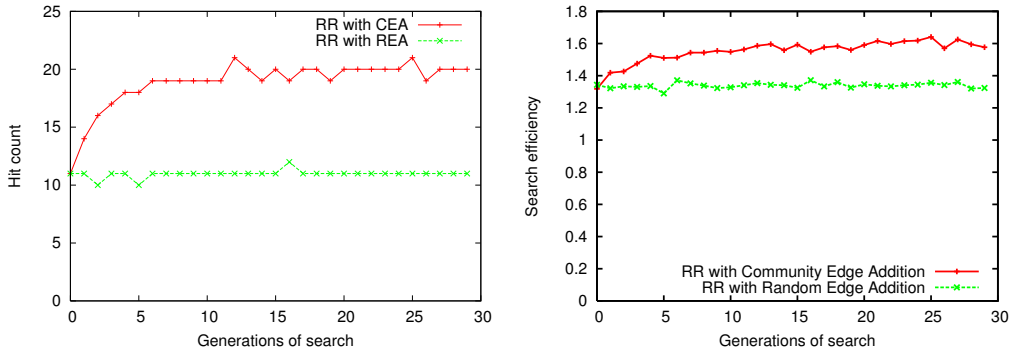
the algorithm is forced to delete edges when a new edge AB causes the network load of A and/or B to exceed its limit. Hence, we delete the edge with the following strategy. If both A and B exceed limits because of the new edge AB, then this edge is removed. If either A or B exceeds the limit, then another community edge is randomly selected for deletion from the corresponding node. Otherwise, since no limits are exceeded, the edge AB is allowed to remain. Furthermore, each edge is added with a probability of P_{add}. Otherwise the network load of each node would reach its limit very fast and further edge rewiring would produce a large amount of unnecessary churn in the network. It must also be noticed that we are churning only the community edges, and not the connectivity edges, which ensures that the whole network remains connected at all times.

Besides the 4 versions of the algorithm – $\mathcal{RR}$, $\mathcal{RG}$, $\mathcal{GR}$ and $\mathcal{GG}$, we also design a Probabilistic Random/Greedy Walk with Greedy Proliferation ($(\mathcal{RG})\mathcal{G}_P$). In this scheme, each query packet now holds another parameter - *Random Walk Probability* (P). At the time of initiation of the search, the value of the probability is set by the initiator node. This probability is also copied to the new packets created at the time of proliferation. Based on this probability, the non-matching nodes through which the packets pass, will either forward the packet either randomly (like $\mathcal{R}*$) or greedily ($\mathcal{G}*$).

In the matching nodes, the behavior is always the same - greedy proliferation (as in $*G$). The probability is set to different values between 0.0 (pure $\mathcal{RG}$) and 1.0 (pure $\mathcal{GG}$).

4.2 Simulation and Results

In order to test out the performance of the proposed algorithms, we resorted to simulations whose details are as follows.



(a) Total number of hits and packets generated

(b) Search efficiency

Figure 4.2: Performance of $\mathcal{RR}$ search using Community Edge Addition (CEA) compared to Random Edge Addition (REA)

4.2.1 Simulation Plan

The algorithms were simulated on a power-law network of 1000 nodes, generated using the Barabasi-Albert preferential attachment method [6], which gave us a gamma of approx 2.0. 16 profiles ($m = 4$) were distributed among the nodes by Zipf's law with a gamma of 0.8. Each search query is propagated in this network up to 15 hops. A set of search queries (generally 200) executed on random nodes constitute a *generation* and all performance metrics were averaged over a generation. Edge addition probability P_{add} is 0.2, while the network load limit X_{max} is 1.5. Y_{max} was chosen to be 3 nodes. A number of generations performed on the same network constitute a *simulation*. Due to the inherent random nature of the algorithm and the system at large, a single simulation is not sufficient to produce stable average results. Hence, multiple simulations are performed on different profile distributions for averaging the performance of the algorithm. It was found that approximately 20 simulations were enough to produce stable and smooth results, thus successfully abstracting out the inherent high degree of randomness present in the system.

In order to prove the importance of community formation, a fairness test was performed by comparing the performance of network formed through community edge addition (CEA) with an equivalent network. In this equivalent network, starting from the same initial power law network as the actual simulated network, edge are added randomly (that is, random edge addition (REA)) to compensate for the increase in the edge count of the original network (due to community edge addition). Also, the general characteristics of CEA are provided to demonstrate the aspects in which it varies from REA.

After this, we simulate the rest three algorithms - $\mathcal{GR}$, $\mathcal{RG}$ and $\mathcal{GG}$. But, due to lack for interesting inferences, we omit the results of $\mathcal{GR}$. The efficiency of the two algorithms are firstly compared with $\mathcal{RR}$, then other aspects of the algorithms are compared with each other and analyzed for understanding its internal dynamics.

4.2.2 Results and Analysis

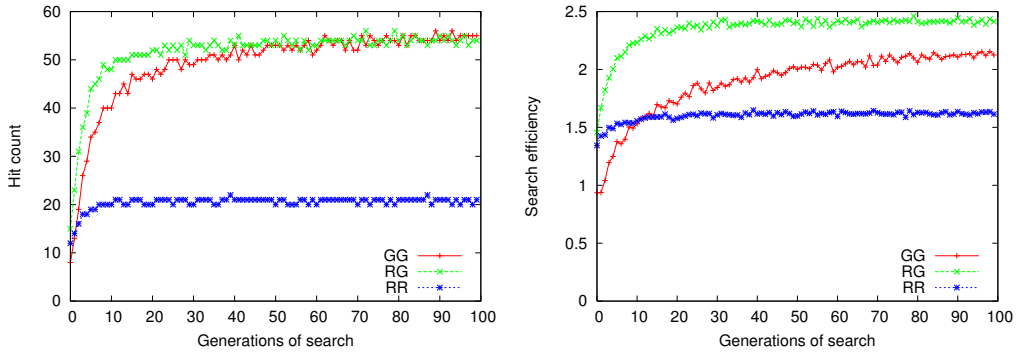
The results of the simulations are presented below.

- **$\mathcal{RR}$ with CEA vs REA**

Comparing the performance of $\mathcal{RR}$ with community edge addition versus random edge addition on an equivalent graph, Fig. 4.2(a) shows that as generations of search progress, the total number of hits returned by community edge addition increases steeply compared to random edge addition. Finally the former produces an average of 20 hits compared to 11 by the latter. In terms of the search efficiency, the former performs up to 20% better than the latter (Fig. 4.2(b)). This clearly proves that strategic addition of edges by community formation improves the search efficiency, unlike random addition edges.

- **$\mathcal{RG}$ and $\mathcal{GG}$**

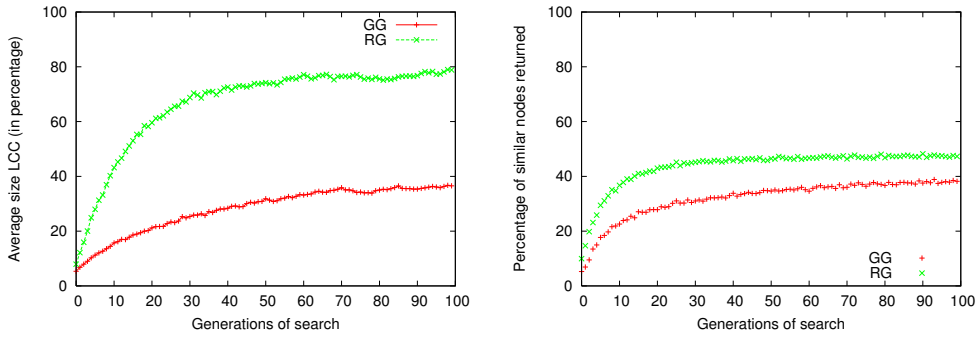
Next we present the performance of $\mathcal{RG}$ and $\mathcal{GG}$. All these cases undergo community edge addition. Figure 4.3(a) shows that on average, the number of results brought by both types of greedy-proliferation based searches are comparable, while being more than 2.6 times better than that of $\mathcal{RR}$. In terms of search efficiency, $\mathcal{GG}$ and $\mathcal{RG}$ perform about 30% and 50% better than $\mathcal{RR}$, respectively. Also, $\mathcal{GG}$ saturates much



(a) Hit count (b) Search Efficiency

Figure 4.3: Performance of $\mathcal{RR}$, $\mathcal{RG}$ and $\mathcal{GG}$ wrt hit count and search efficiency

slower compared to $\mathcal{RG}$. Both these figures confirm without doubt the importance of greedy walking in proliferation. This is actually obvious – only by greedily choosing the community edges can the already formed community be efficiently searched.



(a) Average size of LCC (b) Percentage of similar nodes returned

Figure 4.4: Correspondence between LCC size and fraction of similar nodes returned in $\mathcal{RG}$ and $\mathcal{GG}$

The most obvious question that arises is - what produces the difference in the search efficiencies of $\mathcal{GG}$ and $\mathcal{RG}$? This is primarily because of the extent of community formation in both cases. To quantitatively measure the community formed between nodes of a particular profile, we calculate the size of the largest connected component (LCC) in terms of the fraction of the similar nodes it contains. The larger this fraction, the more well connected they are. Referring to Fig. 4.4(a), the LCC in case of $\mathcal{RG}$ encompasses around 80% of all the similar nodes while it is just 40% in case of $\mathcal{GG}$. Greedy general walking in $\mathcal{GG}$ is unable to produce as good a community structure as the random walking in $\mathcal{RG}$, since it directs all the query packets into already discovered

areas of the network and hence inhibiting the exploration (that is, node discovery). But on the other hand, $\mathcal{RG}$ is also not able to exploit the good community structure created, as it is returning a smaller fraction of similar nodes compared to that present in the LCC. Refer Fig. 4.4(b), $\mathcal{GG}$ is finding almost all (95%) the nodes that constitute the LCC (36%), while $\mathcal{RG}$ returns just around 60% of all such nodes in LCC (79%).

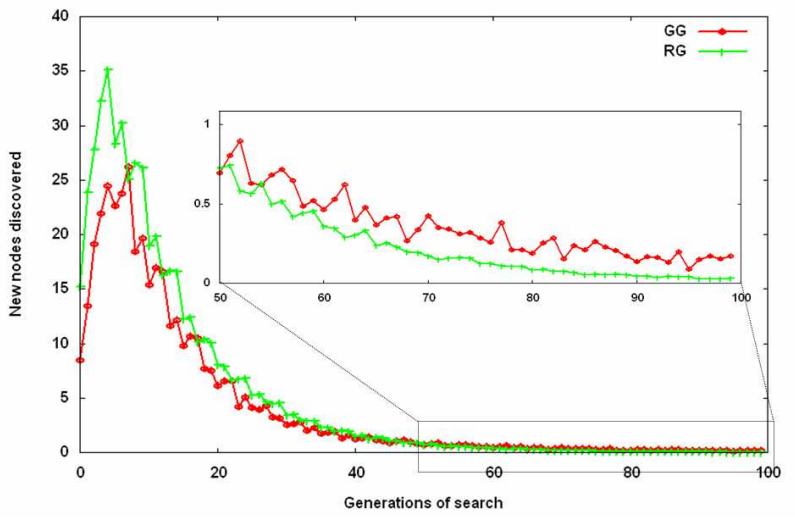
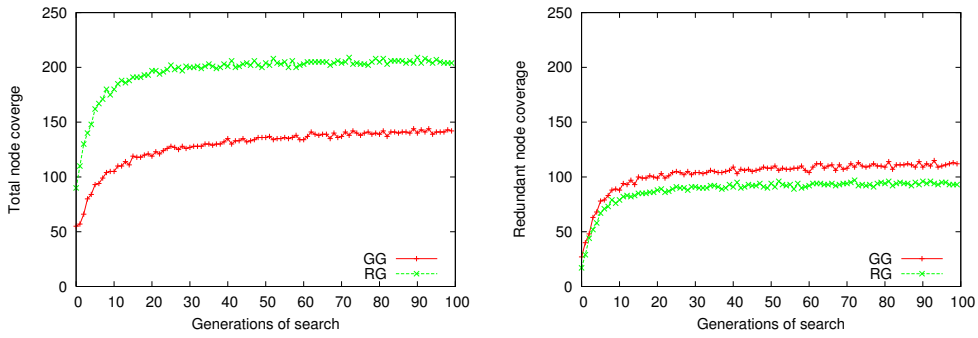


Figure 4.5: Average number of new similar nodes discovered (that is, nodes never found before) in each search by $\mathcal{RG}$ and $\mathcal{GG}$. Inset: $\mathcal{GG}$ continues to find more nodes than $\mathcal{RG}$.

$\mathcal{GG}$ also takes longer time to reach saturation compared to $\mathcal{RG}$. As Fig. 4.5 shows, even though $\mathcal{GG}$ is unable to explore as much in the initial generations, in the later generations, $\mathcal{GG}$ continues to find more new nodes than $\mathcal{RG}$. Hence, the community structure in $\mathcal{GG}$ continues to grow, slowly improving the search efficiency till it saturates.

Besides the search efficiency, the search algorithms must be evaluated based on the number of nodes it visits (node coverage) when it is searching and the number of unnecessary repeated node visits it makes. A good search algorithm should keep both the above counts to the minimum. Compared to $\mathcal{RG}$, $\mathcal{GG}$ has % lower node coverage owing to its focused nature of search. On the other hand, both of them have a pretty high level of redundant node visits. This is probably due to the high clustering coefficient of the final network formed by community formation. Initially, while the CC of the network is about 0.027, finally the CC grows to around 0.051. This rapid growth



(a) Total number of nodes visited during search (b) Number of repeated visits made

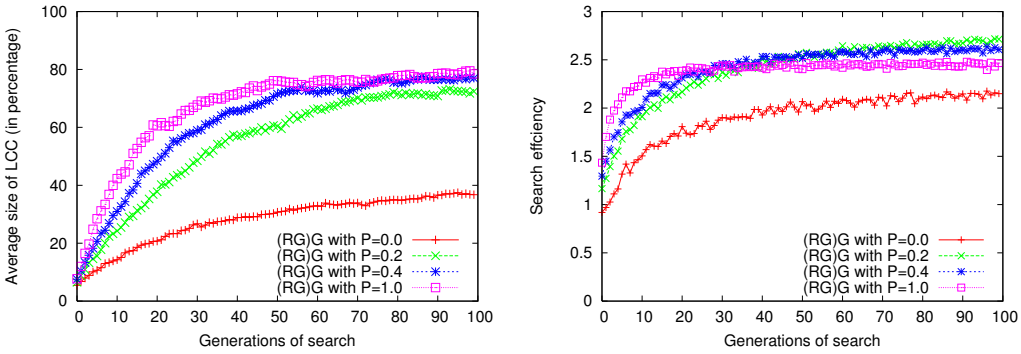
Figure 4.6: Total node coverage and redundant node coverage in $\mathcal{RG}$ and $\mathcal{GG}$

in CC is due to the large number of edges added between the similar nodes. Since a larger percentage of a nodes neighbors are neighbors themselves, continuous proliferation within a community is expected to lead to a large amount of redundant node visits.

4.3 Probabilistic $\mathcal{RG}$ and $\mathcal{GG}$

This is a naive attempt that was attempted in order to incorporate the characteristics of both $\mathcal{RG}$ and $\mathcal{GG}$, in a single algorithm. This was done by the following method. From the algorithm point of view, the only difference between $\mathcal{RG}$ and $\mathcal{GG}$ is the method of General Walk - $\mathcal{RG}$ walks randomly and $\mathcal{GG}$ walk greedily. Hence, to perform the functions of both random and greedy general walk, each non-matching node through which the packets pass, will either forward the packet randomly (like $\mathcal{R}*$) or greedily ($\mathcal{G}*$). In the matching nodes, the behavior is always the same - greedy proliferation (as in $*G$). The probability can be set to different values between 0.0 and 1.0 to get a behavior in between pure $\mathcal{RG}$ and pure $\mathcal{GG}$.

This is termed as *Probabilistic $\mathcal{RG}$ and $\mathcal{GG}$* or in short, $(\mathcal{RG})_P$, where P is the probability of random walk. This scheme is simulated in the same framework as earlier simulations, with the same parameter values. Different probabilities varying from 0.0 to 1.0 is selected for simulations, whose results are shown in the adjoining figures. The results reinforce the above argument regarding the community formation by random and greedy general walk. The relative size of the largest connected component (LCC) in the subgraph induced by nodes of each profile (Fig. 4.7(a)) shows that greater the percentage of random walk, faster is the process of community formation. In terms of search efficiency, P =0.2 performs the best, even better than pure $\mathcal{RG}$ and $\mathcal{GG}$ (Fig. 4.7(b)).



(a) Average size of LCC (b) Search Efficiency

Figure 4.7: Performance of different probabilities of random walk in $(\mathcal{RG})\mathcal{G}_P$ Search

This is contrary to the expectations, as it is intuitive that probabilities between 0.0 and 1.0 should produce efficiencies lying between pure $\mathcal{RG}$ and $\mathcal{GG}$. But, analyzing the underlying dynamics now it is obvious why $(\mathcal{RG})\mathcal{G}_{0.2}$ performs better. On one hand, random walk of $P = 0.2$ is sufficient to produce the best possible community structure, on the other hand, this has the maximum probability (that is 0.8) of efficiently searching the community structure by completely greedy walking (as in $\mathcal{GG}$). This logic fails to hold in the $(RG)G_{0.0}$ due to the sub-optimal community structure. Also, in terms of fraction of similar nodes returned per search, $(\mathcal{RG})\mathcal{G}_{0.2}$ performs better than the rest.

These results doubtlessly confirm that the higher the percentage of random walk, faster is the development of the community structure. Once the community structure has been formed, the higher the percentage of greedy walk, the better is the search efficiency and unless the community structure that is formed is optimal, the search fails to be efficient enough to produce the best possible results. Therefore, to summarize, while a random general walk has a better performance in terms of node discovery and node retrieval, greedy general walk is better at efficiently searching the already discovered nodes. Hence, it will be beneficial if we are able to develop a search algorithm that embraces the best of both.

Chapter 5

An Approach to Self-Adjusting Search ($\mathcal{SA}$)

We wish to design an algorithm that has the intelligence to adjust itself between two phases - Exploratory Phase and Search Phase. In terms of our problem, our search algorithm should, in the initial stages, explore the graph with maximum probability (for developing the best community structure as soon as possible) and in the later stages search the network with maximum efficiency. In other words, it must be able to identify automatically whether it should put the maximum effort in exploring the network or in searching the network efficiently. We propose such an algorithm in the next section.

5.1 The Final Algorithm

As evident in earlier results, $\mathcal{RG}$ performs a better exploration of the network, while $\mathcal{GG}$ performs a better search of the already explored regions. Each of the algorithms is individually suited for each of the two phases, respectively. So we need to design an algorithm that can adjust itself based on the phase of the system, in a decentralized manner. The key requirement for designing such an algorithm is to identify a property / parameter in the network based on which we can control the randomness / greediness of the search process.

In order to make the search tunable to random or greedy schemes, each query packet now holds another parameter - *Random Walk Probability* (P). At the time of initiation of the search, the value of the probability is set by the initiator node. This probability is also copied to the new packets created at the time of proliferation. Based on this probability, similar to $(\mathcal{RG})\mathcal{G}$, the non-matching nodes through which the packets pass, will

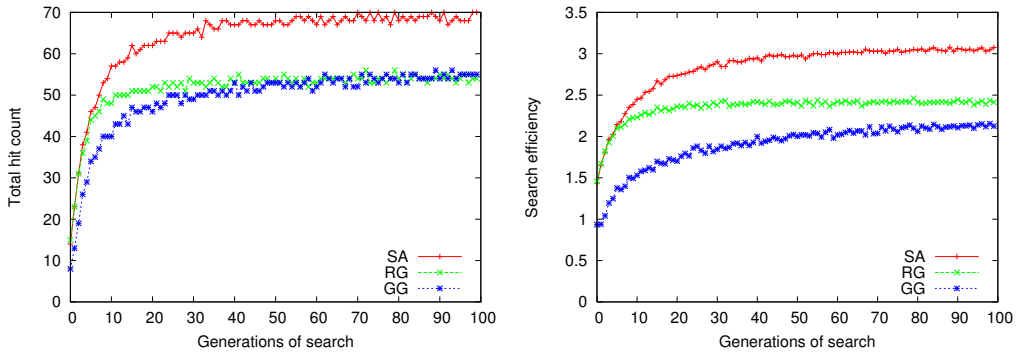
either forward the packet randomly or greedily. Depending on the probability values, the search exhibit a behavior in between pure $\mathcal{RG}$ and pure $\mathcal{GG}$.

Next, a suitable parameter need to be chosen for determining the phase of the system in a decentralized manner. We have chosen this to be the X value of the node. If X is low, then it means that the node has the capacity of accepting new community edges and expanding the community structure. In that case, it should try to explore the network for previously undiscovered similar nodes with a higher probability. Conversely, when X is high and near its limiting value, its capacity of adding to the community structure is low. Therefore, instead of exploring, it should try to efficiently search the community structure that has already been formed around it. More formally, the probability of random walk is calculated as

$$P(random\ walk) = 1 - \frac{X_A}{X_{max}}$$

where $X_A = X$ of the node A that is initiating the search. The overall behavior would be as we desire - initially, when X is 0 for all the nodes, it will behave like pure $\mathcal{RG}$. Later as the X of all the nodes reach X_{max}, the probability of random walk reduces to zero, that is, it performs pure $\mathcal{GG}$ on an optimal community structure.

There are a number of other modifications that are made in this final version. They are enumerated as below.



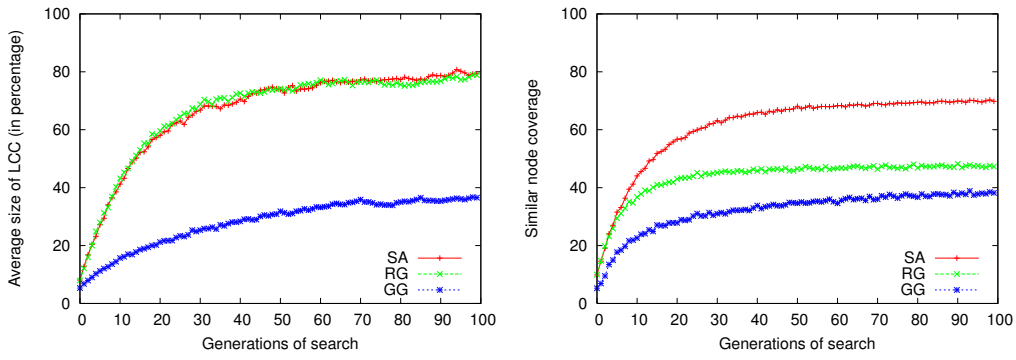
(a) Total hit count (b) Search Efficiency

Figure 5.1: Performance of $\mathcal{SA}$ wrt Hit count and efficiency

- *Low Clustering Coefficient*: As we had hypothesized, the high redundant node visits was due to increase in the clustering coefficient of the graph. Hence the clustering coefficient can be reduced by means of a very simple rule. The rule is

- a commmunity edge is added between two similar nodes if they do not share a common neighbor. In this way an attempt is made to eliminate triangle as much as possible, that will hopefully reduce the probability of the excessive proliferations in very close vicinity, and thus reduce the chances of redundant node visits.

- *Disassortative Network*: Assortative networks are those networks in which high degree nodes primarily connect to high degree nodes and similarly, low degree nodes connect to low degree nodes. Diassortative networks are just the opposite - high degree node connecting to low degree nodes, and vice versa. In literatures related to epidemics, it is well known that diassortative networks lead to faster spread of diseases. This is in a way intuitive. If a disease is initiated from an already high degree node, then it will obviously spread rapidly. But in a disassotative network, even if the disease starts from a low degree node, it will very soon reach a high degree node (as low degree node has high nodes as neighbors) and hence spread rapidly. This principle can be applied in P2P network that we have assumed too, in the following manner. Earlier, while deleting an edge, any random edge was being deleted. Rather than choosing edges randomly, we can preferentially choose the edge to that neighbor that has similar degree as the current node. In other words, the edge to the neighbor having the minimum difference in degree to the node in consideration is deleted. This naturally gives rise to a disassortative network, as the lower degree node will have higher degree nodes after deletion, and vice versa. Experimentations have shown that it boosts the search efficiency by a further 7%.



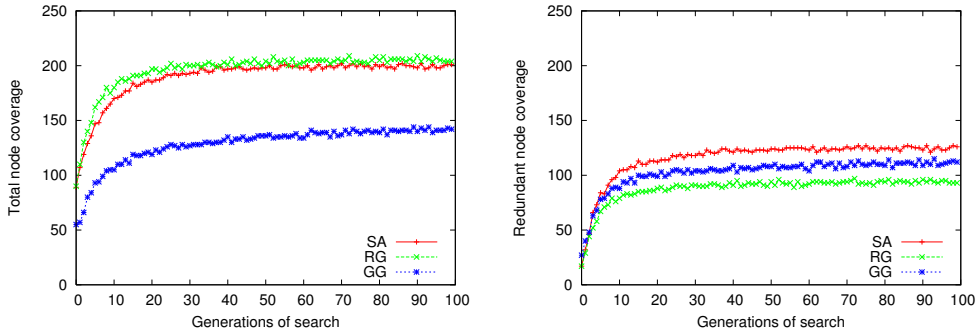
(a) Average size of LCC (b) Similar node coverage

Figure 5.2: Performance of $\mathcal{SA}$ wrt LCC and similar node coverage

After, these modifications the detailed results of the final algorithm are as follows.

5.2 Simulation Results

Figures 5.1(a), 5.1(b), 5.2(a) and 5.2(b) reflect the superiority of $\mathcal{SA}$ scheme. The scheme is able to produce the best possible community structure as fast as $\mathcal{RG}$. Side by side, it overcomes the shortcomings of $\mathcal{RG}$ by being able to find almost all the similar nodes in the LCC. Refer Fig. 5.2(a) and 5.2(b), $\mathcal{SA}$ is finding around 90% of the similar nodes that constitute the LCC, while $\mathcal{RG}$ returns just around 60% of all such nodes, thus producing almost 50% improvement. Finally, we find that the search efficiency of $\mathcal{SA}$ (Fig. 5.1(b)) is about 30% better than $\mathcal{RG}$ (and more than 130% better than $\mathcal{RR}$ with REA).



(a) Total number of nodes visited during search (b) Number of repeated visits made

Figure 5.3: Total node coverage and redundant node coverage in $\mathcal{RG}$ and $\mathcal{GG}$

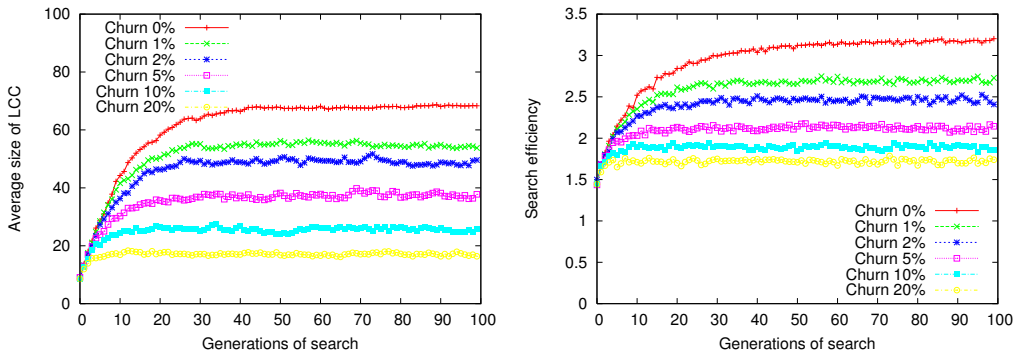
On the other hand in terms of total nodes visited during search (Fig. 5.3(a)), $\mathcal{SA}$ is covering equal number of nodes as $\mathcal{RG}$, and yet the former brings in a higher number search results than the latter.(Fig. 5.1(a)). This could also be another alternate measure of efficiency of the search, in which too, $\mathcal{SA}$ excels over the others. But in terms of redundant node coverage, $\mathcal{SA}$ marginally exceeds both $\mathcal{RG}$ and $\mathcal{GG}$ in spite of the reduction in the clustering coefficient. This shows that the reduction of the clustering coefficient fails to reduce the unnecessary node visits. But overall, $\mathcal{SA}$ exceeds the performance of the previous algorithms in all respects.

5.3 Performance under Node Churn

One of the primary characteristics of the real world P2P networks is the amount of churn its nodes undergo i.e. a large number of nodes continuously enter and leave the network. It is for this reason structure networks like Chord, Pastry and CAN, fail to perform under such scenarios, as the overhead of maintaining the structure under this

churn is high. Hence, any proposed search algorithm must be evaluated in terms of its performance under node churn.

In order to test the performance of $\mathcal{SA}$ under node churn, the model by which the nodes are churned must be defined. This is done in the following manner. The amount of churn is defined as the *C%* of nodes that are *resetted* after every generation of search. Resetting a node is defined as deleting its community edges (leaving the connectivity edge intact) and changing its profile. Effectively, this is similar to a new node with a new profile joining the network by connecting using the connectivity edges. Regarding the



(a) Average size of LCC (b) Search efficiency

Figure 5.4: Performance of $\mathcal{SA}$ wrt LCC and search efficiency under different amount of node churn

new profile, if it is selected arbitrarily, then the profile distribution in the whole network is going to change significantly after the churn. Since in a real world network, in spite of churning, the approximate distribution of the interest / content over the categories remains the same, changing of the profile distribution is not desirable. The solution is to exchange the profiles of the *C%* nodes undergoing churn among themselves, which assigns each of them with a new profile as well as keeps the profile distribution same.

Based on this scheme of node churn, the algorithm $\mathcal{SA}$ was evaluated on 1%, 2%, 5%, 10% and 20% churn per generation. The results are shown in 5.4(a) and 5.4(b). It is fairly evident that there is a graceful degradation in the performance with increasing amount of node churn. As the churn increases, the quality of community structure (measured by LCC) falls, along with the associated search efficiency. Important point to notice is that there is a significantly drop in just 1% node churn.

Incidentally, there might be a way in which the performance under node churn can

be boosted, though at the cost of increased protocol overheads. Recall that there is a parameter P_{add} which is the probability that an edge is added between two similar nodes when it found during a search. This parameter decides at what rate will the edges be added to the system. As aforesaid, a higher value of this means that there will a higher number of edges that we be considered for addition and almost immediate deletion (as in the later generations, most of the nodes are too saturated to accommodate any new edge, thus forcing deletion). Since this manipulation of edges require the similar nodes to communicate among each other. Hence, the higher number edges being added to the system, the higher is the protocol overhead. Again too low a value will make the community formation process too slow. Hence, we had kept P_{add} to an intermediate value of 0.3. But in a system that has too much node churn, it might be of a greater necessity to repair the community structure as soon as possible, by increasing the rate / probability of edge addition, while accepting an increased protocol overhead. It is expected that increasing the probability, the efficiency of the search under node churn will increase. In order to test it out, $\mathcal{SA}$ was performed with p_{add} varying from 0.1 to 0.9, in steps of 0.2, while undergoing a node churn of 5% and 20% per generation. The results are as follows.

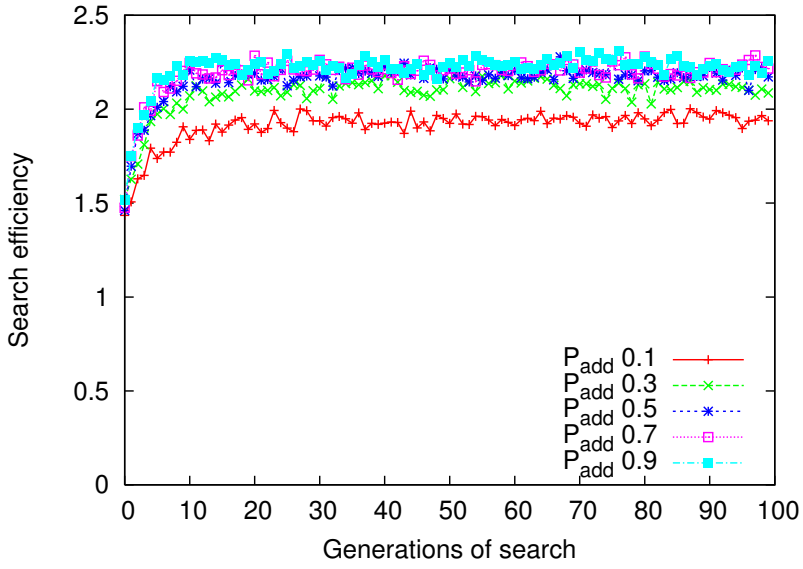


Figure 5.5: Performance of $\mathcal{SA}$ wrt search efficiency under 5% churn with different P_{add}

Referring to Fig. 5.4 and 5.5, the behavior of search efficiency with respect to various P_{add} is similar for both moderate (5%) and high (20%) node churn. Between P_{add} 0.1 and 0.3, there is a significant increase in the efficiency. But beyond 0.3, there

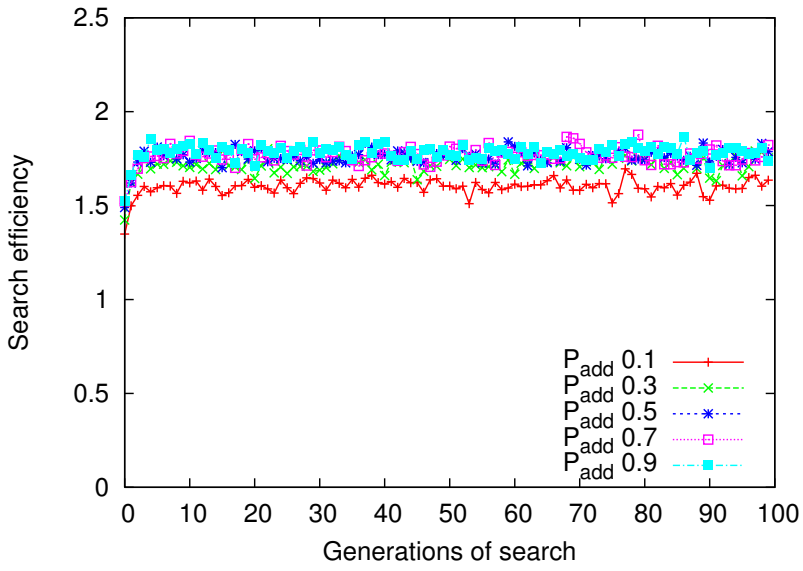


Figure 5.6: Performance of $\mathcal{SA}$ wrt search efficiency under 20% churn with different P_{add}

is only marginal increase in the efficiency. On the other hand, the protocol overhead increases linearly with respect to P_{add}. Hence, it can be concluded that $P_{add} = 0.3$ is sufficiently good value, taking into concern both efficiency as well as protocol overhead, as further increasing P_{add} results in only marginal increase in efficiency, but significant increase in protocol overhead.

5.4 Performance under Overlapping Interest Categories

Till now, each profile was considered to be dissimilar to each other. But as explained in section **??**, the categories of interest/content are inherently overlapping by nature as any particular interest will always have a certain degree of similarity with multiple interest categories. In order to develop a search algorithm that can perform well under realistic scenarios, it should be tested against a P2P system that has such overlapping categorization of interest. In the P2P model that has been used in this work, incorporating the notion of "overlap" (i.e. categories having certain degree of similarity between them) is very simple. Hence, we refine our similarity measure between profiles as follows.

Let there be two profiles P_1 and P_2. These two profiles are said to be *similar* if their Hamming Distance is within a particular threshold, i.e. $HD(P_1,P_2) <= HD_{thresh}$. HD_{thresh} can have values between 0 and d. In our simulations till now, the threshold

was effectively 0, i.e. two profiles were similar if their hamming distance was exactly zero. And, community edges connected nodes whose profiles were exactly equal. Now,

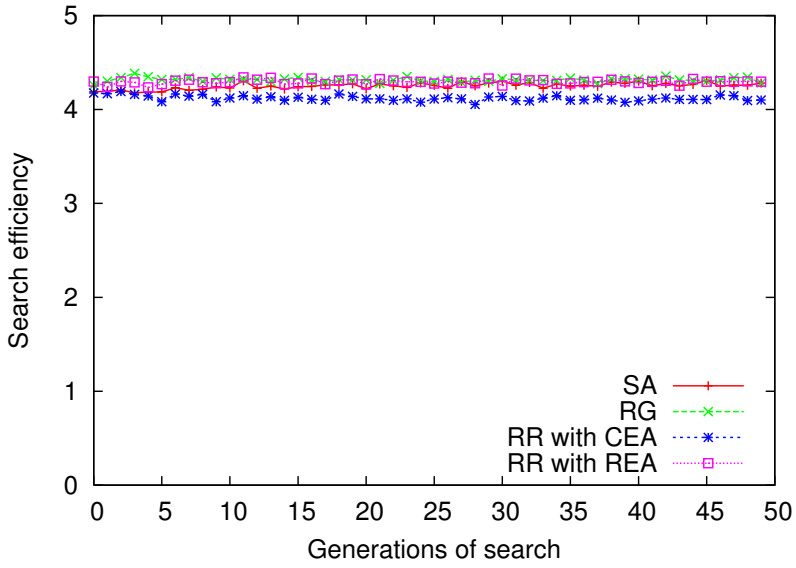


Figure 5.7: Performance of $\mathcal{SA}$ wrt search efficiency under overlapping profiles

in order to, simulate overlapping categories, we increase this threshold to 1. Hence, a profile P, besides matching with P itself, will match with d other profiles, which are at a 1 bit Hamming Distance from P. Rest of the walk strategies remain the same. Simulation results of the explored algorithms in this new scenario are shown in Fig. 5.4. It was very surprising to find that none of the algorithms perform significantly different from each other. In fact, $\mathcal{RG}$ and even $\mathcal{RR}$ with REA seems to be performing marginally better than others.

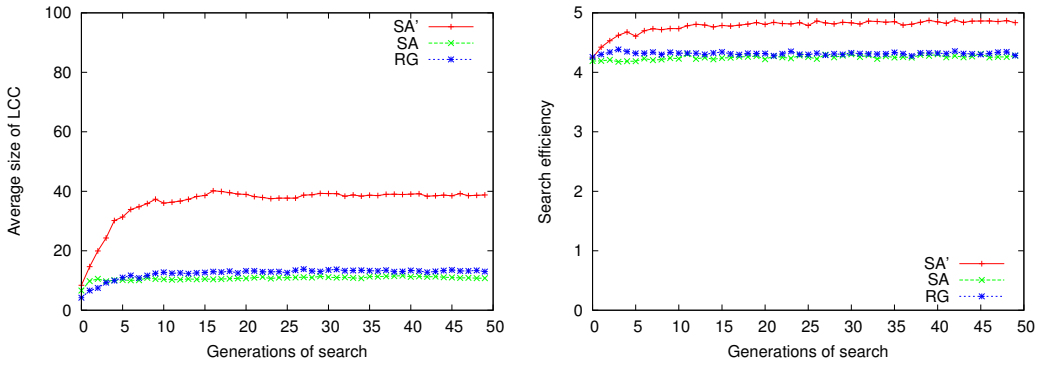
One of the main reason to which this can be attributed to, is the increased number of similar nodes. Unlike before, the number of nodes to which a node is similar is now about d times more than earlier, as it is similar to d new profiles. Hence, even by $\mathcal{RR}$, the search efficiency produced is sufficiently high; high enough that it can hardly be further increased any further. Yet, in the following section, an attempt has been to boost the performance in this scenario still higher.

5.4.1 Modification ($\mathcal{SA}'$)

In order to solve this problem, a small modification was made to the edge deletion strategy. In $\mathcal{SA}$, to make the network disassortative, the edge to the neighbor having the

minimum degree difference from the concerned node was deleted. Here, no consideration was made for the similarity of the nodes profiles with respect that of its neighbors. In fact, since the H_{thresh} was 0, all the neighbors of a node had the same profile as the node itself (since community edges connected only exactly similar nodes). But now, with $H_{thresh} = 1$, the neighbors can have profiles which is not exactly same. Then the question arises that how does the similarity with the neighbor's profile affect the search process.

To analyze this, let us consider two neighbors of node x, u and v. $HD(P_u, P_x) = 1$ and



(a) Average size of LCC (b) Search efficiency

Figure 5.8: Performance of $\mathcal{SA}'$ wrt LCC and search efficiency under overlapping profiles

$HD(P_v, P_x) = 0$. Let us consider the second neighbors of x through its neighbors u and v. The neighbors of u have profiles that are within one hamming distance of P_u. Hence, only if the profile of the neighbor of u be equal to P_x, then will it be similar to x (otherwise it will be two hamming distance away from P_x). But, if we consider the neighbors of v, all the neighbors of v are similar to x (obvious as their profiles are similar to $P_u = P_x$). Therefore, it is evident that v is preferred over u as a neighbor as it gives a higher fraction of second neighbors that are similar to node x. In other words, it is desirable to keep those nodes are neighbors that has the highest similarity with the concerned node.

The new rule for edge deletion is as follows.

- Among all the community edges, the edge to the neighbor having the least similar profile is deleted.
- If all neighbors have the same profile, then the neighbor having the least degree difference is deleted (disassortative).

Simulation results of this scheme shows that this modification is indeed able to boost the efficiency of the search further. Figure 5.8(a) shows that $\mathcal{SA}'$ can amass almost 40% of all the *similar* nodes in the LCC, compared to barely about 10 to 13% in case of $\mathcal{SA}$ and $\mathcal{RG}$. Finally, it is more than 12% more efficient than all the rest.

Chapter 6

Evaluation of P2P Communities: An intuitive approach

As already discussed in the section 2, existing community detection literature lacks a proper scheme for evaluating P2P communities taking into consideration its overlapping nature. Hence, inspiration is derived from the field of document clustering, where the partition compactness measurement used in the *Average Scattering Coefficient* [25], [26] proved to be a good starting point for building the required evaluation metric for P2P community structures. The definition of this scheme along with the evaluation of the explored algorithms using this scheme have been given below.

6.1 Definition

This is an intuitive approach that was made in order to evaluate the overlapping nature of P2P communities. The details are as follows. The aim of any community formation scheme is to bring the nodes that have similar interests / content closer to each other in terms of link distance in the network, in order to easily find them during search. In terms of the defined P2P model, it is desired that the nodes having similar *profiles* come to closer to closer to each other, so that finding one of them

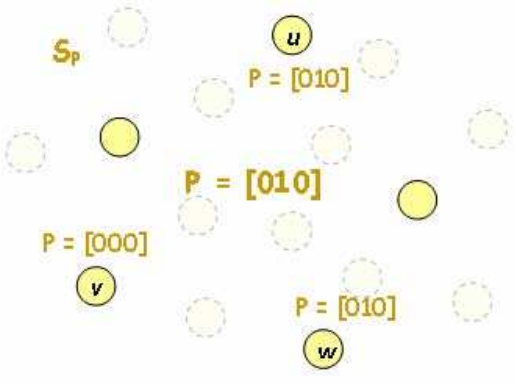


Figure 6.1: Schematic representation of the community evaluation scheme

enables us to quickly find all of them.
Here, similarity between nodes is measured by the Hamming Distance (HD) of their profiles. Now, given an information profile P, let this set of nodes be S_P i.e. $S_P = \{u : HD(P_u, P)\}$, where u is a node in the network, and P_u is its information profile. As foretold, it is desired that the average internodal distance $\delta_{u,v}$ between any two nodes u, v in the community defined by S_P to be low.

But there is a catch - we also have to consider the similarity in the profiles of u and v when calculating this metric. Let us take the case of three nodes u,v,w εS_P. As shown in Fig. 6.1, let us assume that their profiles to be $P_u = [010]$, $P_v = [000]$ and $P_w = [010]$. Also, $\delta_{u,v} = \delta_{u,w}$. Now, while calculating the community structure metric for $P = [010]$, it is intuitive that the path distance of between u and w carries more importance than that between u and v, as the former pair of nodes have exactly the same profile as the P for which we are calculating. Since $\delta_{u,v} = \delta_{u,w}$, $\delta_{u,w}$ should be given higher weight than $\delta_{u,v}$. Hence, we weigh the distance between each pair of nodes with their profile similarity with the profile for which we are calculating the metric. In other words, we define $\delta'_{u,v}$ such that

$$\delta'_{u,v} = \delta_{u,v} \times fn\left(HD(P_u, P), HD(P_v, P)\right)$$

We define the function fn as $1 + \frac{HD(P_u,P)+HD(P_v,P)}{2d}$ thus producing

$$\delta'_{u,v} = \delta_{u,v}\left[1 + \frac{HD(P_u, P) + HD(P_v, P)}{2d}\right]$$

where d = no of bits used to represent a profile. When a pair of node u and v have profiles exactly same as P, then $\delta_{u,v}$ remains the same, otherwise the effective path distance between them increases. $\delta'_{u,v}$ is averaged over all pairs of nodes in S_P.

$$\Delta_P = \frac{1}{|S_P|^2} \sum_{u,v\ \varepsilon S_P} \delta'_{u,v}$$

Hence, the final metric is calculated by adding the Δ_P for all the d^2 profiles. In order to give more importance to the more popular profiles, the Δ_P is weighted by the relative frequency of the profile P. Hence the final metric is calculated as

$$\Delta = \sum_{\forall P} \Delta_P \frac{N_P}{N}$$

where N = number of nodes in the network and N_P = number of nodes having profile P. A decrease in the value of Δ reflects the fact that the similar nodes in the network have

moved closer to each other.
In essence, in this community evaluation scheme, unlike other schemes, the P2P community is defined not based upon the link structure but based upon the similarity of the nodes. Because each profile P is similar to a number of profiles, a node having a profile P is defined to participate in communities of that many profiles. This scheme is able to evaluate overlapping P2P communities by incorporating the original reason behind it in the scheme, i.e. the overlapping nature of the P2P content / interest categories.

At a high level, this scheme provides an idea of the average distance between similar nodes in the network. But more specifically, it is necessary to define a few benchmark values based upon which the goodness of a community structure can be defined. In order to define such benchmark values, two hypothetical community structures were defined. They are defined as follows.

- *Perfect Community* - All the nodes in a network of N nodes have the same profile P and are connected to each other, forming a complete graph / clique. This produces a value of 1 by the evaluation scheme. Since, this is the best possible community structure that can be imagined, 1 is the lower bound of all values possible by any community structure.

- *Ideal Community* - Given a particular distribution of profiles, an hypothetical ideal community structure can be defined. In order to do so, the properties of this hypothetical community structure needs to be defined. They are as follows. In an ideal community structure, it is desired that the more similar nodes are closer to each other than that less similar nodes, in terms of their link distances. In other words, the distance $\delta_{u,v}$ between two nodes u and v should be proportional to their dissimilarity i.e. $HD(P_u, P_v)$. With respect to the equations stated earlier in this section, $\delta_{u,v}$ maybe defined as following

  $$\delta_{u,v} = 1 + c.HD(P_u, P_v)$$

  Here the most similar nodes ($HD(P_u, P_v) = 0$ are at hypothetical distance 1 apart. The maximum value of $HD(P_u, P_v)$ is d as defined earlier. On the other hand, the maximum value of $\delta_{u,v}$ should be the diameter of the network. Let this be defined as δ_{max}. Hence, we define the constant c as $\frac{\delta_{max}}{d}$ and the equation for $\delta'_{u,v}$ is defined as

  $$\delta'_{u,v} = \left(1 + \frac{\delta_{max}}{d}.HD(P_u, P_v)\right).\left[1 + \frac{HD(P_u, P) + HD(P_v, P)}{2d}\right]$$

In this way, given a network of diameter δ_{max} and a distribution of profiles among the nodes, a hypothetical ideal community can be defined. The community structure value of this network is a benchmark value against which the value of the actual network can be compared.

6.2 Evaluation of Explored Algorithms

Finally, the algorithms that have been explored earlier are compared with each other by evaluating their community structure with this new scheme. Referring to Fig. 6.2, the performance of $\mathcal{RR}$ with REA is naturally the worst, random addition of edges does not take any consideration of the similarity of nodes. The decrease in the metric value in this case is simply because of the general reduction of average internodal distance between all nodes (irrespective of similarity) due to randomly added edges. $\mathcal{RG}$ and $\mathcal{RR}$ with

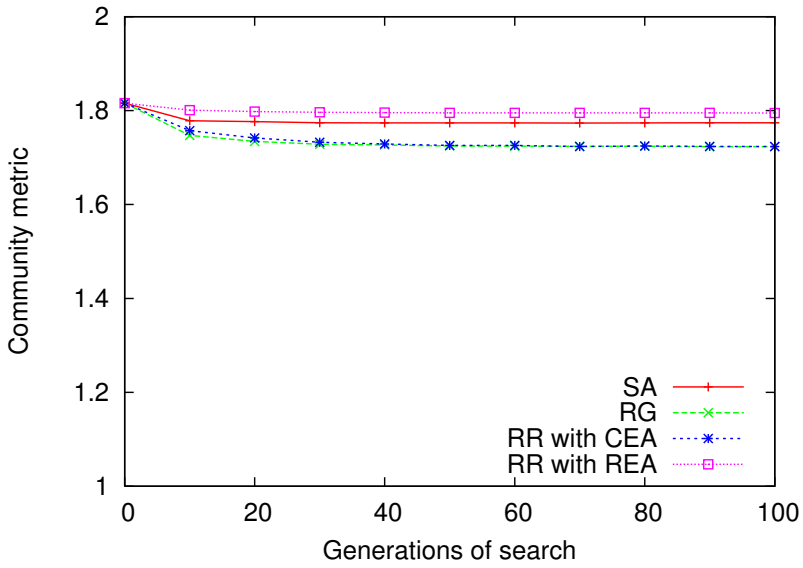


Figure 6.2: Performance of $\mathcal{SA}$ wrt new community evaluation scheme

CEA, both form equally good community structures with respect to this metric. This is obvious as both them is able to explore the network widely due to its random general walk, and there connect together similar nodes that were earlier wide apart. Finally, $\mathcal{SA}$ performs a little less in terms of its community formation. The reason behind this maybe its clustering coefficient reduction scheme. This scheme prevents similar nodes that are two hop apart from getting directly connected. Hence, with a very high probability of $\delta_{u,v} \geq 2$ between two similar nodes u and v. Hence the community metric value increases to a value closer to 2. Yet, it is able to search more efficiently (refer to section

), as unlike $\mathcal{RG}$ and $\mathcal{RR}$, it is able to find a much larger fraction of nodes belonging to the community than the latter schemes.

Chapter 7

Future Work

Further work is necessary to take this search algorithm to a stage where it can be implemented on a practical platform in a large scale. Some areas where work in the future is necessary are as follows.

- Comparison of this random walk scheme with existing practical pure random walk and flooding based schemes (like that used by Gnutella), along with other semi structured search frameworks like Freenet and Symphony
- Measuring the performance of the search algorithm under Gnutella like topology, which is similar but has slight differences from Albert-Barabasi power law topology.
- Measuring the performance in a P2P model where a node's search query may not fall in the same category as the node's interest/content.
- Development of a mathematical frame work to theoretical model the dynamics of the random walk with proliferation based search
- Development of an abstract idea of an ideal community structure that would be provide a standard benchmark for estimating the goodness of a community structure by the new proposed community metric

Chapter 8

Conclusion

In this work, an efficient community based search algorithm has been proposed which is able to outperform many of the existing algorithms. It has been developed after thorough understanding of the dynamics of search based on random walk with proliferation. Along with this, a new scheme for evaluation of overlapping community structures has been proposed. Rigorous testing in different environments and fine tuning is necessary before practical deployment is possible.

Bibliography

[1] Niloy Ganguly, Geoff Canright and Andreas Deutsch. *Design Of An Efficient Search Algorithm For P2P Networks Using Concepts From Natural Immune Systems*, in the 8th International Conference on Parallel Problem Solving from Nature (PPSN VIII). Birmingham, UK, 2004.

[2] Niloy Ganguly, Geoff Canright, Andreas Deutsch. *Design of a Robust Search Algorithm for P2P Networks*, in the 11th International Conference on High Performance Computing. Bangalore, India, 2004.

[3] A. Asvanund, R. Krishnan. *Content-Based Community Formation in Hybrid Peer-to-Peer Networks*, Proceedings of the SIGIR Workshop on Peer-to-Peer Information Retrieval, 2004

[4] M. Khambatti, K. Dong Ryu, P. Dasgupta. *Structuring Peer-to-Peer Networks Using Interest-Based Communities*, Databases, Information Systems, and Peer-to-Peer Computing, 2003

[5] M. Khambatti, K. Dong Ryu, P. Dasgupta. *Peer-to-peer communities: Formation and Discovery.*, 14th IASTED International Conf. Parallel and Distributed Computing, (PDCS02), 2002.

[6] AL Barabsi, R Albert. *Emergence of Scaling in Random Networks*, Science, 1999

[7] M Ripeanu, A Iamnitchi, I Foster. *Mapping the Gnutella Network - Properties of Large-Scale Peer-to-Peer Systems and Implications for System Design*, IEEE Internet Computing, VOL 6; NUMB 1, pages 50-57, 2002

[8] M Faloutsos, P Faloutsos, C Faloutsos. *On power-law relationships of the Internet topology*, Computer Communication Review, VOL 29; NUMBER 4, pages 251-262, 1999

[9] Niloy Ganguly, Lutz Brusch, Andreas Deutsch. *Design and analysis of a bio-inspired search algorithm for peer to peer networks*, in the post proceedings of the workshop SELF-STAR: Self-* Properties in Complex Information Systems. 2005

[10] Ozalp Babaoglu, Geoffrey Canright, Andreas Deutsch, Gianni Di Caro, Frederick Ducatelle, Luca Gambardella, Niloy Ganguly, Mark Jelasity, Roberto Montemanni, Alberto Montresor. *Design Patterns from Biology for Distributed Computing*, ACM Transaction of Autonomous and Adaptive Systems, Vol 1, Issue 1. September 2006.

[11] Sachin Kulkarni, Niloy Ganguly, Geoffrey Canright, Andreas Deutsch. *A bio-inspired algorithm for location search in peer to peer network*, "Advances in biologically inspired information systems: models, methods, and tools". Falko Dressler & Iacopo Carreras (ed) Springer series in Computational Intelligence.

[12] Erica Klarreich. *Inspired by immunity*, Nature 415, 468-470, 2002.

[13] Dickinson RB, Tranquillo RT. A *Stochastic Model for Adhesion-mediated Cell Random Motility and Haptotaxis*, J. Math. Biol. 1993;31(6):563-600.

[14] JD Murray. *Mathematical Biology*, Springer-Verlag, 1989

[15] German Sakaryan and Herwig Unger, *Influence of the decentralized algorithms on topology evolution in P2P distributed networks*, in Design, Analysis, and Simulation of Distributed Systems (DASD), pages 12-18. Orlando, USA, 2003.

[16] Anna Puig-Centelles, Oscar Ripolles and Miguel Chover Centelles, *Reviewing P2P network coummunity detection*, in the IASTED European Conference on Internet and Multimedia Systems and Applications (EuroIMSA). Chamonix, France, 2007.

[17] V. Cholvi, P. A. Felber, and E. W. Biersack, *Efficient search in unstructured peer-to-peer networks*, in Proceedings of the Sixteenth Annual ACM Symposium on Parallelism in Algorithms and Architectures (SPAA). Barcelona, Spain, 2004.

[18] Hanhua Chen and Hai Jin, *Identifying Community Structure in Semantic Peer-to-Peer Networks*, in Proceedings of the 2nd International Confernece on Selmantic, Knowledge and Grid (SKG). Guilin, China, 2006.

[19] Farnoush Banaei-Kashaniy and Cyrus Shahab, *Criticality-based Analysis and Design of Unstructured Peer-to-Peer Networks as Complex Systems*, in Proceedings

of the 3rd IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid). Tokyo, Japan, 2003.

[20] *Gnutella*, www.gnutella.com

[21] *CFinder*, http://angel.elte.hu/clustering/

[22] P. Pollner, G. Palla, T. Vicsek. *Preferential attachment of communities: the same principle, but a higher level*, Europhysics Letters, 73:478, 2006

[23] G. Palla, I. Dernyi, I. Farkas and T. Vicsek. *Uncovering the overlapping community structure of complex networks in nature and society*, Nature, 435:814, 2005

[24] M. E. J. Newman, *Assortative Mixing in Networks*, Phys. Rev. Lett. 89, 208701, 2002

[25] Mohamed Bouguessa, Sheng-Rui Wang. *A new efficient validity index for fuzzy clustering*

[26] Richard Forster. *Document clustering in large german corpora using natural language processing*. PhD Thesis, 2006

[27] Xiuqi Li, Jie Wu. *Searching Techniques in Peer-to-Peer Networks*, in Handbook of Theoretical and Algorithmic Aspects of Ad Hoc, Sensor, and Peer-to-Peer Networks. Auerbach, New York,USA, 2006