Bio-inspired Search and Distributed Memory Formation on Power-law Networks

Tathagata Das[1], Subrata Nandi[1], Andreas Deutsch[2] and Niloy Ganguly[1]

[1] Department of Computer Science & Engineering
Indian Institute of Technology Kharagpur Kharagpur, India
Email: tathagata.das1565@gmail.com, snandi@cse.iitkgp.ernet.in,
niloy@cse.iitkgp.ernet.in
[2] Center for Information Services and High Performance Computing (ZIH)
TU-Dresden, Dresden 01062
Email: andreas.deutsch@tu-dresden.de

Abstract. In this paper, we report a novel and efficient algorithm for searching P2P networks having a power law topology. Inspired by the natural immune system, 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.

1 Introduction

Due to the dynamic nature of large scale peer-to-peer networks, the search algorithms used for such system need to be decentralized, self-adjusting and robust against rapidly changing system environments. Seeking inspiration from the study of biological processes and organisms is one possibility for coping with these problems. It is well known that living organisms can effectively organize large numbers of unreliable and dynamically-changing components (molecules, cells, individuals, etc.) without having explicit central coordination. Consequently, borrowing ideas from the living nature has long been a fruitful research theme in various fields of communication engineering. The inspiration of design patterns from biological systems has been well exploited in our work [4], where we have introduced practically relevant algorithms for distributed computing that naturally inherit the desirable properties of biological systems including adaptivity and robustness.

In particular, we have taken inspiration from different properties of the humoral and secondary immune system to design and test different random walk and proliferation based search and community formation algorithms. 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 [7]) while other have applied it to Erdos-Renyi networks [8]. However, except some of our previous works [1], [3], 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.

Our previous works [1], [3] was a preliminary work on a grid-based topology which could not be ported in a more realistic power law network. The algorithms presented are completely new algorithms that have been developed based upon a much more thorough understanding of the effect of various dynamics performed on the network. We first apply different varieties of random and greedy search mechanisms in order to understand the dynamics. Finally, we suggest an algorithm which consists of a healthy mix of random and greedy walking for producing search efficiencies exceeding conventional random walk and flooding.

Section 2 discusses in detail the behavior of the immune system and our inspirations from it. Section 3 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 4 and a final algorithm based on these results is proposed in section 5. Finally, we conclude in Section 6 with the possible ways of improving these results.

2 Biological Inspiration: The Immune System

The immune system displays a number of amazing behaviors and attributes that can be an inspiration in providing robust solutions to a number of well known technological problems. The behavior can be distinctly attributed to two different parts of the immune system - the humoral (innate) immune system and the secondary immune system. Each of them has been a source of inspiration as described in the following section.

2.1 Humoral Immune System

In our earlier works [1],[2],[3],[4] we had proposed a search algorithm for peer-to-peer networks that is inspired by the simple and well known mechanism of the humoral immune system where B cells upon stimulation by a foreign agent (antigen) undergo proliferation generating antibodies. Proliferation helps in increasing the number of antibodies while mutation implies a variety of generated antibodies. Consequently the antibodies can efficiently track down the antigens (foreign bodies). This is modeled by considering the query message packet as an antibody which is generated by the node initiating a search whereas antigens are the searched items hosted by other constituent members (nodes) of the P2P network. Similar to the natural immune system, the packets walk through the network followed by proliferation based upon the affinity measure between the message packets and the contents of the node visited.

In our current work, we have analyzed the dynamics of the packet movement in greater detail, in order to determine the parameters that control the degree of movement. The movement of the antibodies can either be a purely random walk or it maybe a biased random walk similar to the *adhesion*-based movements of cells within the *extracellular matrix* (ECM) [6]. This phenomenon of cell movement guided by adhesion is called *Haptotaxis*, and the movement of the query in a P2P network resembles a haptotactic cell movement in the ECM [4],[5]. Based on this phenomenon, we model the two basic types of movement strategies as *random* (unbiased movement) or *greedy* (movement biased by properties of the neighborhood), respectively. The query movement itself can be subdivided into two

distinct phases - *general movement / walk* and *proliferation*, each of which can be independently performed either randomly or greedily. The details of the four algorithms inspired from such random and greedy approaches will be discussed in section 3.

2.2 Secondary Immune System

We have also taken inspiration from the secondary immune response mechanism, which has the capability to develop memory over time and accordingly the antibodies produce a quicker response [1],[3]. This decentralized memory is modeled in our P2P network by restructuring the connections in the network in order to form virtual communities of nodes having similar items (antigens). Each search initiates a rewiring of the network towards community formation based upon the information content of the participating nodes. Due to this community formation, the network gets trained with time to find similar nodes with greater efficiency. In essence, the entire network acts as a large memory which is able to optimize the search process. The exact details of the community formation process are dealt with in section 3.

Based on these inspirations we were motivated to test out the concepts and understand the underlying dynamics in order to decide what level of randomness or greediness is optimal for best performance in P2P networks.

3 Model and Algorithms

In this section we first define the model of the P2P network that we will use in the following, then we discuss the detailed implementation of the search and community formation algorithms.

3.1 Peer-To-Peer Model

We assume a realistic power-law topology for the P2P network (as most of the existing P2P networks exhibit a similar topology). 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.

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 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 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}.

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.

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.

3.2 Algorithms

As indicated in section 2, we would like to test out the four major types of the proliferation-based search algorithms – named $\mathcal{RR}$, $\mathcal{RG}$, $\mathcal{G}R$ and $\mathcal{GG}$. 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.

	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

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. We next explain the latter process, that is, Community Formation.

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

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.

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, 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. 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 undergo unnecessarily 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.

3.3 Evaluation Criteria

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

Search related metrics Let us assume that the i^{th} search produces 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. $\sum_i h_i/n$
Efficiency: Average number of hits per search packet, i.e. $\sum_i \frac{h_i}{p_i}/n$
Similar Node Coverage Average fraction of all the similar nodes present in the network that is returned in each search, i.e. $\sum_i \frac{h_i}{H_i}/n.100$.

Metrics related to community formation The community edges make connections between similar nodes only. If we consider nodes of a particular profile, 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, we desire that all the nodes of a profile are well connected by the community overlay network, we take the LCC of the network as 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.

4 Simulation and Results

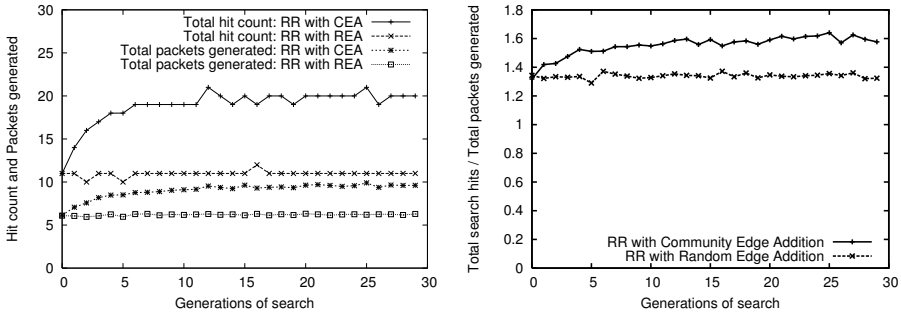
In order to test out the performance of the proposed algorithms, we resorted to simulations whose details are as follows.

4.1 Simulation Scheme

For simulating our algorithm, we took a power-law network of 1000 nodes, generated using the Barabasi-Albert preferential attachment method, 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, we performed a fairness test by comparing the performance of community edge addition (CEA) of $\mathcal{RR}$ with the performance of a $\mathcal{RR}$-type search on a network with an equal number of edges. In this equivalent network, we start from the same initial power law network as the actual simulated network, and we compensate for the increase in the edge count of the latter (due to community edge addition) by randomly adding an equal number of edges (that is, random edge addition (REA))in the equivalent network.

4.2 Results and Analysis

First of all, we present the performance of $\mathcal{RR}$ with community edge addition versus random edge addition on an equivalent graph. Figure 1(a) shows that as generations of search progress, the total number of hits returned by community edge addition increases steeply



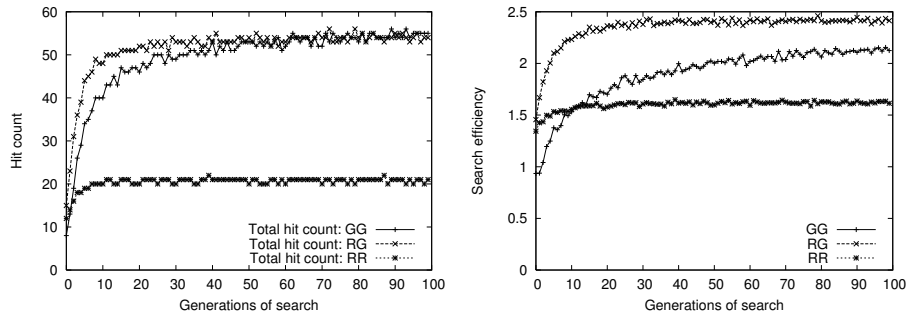
(a) Total number of hits and packets generated

(b) Search efficiency

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

compared to random edge addition, finally producing an average of 20 hits compared to 11 by the latter. In terms of 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 $\mathcal{GG}$ (we omit the result of $\mathcal{GR}$ due to lack of



(a) Hit count

(b) Search Efficiency

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

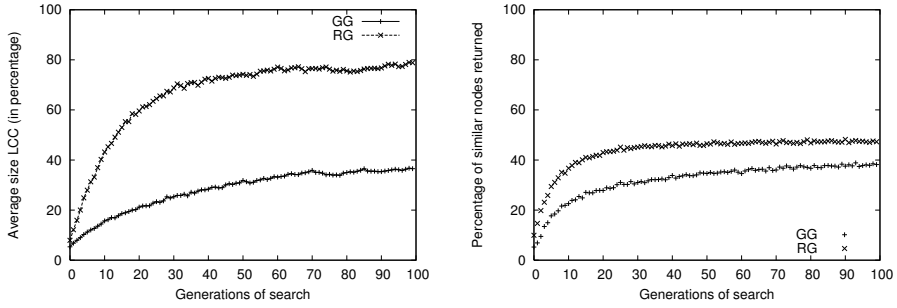
interesting inferences). Figure 2(a) shows that on average, the number of results brought by both types of greedy-proliferation based searches is comparable, while being more than 2.6 times better than that of $\mathcal{RR}$. In terms of search efficiency, $\mathcal{GG}$ and $\mathcal{RG}$ perform about 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 $\mathcal{RG}$? The answer is evident in figure 3(b). On the average, each search by $\mathcal{RG}$ is able to retrieve almost 50% of all the similar nodes that exist in the

network, while $\mathcal{GG}$ is able to retrieve only up to 40%. 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 similar nodes it contains. The larger this fraction, the more well connected they are. Referring to Fig. 3(a) again, we see that the increase in percentage of similar nodes returned correlate well with the average LCC size. This proves the direct correspondence between the goodness of the community structure and the search performance. In fact, on one hand, 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 smaller fraction of similar nodes compared to that present in the average LCC. Overall, it produces a more efficient search in case of $\mathcal{RG}$ than $\mathcal{GG}$.

$\mathcal{GG}$ also takes a long time to reach saturation. This is due to the fact that even though $\mathcal{GG}$ is unable to explore as much in the initial generations, in the later generations, while the node discovery of $\mathcal{RG}$ hits almost zero, $\mathcal{GG}$ continues to find new nodes at a very low rate. Hence, the community structure in $\mathcal{GG}$ continues to grow, slowly improving the search efficiency until it saturates.

To summarize, while a random general walk has a better performance in terms of node



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

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

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.

5 An Approach to Self-Adjusting Search

Extending the idea of antigens and antibodies further, we want to design an algorithm that has the intelligence to adjust itself between two phases - Exploratory Phase and Search Phase. In the former phase, the antibodies would explore the entire network in order to find the location of antigens. In the latter phase, when the antigens have been located, it would like to redirect all its effort towards the affected areas. 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 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, we need to choose a suitable parameter 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

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

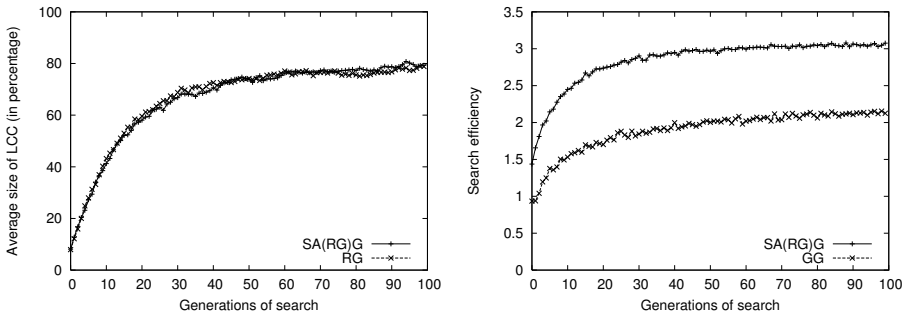
where $X_A = X$ of the node A that is initiating the search. The overall behavior would be as we desire - initially, when X is 0 for all the nodes, it will behave like pure $\mathcal{RG}$, and later as the X of all the nodes reach X_{max}, the probability of random walk reduces to zero, that is, it performs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

5.2 Simulation Results

Figures 4(a) and 4(b) reflect what had been expected. It is able to produce the best possible community structure as fast as $\mathcal{RG}$, and finally produces a search efficiency that is more than 40% better than $\mathcal{GG}$ (and 90% better than the $\mathcal{RR}$).

6 Conclusion & Future Work

This paper has presented a community-based search algorithm which derives its inspiration from natural immune systems. The beauty of the algorithm lies in its simplicity. However,



(a) Average LCC (b) Search Efficiency

Fig. 4. Performance of Self-Adjusting Search

this simple decentralized algorithm generates emergent properties like a complex adaptive system, whose underlying guiding rules are generally also very simple. We find that as a result of the algorithm, the P2P network 'learns' and subsequently develops memory, whereby the search efficiency improves dramatically after some initial learning/training phase. The basic strengths displayed by this algorithm need to be further explored and analyzed by building a mathematical model, and also by applying it in more realistic circumstances in the near 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, 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
3. 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.
4. 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.
5. 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.
6. Dickinson RB, Tranquillo RT. A *Stochastic Model for Adhesion-mediated Cell Random Motility and Haptotaxis*, J. Math. Biol. 1993;31(6):563-600.
7. 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
8. 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