Masters Project Synopsis

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

April 30, 2008

1 Introduction & Motivation

Efficient search in unstructured P2P networks is becoming a real challenge, as the P2P user base is increasing exponentially. 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. The idea essentially is to go for a community-based semi-structured network. 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. In this approach too, there are a number of alternatives possible. The search mechanisms may vary from flooding based searches to random walk based searches. Each different scheme has its own pros and cons. Community formation can boost the efficiency of all the search schemes, each to a different degree. There has been a number 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 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. This is the challenge that motivated this work.

The idea of forming P2P communities to improve search efficiency is an ongoing research field. Many groups have explored this concept with very specific types of networks (distributed libraries for example [3]) while other have applied it to Erdos-Renyi networks [4]. However, except some of our previous works by Prof. Ganguly [2],[1], there has hardly been work on algorithms where the search process itself triggers community formation. Prof. Ganguly's work explored the concept of community formation by restructuring the neighborhood in a grid-like topology. The algorithms developed there could not be ported in a more realistic power law network. Hence, completely new algorithms have been developed based upon a much more thorough understanding of the effect of various dynamics performed on the network. Different varieties of random and greedy search mechanisms are applied in order to understand the dynamics. Finally, an algorithm is suggested which consists of a healthy mix of random and greedy walking.

In the rest of the synopsis, Section 2 describes the model of the P2P network that we use and the details of the various algorithms. Their performance in simulations is analyzed in section 3 and a final algorithm based on these results is proposed in section 4. Finally, we conclude in section 5 with the possible ways of improving these results.

2 Model and Algorithms

In this section, the model of the P2P network is defined, that has been used for this work. Then, the detailed implementation of the search and community formation algorithms are discussed.

2.1 Peer-To-Peer Model

The internet at large, as well as, popular P2P networks exhibit power law topology [7],[6], which is why a realistic power-law topology was assumed for our P2P network. Also, in order to form content-based communities, we have classified the information content of the peers into abstract subcategories. The details are provided below.

2.1.1 Topology & Network Load

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 the 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 the initial network 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}.

2.1.2 Profile Distribution

In a file sharing P2P network, each node shares some data with other nodes in the network. These data are categorized into abstract categories called *Information Profiles*. The profiles (P_I) therefore reflect the informational content as well as the informational interest of the user. A profile is represented in our system as a m-bit

binary value, thus producing 2^m distinct categories. These profiles are distributed among the nodes following Zipf's Law with the idea that some categories of data are highly popular whereas others are not.

2.1.3 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 into 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.

2.2 Algorithms

As indicated in section 2, four major types of the proliferation-based search algorithms – named $\mathcal{RR}$, $\mathcal{RG}$, $\mathcal{G}R$ and $\mathcal{GG}$, were tested out. In this section, these algorithms are described in full detail. As mentioned earlier, there are two distinct processes in the algorithms – Search and Community Formation.

2.2.1 Search

Any node in the networks can start a search query. Let the search be 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 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 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.

2.2.2 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 in the next search attempt. On the other hand, due to the network load limit of X_{max}, 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, deletion of edges is done 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. Furthermore, each edge is added with a probability of P_{add}. This regulates the speed of addition and prevents the network load of each node from reaching its limit very fast. Hence each node gets 'time' to learn and the network does not unnecessarily undergo a huge amount of churn to stabilize. 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.

2.3 Evaluation Criteria

We will like to evaluate the performance of these algorithms based on the following criteria.

2.3.1 Search related metrics

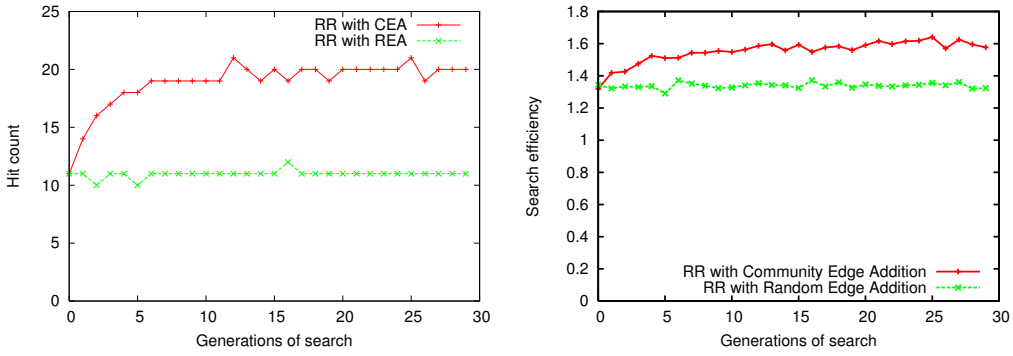
Let the i^{th} search produce a total of h_i search hits using a total of p_i packets. Let the total number of nodes similar to the initiator node (that is the maximum possible search hits) be H_i. Let the search be performed n times. Then the search-related metrics are defined as follows:
Total Hit 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. $\frac{1}{n}\sum_i \frac{h_i}{p_i}$
Similar Node Coverage Average fraction of all the similar nodes present in the network that is returned in each search, i.e. $\frac{1}{n}\sum_i \frac{h_i}{H_i} \times 100$.

2.3.2 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.

3 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 1: Performance of $\mathcal{RR}$ search using Community Edge Addition (CEA) compared to Random Edge Addition (REA)

3.1 Simulation Scheme

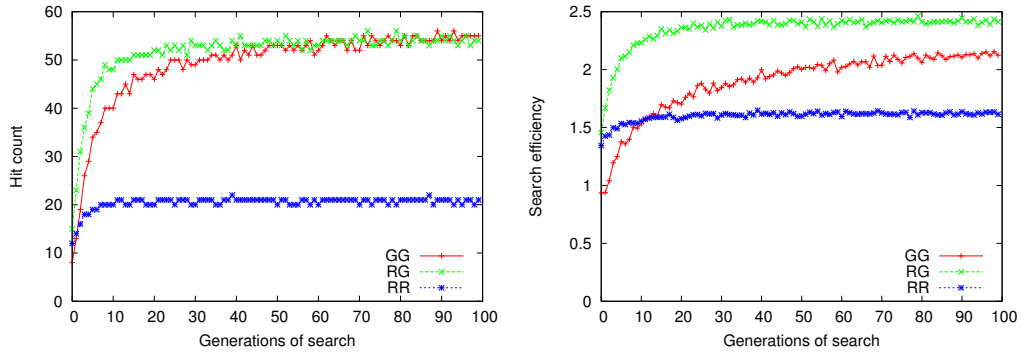
The algorithms were simulated on a power-law network of 1000 nodes, generated using the Barabasi-Albert preferential attachment method [5], which gave us a gamma of approx 2.0. 16 profiles ($m = 4$) were distributed among the nodes by Zipf's law with a gamma of 0.8. Each search query is propagated in 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. Multiple simulations are performed on different profile distributions for averaging the performance of the algorithm.

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). The results of these simulations follows next.

3.2 Results and Analysis

Comparing the performance of $\mathcal{RR}$ with community edge addition versus random edge addition on an equivalent graph, Fig. 1(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. 1(b)). This clearly proves that strategic addition of edges by community formation improves the search efficiency, unlike random addition edges.

Next we present the performance of $\mathcal{RG}$ and $\mathcal{GG}$ (the result of $\mathcal{GR}$ is omitted due

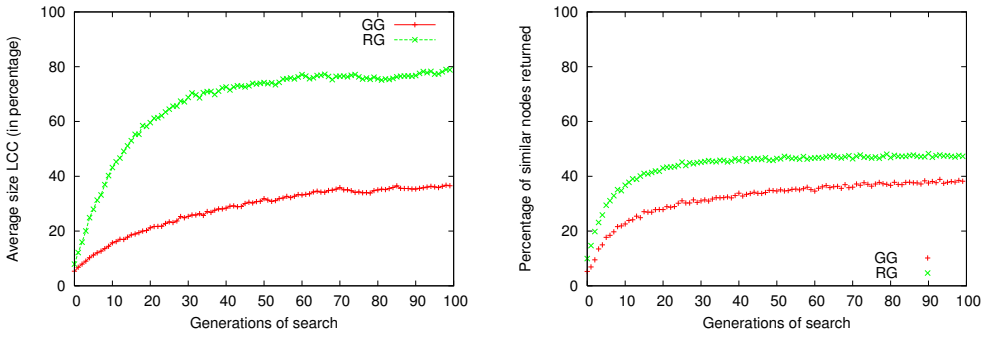


(a) Hit count (b) Search Efficiency

Figure 2: Performance of $\mathcal{RR}$, $\mathcal{RG}$ and $\mathcal{GG}$ wrt hit count and search efficiency

to lack of interesting inferences). All these cases undergo community edge addition. Figure 2(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 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.

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. 3(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. 3(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%).



(a) Average LCC of community formed (b) Percentage of similar nodes returned

Figure 3: Correspondence between LCC size and fraction of similar nodes returned in $\mathcal{RG}$ and $\mathcal{GG}$

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.

4 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.

4.1 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.

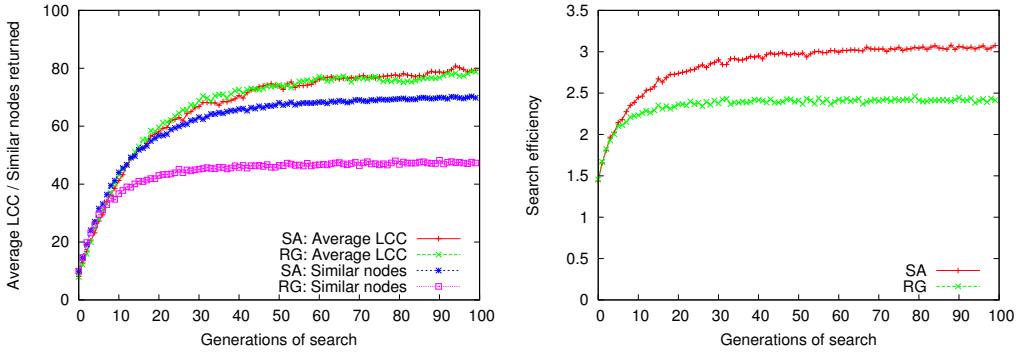
In order to make the search tunable to random or greedy schemes, each query packet now holds another parameter - *Random Walk Probability* (P). At the time of initiation of the search, the value of the probability is set by the initiator node. This probability is also copied to the new packets created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

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.



(a) Average LCC (b) Search Efficiency

Figure 4: Performance of Self-Adjusting Search

4.2 Simulation Results

Figures 4(a) and 4(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. 4(a), $\mathcal{SA}$ 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 $\mathcal{SA}$ is about 30% better than $\mathcal{RG}$ (and more than 130% better than $\mathcal{RR}$ with REA).

5 Conclusion & Future Work

This paper has presented a community-based search algorithm applicable on power-law network which derives its inspiration from natural immune systems. Detailed study of the dynamics of the walk has been done which resulted in an elegant time-varying algorithm. The algorithm outperforms by far any conventional system and may have far reaching impact in designing efficient P2P communities in the future. Further enhancements and a a rigorous testing of the algorithm is the main focus of the work in the future.

References

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

[6] 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

[7] 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