Developing Efficient Search Algorithms for P2P Networks Using Proliferation and Mutation*

Niloy Ganguly, Andreas Deutsch

Center for High Performance Computing, Dresden University of Technology, Dresden, Germany. {niloy, deutsch}@zhr.tu-dresden.de

Abstract. Decentralized peer to peer networks like Gnutella are attractive for certain applications because they require no centralized directories and no precise control over network topology or data placement. The greatest advantage is the robustness provided by them. However, flooding-based query algorithms used by the networks produce enormous amounts of traffic and substantially slow down the system. Recently flooding has been replaced by more efficient k-random walkers and different variants of such algorithms [5]. In this paper, we report immune-inspired algorithms for searching peer to peer networks. The algorithms use the immune-inspired mechanism of affinity-governed proliferation and mutation to spread query message packets in the network. Through a series of experiments, on different types of topologies, we compare proliferation/mutation with different variants of random walk algorithms. The detailed experimental results show message packets undergoing proliferation and mutation spread much faster in the network and consequently proliferation/mutation algorithms produce better search output in $p2p$ networks than random walk algorithms.

1 Introduction

Among different desirable qualities of a search algorithm for $p2p$ networks, robustness is a very important aspect. That is, the performance of a search algorithm should not radically deteriorate in face of the dynamically changing condition of the network. As is known, the big share of Internet users, consequently participants in $p2p$ networks, still use dial-up modems, who besides being slow and unreliable, also leave the community at very short intervals. Thus in order to give robustness a high priority, algorithms generally avoid precise routing algorithms for forwarding query message packets. Instead random forwarding of the message packets forms the basis of their algorithms [5]. The goal of this paper is to study more efficient alternatives to the existing k-random walk. In this connection, we draw our inspiration from the immune system.

Our algorithm has been inspired by the simple and well known mechanism of the humoral immune system where B cells upon stimulation by a foreign agent

* This work was partially supported by the Future & Emerging Technologies unit of the European Commission through Project BISON (IST-2001-38923).

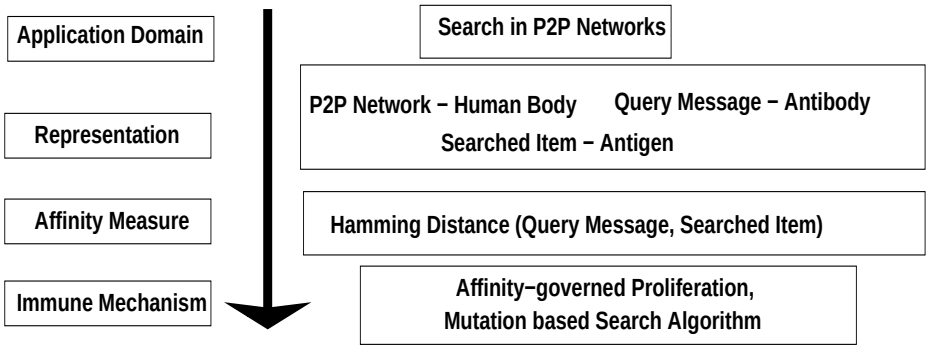


Fig. 1. Immune system concepts used to develop search algorithms

(antigen) undergo proliferation and mutation generating antibodies. Proliferation helps in increasing the number of antibodies while mutation implies a variety of generated antibodies. The antibodies consequently can efficiently track down the antigens (foreign bodies). *Fig. 1* provides an illustration explaining how we have mapped immune system concepts to our search problem. In our problem, the query message packet is conceived as 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 networks. Like in the natural immune system, the packets undergo mutation and proliferation based upon the affinity measure between the message packets and the contents of the node visited which results in an efficient search mechanism.

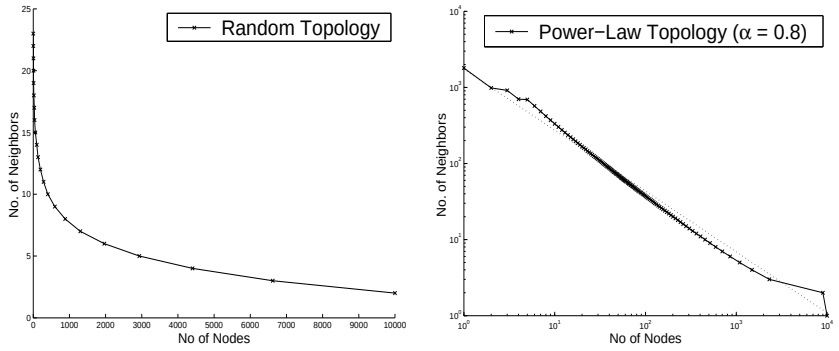
In the next section, we detail the modeling abstractions upon which the algorithms are based. Moreover, we elaborate our algorithms as well as different variants of k-random walk algorithms. The evaluation metrics used to compare the different schemes are also elaborated. The experimental results are noted next in *Section 3*.

2 Modeling and Evaluation Methodology

It is impossible to model the complete dynamics of a $p2p$ system. While our simple models do not capture all aspects of reality, we hope they capture the essential features needed to understand the fundamental qualitative differences between k-random walk and proliferation/mutation algorithms.

2.1 Model Definition

$P2p$ networks are the networks formed through associations of computers, each providing equivalent services, eg. search facility, to the network. Thus each peer can be conceived as both client and server of a particular service [5]. To model search service, we focus on two most important aspects of a $p2p$ system: $p2p$



a. Random graph, 10000 nodes, with $\mu = 4$

b. Power-law graph, 10000 nodes, with $\mu \approx 4$ and $\alpha \approx 0.8$

Fig. 2. Distribution of node degrees in the two network topology graphs. Note that we use log scale for the power-law graph, and linear scale for the random graph.

network topology, query and data distribution. For simplicity, we assume the topology and distribution do not change during the simulation of our algorithms. For the purpose of our study, if one assumes that the time to complete a search is short compared to the time of change in network topology and change in query distribution, results obtained from the fixed settings are indicative of performance in real systems.

Network Topology : By network topology, we mean the graph formed by the *p2p* overlay network; each *p2p* member has a certain number of neighbors and the set of neighbor connections form the *p2p* overlay network. We use two different network topologies in our study. The two types of graph - power-law and random graph - best represent the majority of the realistic network topologies formed in the Internet [3, 5]. In each of the topologies, we take a representative graph.

(a). Pure Random Graph : A 10000-node graph generated with the help of the topology generator BRITE[4] with mean $\mu = 4$. The distribution is shown in (*Fig. 2(a)*).

(b). Power-law Graph : The node degrees follow a power-law distribution; if one ranks all nodes from the most connected to the least connected, then the i^{th} most connected node has ω/i^{α} neighbors, where ω is a constant and α is the power-law exponent. The power-law random graph is generated by the topology generator Inet3.0 [2]. The number of nodes in the graph is 10000, the mean in-degree $\mu = 4.3$ and the power-law exponent $\alpha \approx 0.8$ (*Fig. 2(b)*).

Query and data distribution : In order to model query and data distribution, we define two different profiles for each peer - the *informational profile* and the *search profile*. The *informational profile* (P_I) of the peer is formed from the information which it shares with the other peers in the *p2p* network. The *search profile* (P_S) of a peer is built from the informational interest of the user; formally it is represented in the same way as is P_I. In general, the search profile may differ from the information stored on the peer. For simplicity we assume that there

are 1024 coarse-grained profiles, and let each of these profiles be represented by a unique 10-bit binary token. The query message packet (M) is also a 10-bit binary token. From now on we interchangeably use the term profile and token. Zipf's distribution[6], is chosen to distribute each of the 1024 unique alternatives in the network. The ranking of tokens in terms of frequency is the same for both information and search profiles.However the profiles are distributed in random nodes with no correlation between similar profiles.

We now describe the proliferation/mutation and random walk algorithms.

2.2 Algorithms

In this section, we explain two proliferation/mutation based as well as three random walk based search algorithms. The important aspects of all these algorithms are that although random walk or proliferation/mutation is exhibited by the message packets, however, the algorithms are independently implemented by each node. And coordinated behavior of the nodes produces the required packet dynamics. All the algorithms can be expressed in terms of the same basic premise which is stated next.

Basic Premise : The search in our $p2p$ network is initiated from the user peer. The user (U) emanates k ($k \geq 1$) message packets (M) to its neighbors - the packets are thereby forwarded to the surroundings. The message packet (M) is formed from the search profile P_S of U. We next present the search initiation process in algorithmic form.

Algorithm 1 InitiateSearch(U)
Input : *Signal to initiate search.*
Form Message Packet (M) = $P_S(U)$
Flood k message packets(M) *to the neighbors of the user peer.*

The message packets travel through the network and when a node (say) A receives a message packet (M), it performs the following two functions.
Function 1 :- It checks whether the P_I of A is equal to the incoming message M. If so, it returns a successful event.
Function 2 :- It forwards the content of the message packet in some *defined* manner to its neighbor(s).

In algorithmic form, we can represent the functions as *Reaction_p2p*:

Algorithm 2 Reaction_p2p(A)
Input : Message packet(M)
If ($P_I = M$) *then* {*Report a successful match /*Function 1*/*}
Algorithm Message_Forward(A) / Function 2*/*

Each of the proliferation/mutation and random walk schemes defines *Algorithm Message_Forward(A)* differently. Elaboration of the algorithms corresponding to each of the schemes follows.

Proliferation/Mutation (PM) : In the proliferation/mutation scheme, the packets undergo proliferation at each node they visit. The proliferation is guided by a special function, whereby a message packet visiting a node proliferates to

form N_{new} message packets which are thereby forwarded to the neighbors of the node. A randomly selected bit of each of these N_{new} messages has a probability β of getting mutated; β is the mutation probability of the system. Mutation is introduced into the system to increase the chance of message packets meeting similar items, which in turn helps in packet proliferation.

However, since mutation changes the content of the packet, it is assumed that the original information is also carried along with the packet. Hence, during the execution of the algorithm *Reaction_p2p*, comparison with P_I (Function 1) is carried out on the basis of the original message, while the input for *Algorithm Message_Forward(A)* (here Algorithm $PM(A)$) is the mutated packet.

Algorithm 3 PM(A)
Input : Message packet(M)
Produce N_{new} message packets(M)
Mutate one randomly selected bit of each of the N_{new} message packets with prob. β
Spread the N_{new} packets to N_{new} randomly selected neighbors of A

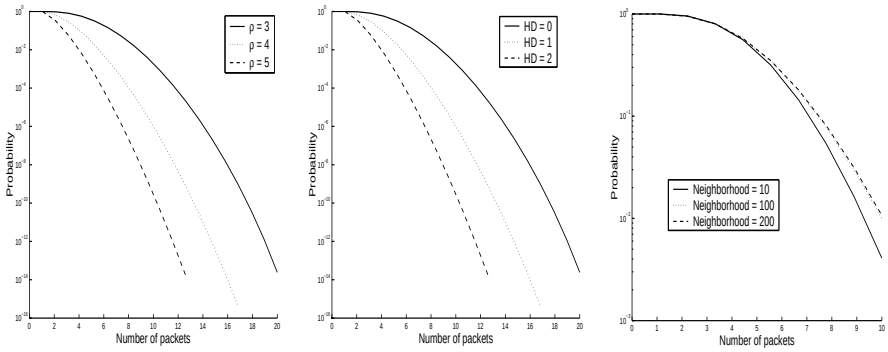
The function determining the value of 'N_{new}' ensures that N_{new} is $< n(A)$, where $n(A)$ is the number of neighbors of A and ≥ 1. [Note that if $N_{new} = 1$, proliferation/mutation behaves similar to random walk.]

Restricted Proliferation/Mutation (RPM) : The restricted proliferation/mutation algorithm, similar to PM, produces N_{new} messages and mutates one bit of each of them with probability β. But these N_{new} messages are forwarded only if the node A has $\geq N_{new}$ free neighbors. By 'free', we mean that the respective neighbors haven't been previously visited by message M. If A has $\mathcal{Z}$ 'free' neighbors, where $\mathcal{Z} < N_{new}$, then only $\mathcal{Z}$ messages are forwarded, while the rest are destroyed. However, if $\mathcal{Z} = 0$, then one message is forwarded to a randomly selected neighbor. The rationale behind the restricted movement is to minimize the amount of message wastage. Because, two packets of message M visiting the same peer essentially means wastage of the second packet.

Algorithm 4 RPM(A)
Input : Message packet(M)
Produce N_{new} message packets (M)
Mutate one bit of each of the message packets with probability β
$\mathcal{Z}$ = No of 'free' neighbors
if ($\mathcal{Z} \geq N_{new}$)
    Spread the N_{new} packets in N_{new} randomly selected neighbors of A
else
    if ($\mathcal{Z} > 0$)
        Spread $\mathcal{Z}$ packets in $\mathcal{Z}$ free neighbors of A
        Discard the remaining (N_{new} - $\mathcal{Z}$) packets
    else
        Forward one message packet to a randomly selected neighbor of A
        Discard the remaining (N_{new} - 1) packets

We now elaborate the function which controls the amount of proliferation.

Proliferation Controlling Function : The proliferation of message packets at any node A is heavily dependent on the similarity between the message



a. Case 1: ρ varies; $HD = 0$, $n = 20$

b. Case 2: HD varies; $\rho = 3$, $n = 20$

c. Case 3: n varies; $\rho = 3$, $HD = 0$

Fig. 3. Probability of proliferation of at least η messages (semilog(y) scale)

packet (M) and the information profile (P_I) of A. Also care is taken to avoid producing more than $n(A)$ number of messages, where $n(A)$ represents the number of neighbors of A. In this connection we define the following expression p, where $p = e^{-HD} \times \frac{\rho}{n}$, HD is the Hamming distance(M,P_I); ρ represents the proliferation constant; it is same for all nodes. and is generally kept less than the mean indegree of the underlying network. However, since in networks (both random and power-law network), the neighborhood distribution varies widely; if for some particular node (say B) $\rho > n(B)$, then for that node ρ is set to $n(B)$.

With the help of the expression p, we define $P(\eta)$ - the probability of producing at least η packages during proliferation by the following equation.
$P(\eta) = \sum_{i=\eta}^{n} \binom{n-1}{i-1} \cdot p^{i-1} \cdot (1-p)^{n-i}$

The significance of the above equation is elaborated through the three figures (*Fig. 3*). The three figures explain the behavior of $P(\eta)$, w.r.t. different HDs, *proliferation constants* and neighborhood sizes. The number of packets is plotted in the x-axis, while the y-axis represents the probability of proliferation of at least those number of packets. All the figures illustrate some commonality. (i). At least one packet necessarily proliferates; (ii). The probability of proliferation of larger number of packets exponentially decreases. *Fig. 3(a)* shows the variation w.r.t. different proliferation constants ρ; three different curves are drawn for $\rho = 3, 4, 5$. In each of the curves, we find that the probability of proliferating at least η numbers of packets is almost equal to 1 till $\eta = \rho$; if $\eta > \rho$, the probability decreases exponentially. *Fig. 3(b)* shows the variation w.r.t. different Hamming distances; three curves are drawn corresponding to $HD = 0, 1, 2$. It is seen that for $HD > 0$, the probability of proliferation of at least η packets decreases exponentially for $\eta > 1$, which implies that the chance of proliferation is quite low if $HD > 0$. However, the rate of decrease in probability varies inversely to the value of the *proliferation constant*. The three curves in *Fig. 3(c)*, plotted w.r.t. different neighborhood sizes show that variation of probability of proliferation with respect to different neighborhood sizes is negligible.

We now describe a simple k-random walk algorithm and subsequently two different variations of it.

k-random walk (RW) : In k-random walk, when a peer receives a message packet after performing the task of comparison, as mentioned in *Algorithm 2*, it forwards the packet to a randomly selected neighbor. The algorithm (RW) is quite straightforward and is defined as

Algorithm 5 RW(A)
Input : Message packet(M)
Send the packet M to a randomly chosen neighbor peer

The restricted random walk (RRW) algorithm which is similar to RPM (*Algorithm 4*), is discussed next.

Restricted Random Walk (RRW) : In RRW, instead of passing the message (M) to any random neighbor, we pass on the message to any randomly selected 'free' neighbor. However, if there is no 'free' neighbor, we then pass on the message to any randomly selected neighbor.

Algorithm 6 RRW(A)
Input : Message packet(M)
Send the packet M to a randomly chosen 'free' neighbor peer
If (no 'free' neighbor)
    Send the packet M to a randomly chosen neighbor peer

The next algorithm is a special type of a random algorithm [1] to enhance the speed of simple random walk in a power-law network. The special type of random walk is termed as *high degree restricted random walk (HDRRW).*

High Degree Restricted Random Walk (HDRRW) : Adamic et. al. in [1] showed that for a single random walker in a power-law network, *high degree random walk* is better than simple random walk. To compare it with our proliferation/mutation scheme, we have simulated the algorithm with k-random walkers participating in the search. In the algorithm, a peer has special affinity to forward the message packet to the neighbors which have higher degree of connection. However, the algorithm also takes into consideration the restricted movement discussed in the previous section.

To balance these two trends, when sending a message packet, a peer checks the first $\mathcal{H}$ most high degree nodes to identify a 'free' node; if it doesn't find, it randomly tries to identify any 'free' node another $\mathcal{L}$ number of times. Even if then a 'free' node is not found, then the message is forwarded to a neighboring node. This forwarding scheme is also biased towards neighbors with high in-degree.

Algorithm 7 HDRRW(A)
Input : Message packet(M)
Find the 'free' neighbor with highest in-degree among the $\mathcal{H}$ most connected neighbors
If (no 'free' neighbor)
    Randomly try $\mathcal{L}$ times to find a 'free' neighbor among the rest of the peers
If (still no 'free' neighbor found)
    Chose a neighbor through a probability function $f(neigh)$;
        where $f(neigh_1) > f(neigh_2)$ if $n(neigh_1) > n(neigh_2)$
Send the packet M to the chosen peer

2.3 Metrics

In this paper we focus on efficiency aspects of the algorithms solely, and use the following simple metrics in our abstract $p2p$ networks. These metrics, though simple, reflect the fundamental properties of the algorithms.
(a). *Success rate:* The number of similar items found by the query messages within a given time period.
(b). *Coverage rate:* The amount of time required by the messages to cover a percentage of the network.
(c). *Cost per search output:* The number of messages required to output a successful search.
(d). *Bad visits:* The number of times the same message re-visits the same node. If a message packet visits the same node more than once, it amounts to wastage of that packet.

3 Simulation Results

The experiments are performed on the two types of topology discussed in *Section 2.1.* As mentioned earlier, each of the above algorithms is distributed in nature and the nodes perform the task independent of the others. However, to assess the speed and efficiency of the algorithm, we have to ensure some sort of synchronous operation among the peers. In this context we introduce the concept of time whereby it is assumed that in one time unit, all the nodes in the network execute the algorithm once. That is, if a peer has some message in its message queue, it will process one message within that time frame. We believe although approximate, it is a fair abstraction of reality of $p2p$ networks where each node is supposed to provide equivalent services. The sequence of operation of the peers during one time step is arbitrary. The length of the message queue is considered to be infinite.

In order to assess the efficiency of different algorithms, we have also to ensure fairness of 'power' among them which is explained next.

3.1 Fairness in power

To ensure fair comparison among all the processes, we must ensure that each process (PM, RPM, RW, RRW, $HDRRW$) should participate in the network with the same 'power'. To provide fairness in 'power' between a proliferation/mutation algorithm (say PM) and a random algorithm (say RW), we ensure that the total number of query packets used is roughly the same in all the cases. Query packets determine the cost of the search; too many packets cause network clogging bringing down the efficiency of the system as a whole. It can be seen that the number of packets increase in the proliferation/mutation algorithms over the generations, while it remains constant in the case of random walk algorithms. Therefore the number of message packets - k in *Algorithm 1* is set in a fashion so that the aggregate number of packets used by each individual algorithm is roughly the same.

To ensure fairness in 'power' between two proliferation/mutation algorithms (say [PM & RPM]), we keep the proliferation constant ρ and the value of k the same for both processes. The value of k for the proliferation/mutation algorithm is generally set as $k = n(U)$, where $n(U)$ is the in-degree of the initiator peer U.

3.2 Experiments

To explore the different properties of the algorithms, two major types of experiments have been performed on different topologies and with different initial conditions. The experiments are noted one by one.

COVERAGE : In this experiment, upon initiation of a search, the search operation is performed till the message packets cover the entire network. The experiment is repeated 500 times on randomly selected initial nodes.

During the experiment, we collect different statistic at every 10% of coverage of the network that is, we collect statistic at [20%, 30% $\cdots$ 90%, 100%] of coverage of the network. Since the message forwarding algorithms (*Algo. 3 -7*) are non-deterministic in nature, message packets find it increasingly difficult to visit the last 10% of the network. This is true for all the different variants of message forwarding algorithms and also for all types of topologies. Consequently, the results of *Figs. 4, 5, 7* reflect this characteristic.

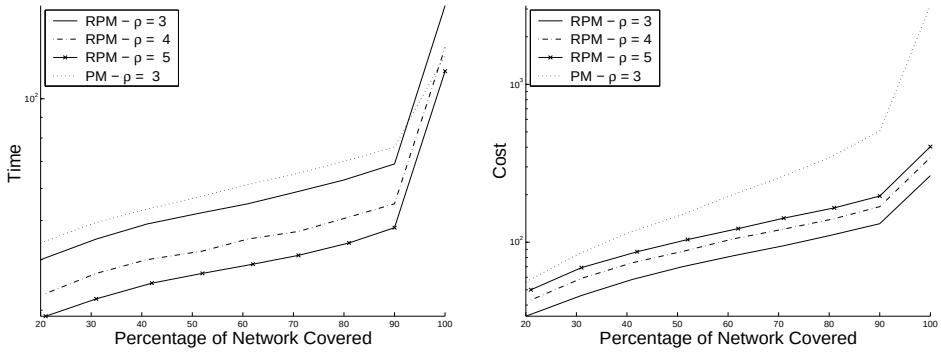
TIME-STEP : In this experiment, upon initiation of a search (*Algorithm 1*), the search operation is performed for $\mathcal{N}$ (= 50) time steps. The number of search items (n_s) found within 50 time steps from the commencement of the search is calculated. The experiment is repeated for one generation where one generation is defined as a sequence of 100 such searches. The search output (n_s) is averaged over one generation (100 different searches), whereby we obtain N_s, where $N_s = \frac{\sum_{i=1}^{100} n_s}{100}$. The value of N_s, provides the indication of search efficiency.

The above mentioned two experiments have been performed for proliferation/mutation and random walk processes (*Algorithm 3 - 7*). The interesting results derived from such experiments are noted next.

3.3 Experimental Results - Random Network

In this section we report the results obtained by performing the **COVERAGE** experiment on the random network of *Fig. 2(a)*. The results reported below pertains to experiments performed for PM, RPM with different proliferation constants (ρ = 3, 4, 5), RRW. In case of the random network, our observation is that mutation does not help in improving the efficiency of *network coverage*. Hence all the proliferation mutation experiments reported here are performed with $\beta = 0$. The major experimental observations are elaborated one by one.

Result I: Comparison between PM and RPM algorithm : *Fig. 4(a)* shows the network coverage rate of the PM algorithm at $\rho = 3$, and three different instances of the RPM algorithm at $\rho = 3$, 4, 5 respectively. The graph plots the % of network covered in the x-axis, while the time taken to cover corresponding % of network is plotted on the y-axis (semilog scale). It is seen



a. Time taken to cover the network b. Cost incurred to cover network

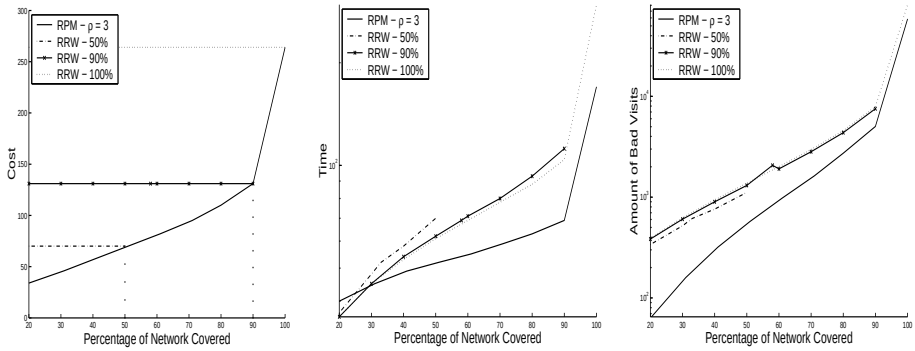
Fig. 4. Graphs plotting (in semilog(y) scale) the cost and network coverage time of PM [$\rho = 3$] and RPM [$\rho = 3, 4, 5$] algorithms in random networks.

that PM ($\rho = 3$) takes more time to cover up to 90% of the network than RPM with $\rho = 3$. Only while covering the last 10% it overtakes RPM. However, if we increase ρ for RPM, we see that even at 100%, RPM ($\rho = 5$) performs better than PM. An interesting observation to be noted is that RPM ($\rho = 5$) produces a smaller number of packets than PM ($\rho = 3$).

Fig. 4(b) plots the increase in the average number of message packets present in the network (also referred to as cost) in the y-axis with respect to the percentage of network coverage for the four schemes. It is seen that the number of message packets produced in $PM(\rho = 3)$ is about 10 times larger than $RPM(\rho = 3, 4, 5)$. Similarly, in case of random walks, it is found that RRW is much more efficient than RW (the experimental results are not reported here to avoid repetition). So, in our subsequent discussions, we drop PM and RW and concentrate on comparison between RPM and RRW.

Result II: Comparison Between RPM and RRW algorithm : The comparison between RPM and RRW is elaborated through the results highlighted in *Fig. 5*. In case of RRW, we perform three different sets of experiments by varying the initial condition (value of k in *Algorithm 1*). The three different experiments are termed as RRW(50%), RRW(90%), RRW(100%) respectively. The value of k for these experiments are set from collecting information about the average number of packets used by $RPM(\rho = 3)$ to complete coverage of 50%, 90% and 100% of the network respectively. *Fig. 5(a)* plots the average number of packets used by RPM, RRWs (y-axis) vs. network coverage (x-axis).

Fig. 5(b) plots the percentage of the network covered (x-axis) by the four processes RPM($\rho = 3$), RRW(50%), RRW(90%), RRW(100%) against time step (y-axis, in semilog scale). It is seen that the time taken by $RPM(\rho = 3)$ to cover the network is uniformly less than all the other processes beyond the 30% coverage. This is particularly significant because when we are at 30%-40% network coverage zone, the number of message packets used by RPM is significantly lower than RRW algorithms (Refs. Fig. 5(a)). As expected, the time



a. Average number of message packets used.

b. Network coverage time (semilog(y) scale).

c. Number of bad visits (semilog(y) scale).

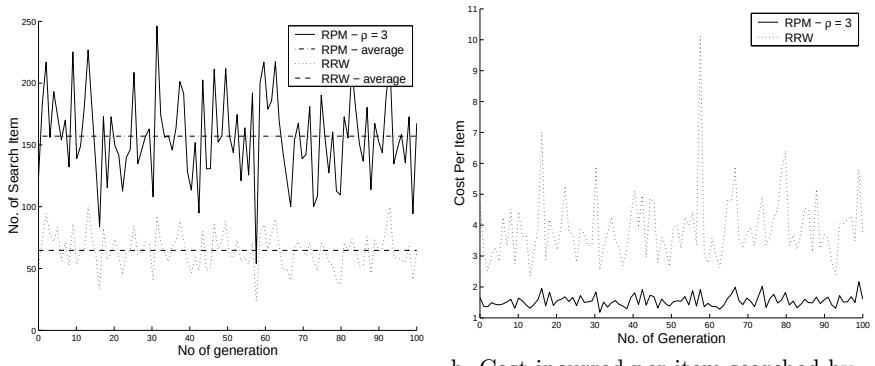
Fig. 5. Graphs highlighting the (a) average number of message packets used (cost), (b) network coverage time, and (c) number of bad visits by RPM and RRWs ($RRW(100\%)$, $RRW(90\%)$ & $RRW(50\%)$) in random network.

taken to cover the network decreases progressively for RRW(100%), RRW(90%) and RRW(50%). This is because the three processes have an increasing number of message packets. However, as seen the time does not decrease significantly even if we go on adding more messages. The RPM functions are better than RRWs because the packets here exhibit a much smaller probability to visit the same nodes again and again. *Fig. 5(c)* shows the number of *bad visits* (defined on page 8) performed by RPM and RRWs (y-axis) vs. network coverage (x-axis). It is seen that the tendency of RPM to visit the same node again and again is significantly lower.

3.4 Experimental Results - Search Efficiency

To compare the search efficiency of RPM & RRW, we perform the *TIME-STEP* experiment on the random graph for RPM and RRW, each spanning over 100 generations. The graph of *Fig. 6(a)* shows the average value N_s against generation number for RPM and RRW. The x-axis of the graph shows the generation number while the y-axis represents the average number of search items (N_s) found in the last 100 searches. In this figure we see that the search results for both RPM and RRW show fluctuations. The fluctuations occur due to the difference in the availability of the searched items selected at each generation. However, we see that on the average, search efficiency of RPM is almost 2.5-times higher than that of RRW. (For RPM, the number of hits ≈ 157, while it is ≈ 64 for RRW.) The fluctuations in the results help us to understand an important aspect about cost which is discussed next.

Fig. 6(b) displays the cost/search item (the number of messages required to produce a search output) each scheme incurs to generate the performance of *Fig. 6(a)*. We see that the cost of RPM is hardly changing (it stays constant



a. Search efficiency of RPM and RRW

b. Cost incurred per item searched by RPM and RRW

Fig. 6. Graphs showing (a). search efficiency, and (b). cost incurred per item searched by RPM and RRW in random network.

at around 1.5) even though the corresponding search output is differing hugely, while in RRW there is significant fluctuation in terms of cost. This can be easily understood from the fact that RRW always starts with the same number of packets irrespective of the availability of the items. While in RPM, the packets are not generated blindly, but are instead regulated by the availability of the searched item. Therefore, if a particular searched item is sparse in the network, RPM produces a lower number of packets and vice versa.

3.5 Experimental Results - Power-Law Graph

In this section, we report the results obtained by performing the COVERAGE experiment on the power-law graph of *Fig. 2(b)*.

Result I : Comparison between power-law network and random network : *Fig. 7* plots % of network coverage in the x-axis, while the time taken to cover the network space in y-axis (semilog scale). The results for RPM and RRW are plotted for both random and power-law network, while $HDRRW$ is plotted only for power-law network. It is seen that to cover the power-law network is almost 10 times more difficult than random network. This happens because in power-law network a few nodes have a huge number of connections while most of the nodes have very few connections (Refs. *Fig. 2(b)*). Hence, to reach a particular node (say x) from another node (say y) (both sparsely connected), the message has to pass through one of the more connected nodes, consequently creating an overload of message on those more connected node. Similar to the results in random networks, the RPM works much better than RRW in power-law networks too. The $HDRRW$ works better than simple RRW however only slightly. The efficiency of $HDRRW$ for multiple random walkers is not as good as in the case of single random walker (as illustrated by Adamic et. al. in [1]).

The next set of results shows that in power-law networks, mutating the proliferated packets indeed helps in improving the coverage rate.

Result II : Effect of mutation on network coverage rate : *Fig. 8(a)* plots the time taken to cover the network (y -axis) by RPM at mutation probabilities ($\beta = 0,\ 0.1,\ 0.5$) for $\rho = 3$, and at $\rho = 3.5$, $\beta = 0$ vs. % of network coverage (x-axis). It is seen that the performance is best for RPM with ($\beta = 0.1$, $\rho = 3$). It performs better than even RPM with $\rho = 3.5$. *Fig. 8(b)* plots the number of message packets (y -axis) produced by the corresponding schemes vs. % of network coverage (x-axis). It is seen that among the RPMs with $\rho = 3$, RPM with $\beta = 0.1$ produces the largest number of message packets. However, it is less than the number of packets produced by RPM ($\rho = 3.5$). This shows that mutation plays a distinct role in increasing the network coverage efficiency. Merely, by increasing the value of the proliferation constant ρ, we are not able to produce the combined effect of proliferation and mutation.

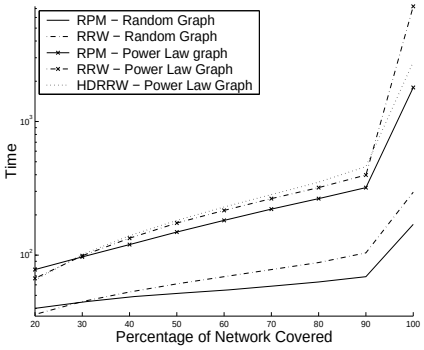
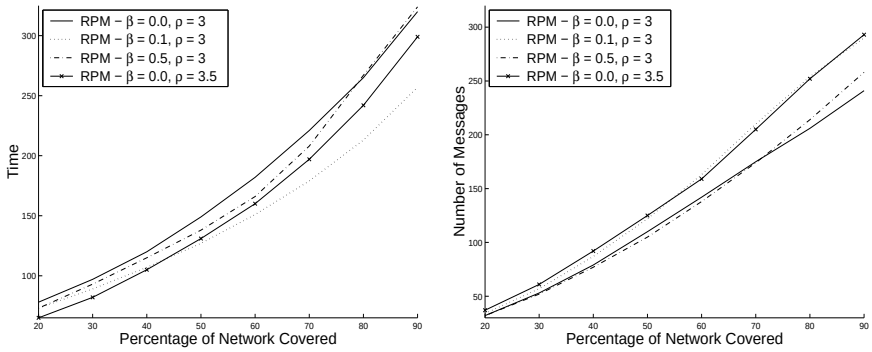


Fig. 7. Graphs showing network coverage time (semilog(y) scale) by RPM, RRW and $HDRRW$ in random and power-law network.

In power-law networks, a few nodes have a huge number of connections which implies that messages most of the time have to pass through those nodes to reach other nodes. Hence, in order to get a message directed through those nodes more effectively, a high amount of proliferation in their neighboring nodes is desirable. Since the amount of proliferation directly depends on the level of similarity between the message and the information profile (P_I) of the node, a wider variety in the messages improves the chance of similarity between message and information profile. However, these neighboring nodes are only a subset of the total nodes and the frequency distribution of their information profiles are guided by Zipf's law. Excessive mutation tends to make the message packets distribution more uniform and thus the inherent frequency inequality in the distribution of different information profiles cannot be exploited.

3.6 Summarization

The following is the summarization of the results.
(a). RPM is more effective than PM.
(b). RPM is more effective than any random walk algorithm.
(c). RPM has an in-built excellent cost regulatory mechanism.
(d). The search efficiency of RPM is roughly three times higher than RRW.
(e). Coverage is more difficult in power-law networks than random networks.
(f). RPM with mutation probability (β) > 0 functions more effectively in power-law networks, however regulation of mutation probability is important.



Graphs showing network coverage time Graphs showing incurred cost

Fig. 8. Graphs showing network coverage time and cost incurred in power-law network by RPMs with ($\rho = 3$, $\beta = 0.0$), ($\rho = 3$, $\beta = 0.1$),($\rho = 3$, $\beta = 0.5$), ($\rho = 3.5$, $\beta = 0.0$).

4 Conclusion

In this paper, we have produced detailed experimental results showing that the simple immune-inspired concept of proliferation/mutation can be used to cover the network more effectively than random walk. The proliferation/mutation algorithm can regulate the number of packets to be produced during a search operation according to the availability of the searched material, thus improving the efficiency of search. The effectivity of search is demonstrated across the two major types of topologies we have taken into account in this paper. This we believe is a fundamental result and can be applied beyond the domain of the proposed $p2p$ search application. However, a detailed theoretical analysis to explain these interesting results has to be undertaken in the future to explore the full potential of proliferation/mutation algorithms.

References

1. L A Adamic, R M. Lukose, A R. Puniyani, and B A Huberman. Search in Power-Law Networks. *Physical Review E*, 64:046–135, 2001.
2. C Jin, Q Chen, and S Jamin. Inet: Internet Topology Generator. University of Michigan Technical Report CSE-TR-433-00, 2002.
3. M. A. Jovanovic, F. S. Annexstein, and K. A. Berman. Scalability Issues in Large Peer-to-peer Networks - A Case Study of Gnutella. Technical Report University of Cincinnati, 2001.
4. A Medina, A Lakhina, I Matta, and J Byers. BRITE: An Approach to Universal Topology Generation. In *MASCOTS*, August 2001.
5. A. Oram, editor. *Peer-to-Peer: Harnessing the Power of Disruptive Technologies*. O Reilly Books, 2001.
6. G. K. Zipf. *Psycho-Biology of Languages*. Houghton-Mifflin, 1935.