A Cellular Automata Model for Immune Based Search Algorithm*

Niloy Ganguly Andreas Deutsch

Contact Author : niloy@zhr.tu-dresden.de

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 amount of traffic and substantially slow down the system. Recently flooding is increasingly replaced with more efficient k-random walkers and different variants of such algorithms [6]. In this paper, we develop an efficient search algorithm for $p2p$ networks with the help of a 2-dimensional Cellular Automata model. The rules followed by each individual cell of the CA are inspired by concepts of natural immune systems whereby the query message packets in the network are spread through proliferation. Through a series of experiments, we compare proliferation with different variants of random walk algorithms. The detailed experimental results show message packets undergoing proliferation spread much faster in the network and consequently produce better search output in $p2p$ networks. Moreover, experimental results show that proliferation rules are extremely scalable and their performance is largely insensitive to the change in dimension of the CA grid.

1 Introduction

Significant research efforts have recently been undertaken in the development of peer-to-peer ($p2p$) computing systems. Due to their flexibility, reliability and adaptivity, $p2p$ solutions can overcome a lot of disadvantages of client-server systems, and can fit to the dynamically changing Internet environment [7]. This explains the high popularity of file sharing systems like Gnutella [2], Napster [1], and Freenet [3].

Among different desirable qualities of a search algorithm, 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 ($p2p$ network) 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 is pursued.

The goal of this paper is to study more-efficient alternatives to the existing k-random walkers[1] algorithms, which form the base algorithm for a large class of $p2p$ networks. In this connection, we draw our inspiration from the natural immune system. *Our algorithm has been inspired by the simple and well known concept of the humoral immune system where B cells undergo proliferation generating antibodies. The increasing number of antibodies consequently efficiently track down the antigens (foreign bodies).* Based upon the above concept

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

[1] A system where there are k message packets each performing a random walk independently

of proliferation, there has been some initial work on searching landmines [5]. However, to the best of our knowledge, immune system concepts haven't been used to develop complex search algorithms in *p2p* networks. Also, to the best of our knowledge, there isn't any published literature where Cellular Automata have been used to develop a search algorithm for *p2p* networks.

In this paper, we develop a 2-dimensional Cellular Automata model to simulate the *p2p* environment. Performance by different random walk and proliferation techniques are evaluated with the help of the 2-dimensional CA. Hence in the next section, we detail the CA model used to abstract the *p2p* environment. The experimental results are noted in *Section 3*.

2 Modeling and Evaluation Methodology

In this section, we elaborate the two-dimensional CA model. Both state definition of each cell of the CA as well as the rules guiding the state transition behavior of the cells are defined. The transition rules can be divided into two parts, rules which are common to both proliferation and random walk techniques and rules which are unique to each of these techniques. In *Section 2.1*, we describe the features common to all the processes, whereas in *Section 2.2*, we explain the unique rules. The evaluation metrics based upon which comparisons among the techniques are carried out is defined in *Section 2.3*.

2.1 CA Model

It is impossible to model all the dynamics of a *p2p* system. In this paper, we are not trying to resolve small quantitative disparities between k-random walk and proliferation algorithms, but instead are trying to reveal fundamental qualitative differences. While our simple models do not capture all aspects of reality, we hope they capture the essential features needed to understand the qualitative differences.

Each cell of the two-dimensional $(m \times n)$ CA represents a peer/node in a *p2p* system. The CA maintains the Von-Neumann neighborhood configuration, which implies each peer in the network (consequently cells[2] in the CA) can communicate with four neighbors. The peers (cells) host some data which are searched by other peers (cells) during a search operation. For simplification, we have assumed that there are 1024 different types of data present in the network, each of which can be represented by a 10-bit string. Hence, each cell hosts a 10-bit data string (say) P which we also refer to as token.

The search operation is performed with the help of three rules - *Query Initiation Rule (QIR)*, *Query Processing Rule (QPR)* and *Query Forwarding Rule (QFR)*. Each of the rules are explained one by one.

Query Initiation Rule : A query is initiated by a randomly selected peer which requests for some data (say M). To obtain an answer to the request, the peer floods message packets (M) in its neighboring peers. In order to receive these message packets, each cell of the CA maintains a message queue MQ. The length of the message queue is considered as infinite. Like the tokens, there are 1024 query messages (M) each of width 10-bit. During experimentation, we perform search operation several times. In this connection, we form a query set Q, where these 1024 different messages are distributed in varying frequency.

The Query Initiation Rule is elaborated below

Rule 1 QIR : *Form Message Packet M /*Pick an element M from Query Set Q*/*
Flood k message packets(M) in the neighbors.

[2] Throughout the rest of the paper, we use the terms cell, peer and node synonymously

The distribution of the 1024 unique tokens in the network (data distribution) as well the messages in query set Q (query distribution) follow Zipf's law, which is discussed next.
Data and Query Distribution : Zipf's distribution[8], is chosen to distribute each of the 1024 unique alternatives. In Zipf's distribution the frequency of occurrence of some event (here token/message) t, as a function of the rank i, where the rank is determined by the above frequency of occurrence, follows a power-law $t_i \; \alpha \; \frac{1}{i^a}$. In the experimental setup, the Zipf's exponent (value of a) for both the data and query distribution is 1. The ranking in terms of frequency is the same for both tokens and messages—for instance, the most popular token in the network is also the most popular message in the query set.

We now describe the rule used to process query messages.
Query Processing Rule : Once the search is initiated, the messages hop from cell to cell in subsequent time steps. Whenever, a cell encounters a message packet, it checks whether the message has earlier visited it (the cell) or not. In this connection, each cell maintains a field named V. If the message has not previously visited the cell, then the message is compared with P. A successful search is reported if $P = M$. A field SS which initially is set to 0 is updated to 1 to indicate a successful search. We present the *query processing rule* (QPR) in an algorithmic form.

Rule 2 *QPR : If (Message packet(M)) Start*
```
        V++;
        if ((V == 1) AND (P = M)), /*Report a match, V = 1 indicates first time
                                              visit by a message*/
         SS = 1
```

Once the queries are processed, the messages have to be forwarded to the neighboring cells through a *query forwarding rule*. Each of the proliferation and random walk techniques has a unique *query forwarding rule* which will be discussed in the next section. We sum up the common features of the CA model applicable across all the techniques.
CA Model - Summarization : The states, rules, neighborhood configuration and updation synchrony of each cell of the two dimensional CA is next summarized. A sketch of the summarization is also provided through *Fig. 1.*

A **States**
Token P : A 10-bit string which doesn't change over time.
Message Queue(MQ) : A queue of messages of infinite length initialized at the end of each search.
Visit Count(V) : A counter which tracks the number of messages visiting the cell.
Success Indicator(SS) : A one bit switch which is set to one if there is a successful search. Both SS and V are set to 0 at the beginning of each search.

B **Rules**
Query Initiation Rule (QIR) : During every search operation it is executed once only by a cell randomly selected to initiate the search.
Query Processing Rule (QPR) & *Query Forwarding Rule (QFR) :* These rules are executed by all the cells at each time step. QFR differs from process to process, which will be discussed in *Section 2.2.*

C **Neighborhood**
The CA has a Von-Neumann neighborhood configuration, that is each cell has *four neighbors.* The rules are periodic in nature, that is, in effect, the CA is a two-dimensional toroidal grid.

D **Updation Synchrony**
The CA is *updated asynchronously*, that is each cell gets updated sequentially. However, the sequence in which cell gets *updated is random* in nature. The asynchronous updation mimics the real-life environment, where peers in $p2p$ network update their states without maintaining any synchrony with the neighboring peers.

The query forwarding rules which take care of the manner in which a query should be forwarded to the next cell are next presented one by one.

2.2 Query Forwarding Rule (QFR)

In this section, we describe two proliferation based as well as two random walk based QFRs.
Proliferation (Pr) : In the proliferation scheme, the packets undergo proliferation at each cell 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.

Rule 3 Pr*: If(Message packet(M)) Start*

```
Produce $N_{new}$ message packets(M)
Spread the $N_{new}$ packets to $N_{new}$ randomly selected neighbors of A
```

The function determining the value of 'N_{new}' ensures that $N_{new} \leq 4$, where 4 is the number of neighbors of any cell. The function is discussed in detail after we elaborate the second proliferation rule which imposes some restriction in packet movement.
Restricted Proliferation (RP) : The rule RP, similar to Pr, produces N_{new} messages. 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 cells 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 is discarded. However, if $\mathcal{Z} = 0$, then one message is forwarded to a randomly selected neighbor.

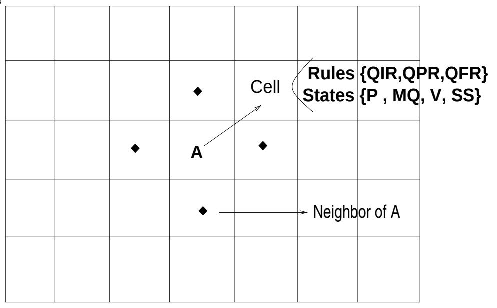


Fig. 1. The CA used to perform search operation with elaboration of the states, rules, and neighborhood of each cell

The rationale behind the restricted movement is to minimize the number of message wastage. Because, two packets of message M visiting the same cell essentially means wastage of the second packet.

Rule 4 RP *: If (Message packet(M)), Start*

```
Produce $N_{new}$ message packets (M).
$\mathcal{Z}$ = No of 'free' neighbors
if ($\mathcal{Z} \geq N_{new}$)
    Spread the 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 a message packet to a randomly selected neighbor of A
    Discard the remaining ($N_{new}$ - 1) packets
```

We now elaborate the function used to control the amount of proliferation.
Proliferation Controlling Function : The proliferation of message packets at any cell A is heavily dependent on the similarity between the message packet (M) and the token (P) of A. In this connection we define the following expression p, where $p = e^{-HD} \times \frac{\rho}{4}$, ρ represents the proliferation constant, it is same for all nodes; HD is the Hamming distance between the message (M) and the information profile (P).

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.

$$P(\eta) = \sum_{i=\eta}^{4} \binom{4-1}{i-1} \cdot p^{i-1} \cdot (1-p)^{4-i}$$

The significance of the above equation is explained through *Fig. 2*. The two figures show the behavior of $P(\eta)$, with respect to different HDs and the different values of *proliferation constant.* 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 illustrates that at least one packet necessarily proliferates. [Note: If one packet proliferates, the event is similar to that which occurs in random walk.] *Fig. 2(a)* shows the variation with respect to different values of proliferation constant ρ; when $HD = 0$; three different curves are drawn for $\rho = 4$, 3.5, 3. We see that for $\rho = 4$, 4 packets always proliferate; for $\rho = 3$, 3.5, the probability of proliferation decreases exponentially if $\eta > \rho$. *Fig 2(b)* shows the variation with respect to different Hamming distances for $\rho = 4$; 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 $HD > 0$.

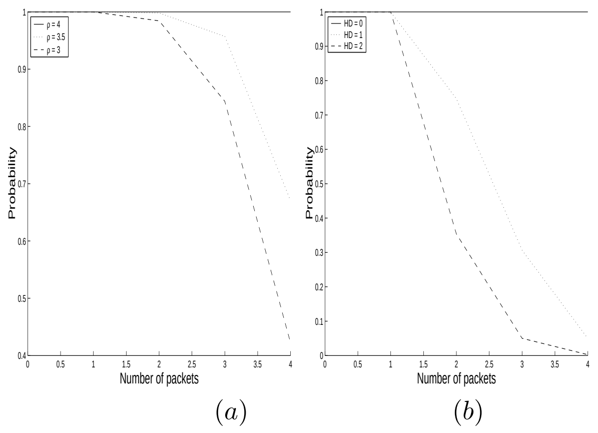


Fig. 2. Graphs plotting the probability of proliferation of at least η messages (a). when the value for proliferation constant varies ($HD = 0$) and (b). when Hamming Distance varies ($\rho = 4$)

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

k-random walk (RW) In k-random walk, the cell forwards the message packet to a randomly selected neighbors. The rule(RW) is represented as

Rule 5 RW *: If(Message packet(M)) Start*
Send the packet M to a randomly chosen neighbor peer

Similar to RP (*Rule 4*), we also design the restricted random walk (RRW) rule which is defined 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.

Rule 6 RRW *If (Message packet(M)) start*
Send the packet M to a randomly chosen 'free' neighbor peer
If no 'free' neighbor
Send the packet M to a randomly chosen peer

The next section describes various metrics used to compare performances of the algorithms.

2.3 Metrics

In this paper we focus on efficiency aspects of the processes solely, and use the following simple metrics in our abstract *p2p* networks. These metrics, though simple, reflect the fundamental properties of the different *query processing rules.*

- Success rate: The number of similar items (say $\mathcal{K}$) found by the query messages within a given time period, that is $\mathcal{K} = \sum_{i=1}^{N} SS_i$, where i represents each cell and N is the total number of cells.
- Coverage rate : The amount of time required by the messages to cover a percentage of the network. The network is fully covered when the visit field - $V_i \geq 1$ in each cell of the CA.
- Cost per search output : The number of messages required to output a successful search. In some sense, it is the inverse of the success rate, but it also provides different insights into the system, which we will discuss in the next section.
- Bad visits (BV) : The number of times the same message re-visits the same cell. If a message packet visits the same cell more than once, it amounts to wastage of that packet. $BV = \sum_{i=0,(ifV_i \geq 1)}^{N}$ (V_i - 1). So, a good design should minimize the amount of *bad visits.*

Based upon the above mentioned model and metric definition, we now present the experimental results.

3 Simulation Results

The experimental results compare efficiency of different *query processing rules* (rule 3 - 6). The efficiency comparison is carried out with respect to the metrics defined in *section 2.3.* Beside assessing efficiency, we also perform different scalability tests with the QFRs. In order to assess the efficiency of different QFRs, we have also to ensure fairness of 'power' among them which is elaborated next.

3.1 Fairness in Power

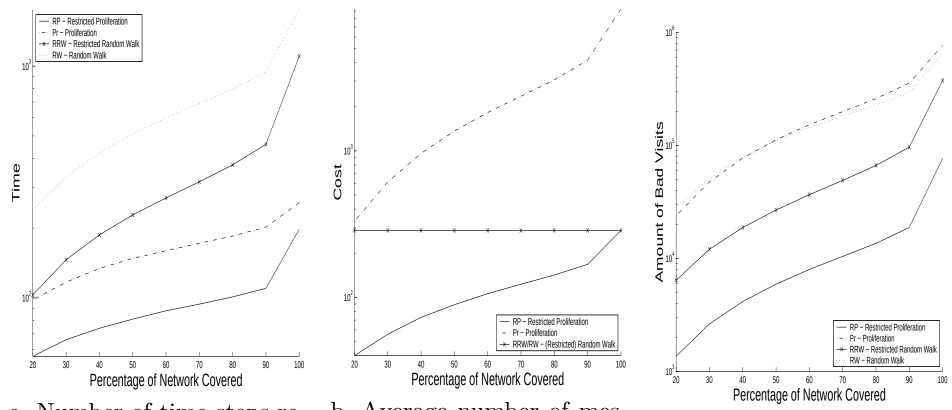
To ensure fair comparison among all the QFRs, we must ensure that each QFR (Pr, RP, RW, RRW) should participate in the network with the same 'power'. To provide fairness in 'power' between a proliferation QFR (say Pr) and a random QFR (say RW), we ensure that the total number of query packets used is roughly the same in both 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 increases in the proliferation algorithms over the generations, while it remains constant in the case of random walk algorithms. Therefore the number of message packets - k in *rule 1* is set in a fashion so that the aggregate number of packets used by each individual algorithm is roughly the same, that is, for RW

$$k_{RW} = \frac{Tot\_Pack[Pr]}{Tot\_Step[Pr]}$$

where $Tot\_Pack[Pr]$ is the total number of message packets used by Pr to perform a particular experiment, while $Tot\_Step[Pr]$ indicates the total number of time steps required to perform that task.

To ensure fairness in 'power' between two proliferation rules (Pr & RP), we keep the proliferation constant ρ and the value of k the same for both processes. The value of k is generally set as $k = 4$, where each cell has 4 neighbors.

We are now reporting the major experimental results one by one.



a. Number of time steps required to cover the grid by *RP*, *Pr*, *RRW* & *RW*

b. Average number of message packets used by *RP*, *Pr*, *RRW*/*RW*

c. Number of bad visits by *RP*, *Pr*, *RRW* & *RW*

Fig. 3. Graphs plotting average message used, network coverage time and number of bad visits by *Pr*, *RP*, *RW* and *RRW*

3.2 Experimental Result - Network Coverage

To evaluate the time taken by the message packets to visit all the cells of the CA using different QFRs, we consider a 100×100 two-dimensional CA comprising of 10000 cells and perform the following *coverage* experiment.
Coverage experiment : In this experiment, upon initiation of a search (*Rule 1*) from a randomly selected cell, the CA is allowed to run till the message packets cover the entire network (all cells of CA^3). The experiment is repeated 500 times on randomly selected initial cells. During the experiment, we collect different statistic at every 10% of coverage of the network.

Fig. 3 plots statistic collected through *coverage* experiments. *Fig. 3(a)* shows the network coverage rate of the RP, Pr algorithm at $\rho = 3$, as well as RRW and RW algorithms. 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). From the figure, it is seen that the time taken to cover the network is much shorter in RP followed by Pr, RRW and RW.

Fig. 3(b) plots the average number of packets used by the different QFRs (y-axis) vs. network coverage (x-axis). As expected, the number of packets increase in RP/Pr over the period of network coverage while it remains constant for random walk (RW/RRW) schemes. Therefore, while performing *coverage* experiments, the value of k for RRW/RW is set to the average number of message packets used by RP to cover 100% of the network. Although the proliferation constant is same for Pr and RP, from the figure it is seen that Pr produces enormous amount of messages, almost 10 times more than RP. This happens because Pr indiscriminately proliferates without checking whether the neighboring cells are already visited or not. The tendency is further detailed through the next figure (*Fig. 3(c)*).

Fig. 3(c) shows the number of *bad visits* (defined on page 6) by RP, Pr, RRW and RWs (y-axis) vs. network coverage (x-axis). It is seen that execution of both RW and Pr results in a huge amount of *bad visits*. This is because, since the rules don't have any restriction, they have a tendency to visit the same node again and again. However, even though both RRW and RP generally inherently tries to avoid already visited nodes, we find that RP can avoid such visits much more efficiently. (The number of bad visits by RRW

[3] The term 'network' and 'CA' are used interchangeably throughout the section

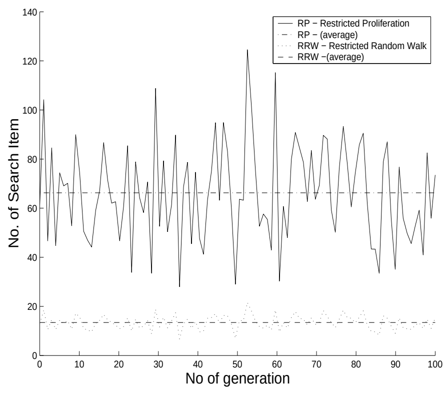
is 10 times higher than *RP*.) Therefore, we can conclude that *RP* by far outperforms all other techniques in terms of network coverage.

The next experimental results highlight the search efficiency of *RP* and *RRW*.

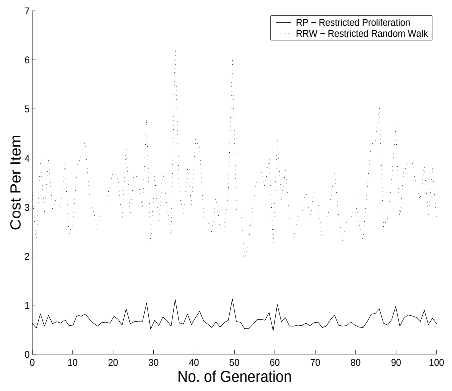
3.3 Experimental Results - Search Efficiency

To compare the search efficiency of *RP* & *RRW*, we take a 100 × 100 two-dimensional *CA* and perform a *time-step* experiment. The *time-step* experiment is elaborated next.
Time-step experiment : In the experiment, after initiation of a search (*Rule 1*), the *CA* is allowed to run for $\mathcal{N}$ (= 50) time steps. The search is initiated by a randomly selected node and the number of similar 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 $N_s = \sum_{i=1}^{100} n_s/100$. The value of N_s, provides the indication of search efficiency. *Fig. 4* is the result of running the *time-step* experiment for 100 generations.

The graph of *Fig. 4(a)* displays the average value N_s against generation number for *RP* and *RRW*. In this figure we see that the search results for both *RP* 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 *RP* is almost 5-times higher than that of *RRW*. (For *RP*, the number of hits ≈ 65, while it is ≈ 13 for *RRW*.) The fluctuations in the results help us to understand an important aspect about cost which is discussed next.



a. Search efficiency of *RP* ($\rho = 3$) and *RRW*



b. Cost incurred per item searched by *RP* ($\rho = 3$) and *RRW*

Fig. 4. Graphs showing search efficiency and cost incurred per item searched by *RP* and *RRW*

Fig. 4(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. 4(a)*. We see that the cost of *RP* is hardly changing (it stays constant at around 0.7) 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. Therefore, when the search output is low, the cost shoots up sharply. While in *RP*, 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, *RP* produces a lower number of packets and vice versa.

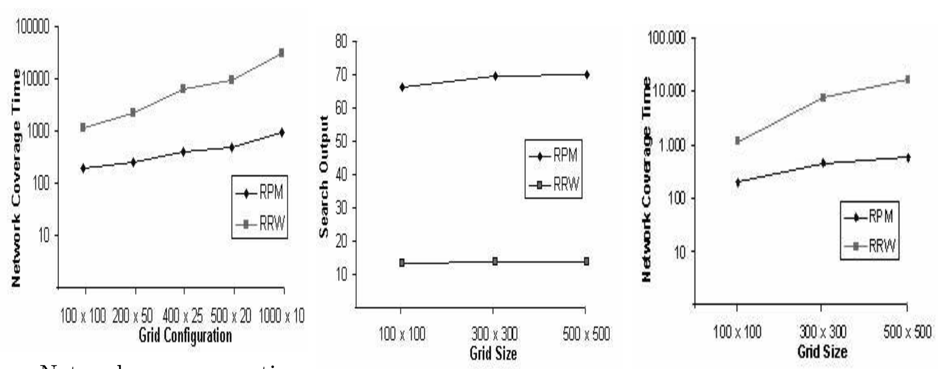
The next set of experiments investigates whether *RP* is scalable.

3.4 Experimental Result - Scalability

In this section, we describe the different scalability tests performed. Two different types of scalability tests are chosen. In the first type, we change the shape of the two-dimensional CA so far used, from the square CA (100×100) to a more rectangular shape ($200 \times 50, \cdots$), and observe the impact it creates on the efficiency of RP & RRW. In the second case, we increase the dimension of the CA (from 100×100 to $300 \times 300 \cdots$) and observe the outcome.

Shape and Scalability : In this experiment, we consider different two-dimensional CAs each having 10000 cells. However, the shape of each CA differs. The dimensions of such CAs are respectively 100×100, 200×50, 400×25, 500×20, 1000×10. Testing the efficiency in all such configurations is particularly important because in real *p2p* environments, we may not be able to develop exactly square connection among the peers and it is necessary for a proposed algorithm to perform well irrespective of the nature of connection layout. We perform the *coverage* experiments (elaborated in *Section 3.2*) on these differently shaped CA; the experimental results are illustrated in *Fig. 5(a)*.

Fig. 5 plots the network coverage time (y -axis) taken by RP and RRW against different configuration (x -axis). For each, RP and RRW, we plot the time taken to cover all the cells of the CA. The figure shows that for both RP and RRW, as the shape becomes more rectangular, the network coverage time increases. For example, RP takes 198 time steps to cover the network when the CA is square, whereas it takes 1951 time steps when the CA has the dimension 1000×10. Whereas in case of RRW, the respective figures rise from 1105 to 31025. Therefore, we see that RP has a five-fold rise in network coverage time, while for RRW it increases by a factor of (around) 30. So, we can conclude that performance of RP is much less insensitive to change in shape than RRW.



a. Network coverage time taken by RP and RRW with different shape of CA, each CA having 10000 cells

b. Search efficiency of RP and RRW at different CA size

c. Network Coverage time taken by RP and RRW at different CA size

Fig. 5. Graphs illustrating scalability of RP and RRW under different conditions

Size and Scalability : To measure the performance of RP and RRW with respect to CAs of different sizes, we consider CAs of dimension 100×100, 300×300 and 500×500. We perform both *time-step* and *coverage* experiments with RP and RRW on the three CAs. In case of the *time-step* experiment, a scalable algorithm should ensure that the number of search outputs does not deteriorate with the increase in dimension. *Fig. 5(b)* shows the result of the *time-step* experiment on the three different grids. It plots the average number of search output (y-axis) by RP and RRW on different grid sizes (x-axis). The result shows that search efficiency for both RW and RP is independent of grid dimensionality. It is seen that the output of RRW virtually doesn't change while for RP, in fact, it improves slightly when the grid size increases from 100×100 to 300×300. However, the output of RP virtually doesn't change when the grid size increases from 300×300 to 500×500. The reason for initial enhancement of performance when the grid size increases from 100×100 to 300×300 is as follows. Since the CA in effect is a toroidal grid, the diameter of the 100×100 CA is 50. The *time-step* experiment is carried on for 50 time steps which results in some packets covering the entire length of the grid and colliding (visiting the same cells) with packets which have traveled to the edges from the opposite direction. This naturally decreases the output. Whereas when the grid is of size 300×300 or 500×500, there is no possibility of such collision. Note that the packet movement is much faster in RP than RRW, as we don't see any improvement in RRW, thus can conclude the packets in RRW doesn't reach the edges within 50 time steps.

The more important test of scalability is elaborated through the next result derived by executing a *coverage* experiment and is depicted in *Fig. 5(c)*. *Fig. 5(c)* plots the time (y -axis) taken by RP and RRW to cover 100% of the network against different configuration (x -axis). The coverage time in RP increases from 198 (for 100×100) to 586 (for 500×500). The rate of increase is less than the logarithm of the rate of increase in the number of nodes. Any search algorithm in *p*2*p* network with such low rate of increase is considered to be extremely scalable [4]. However, for RRW, the network coverage time increases from 1105 to around 16161, increasing almost linearly with the number of nodes.

4 Conclusion

Peer to peer network is increasingly becoming a very popular component of the Internet. One of the most important functionality of *p2p* network is *search.* In this paper, we have concentrated on developing efficient search algorithms for *p*2*p* networks. We have produced detailed experimental results showing that the simple immune-inspired concept of proliferation can be used to cover the network more effectively than random walk. Moreover, the search efficiency of rule based on proliferation (RP) is five times higher than that of restricted random walk (RRW) algorithms. The proliferation algorithm is extremely scalable. There is only minor deterioration in network coverage time with respect to changing shape and size. The effectivity of proliferation rules 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 recent future to explore the full potential of proliferation algorithm.

References

1. Napster. *http://www.napster.com*, 2000.
2. Gnutella. *http://www.gnutellanews.com*, 2001.
3. I. Clarke, O. Sandberg, B. Wiley, and T. W. Hong. Freenet: A Distributed Anonymous Information Storage and Retrieval System. In *Designing Privacy Enhancing Technologies: International Workshop on Design Issues in Anonymity and Unobservability*, pages 46–59, 2000.
4. A Deutsch, N Ganguly, G Canright, M Jelasity, and K Monsen. Models for advanced services in AHN, P2P Networks. Bison Deliverable, www.cs.unibo.it/bison/deliverables/D08.pdf, 2003.
5. E. Klarreich. Inspired by Immunity. *Nature*, 415:468 – 470, 2002.
6. Q. Lv, P. Cao, E. Cohen, and S. Shenker. Search and Replication in Unstructured Peer-to-Peer Networks. In *Proceedings of the* 16^{th} *ACM International Conference on Supercomputing*, June 2002.
7. H. Unger and M. Wulff. Cluster-building in P2P-Community Networks. In *Parallel and Distributed Computing and Systems (PDCS 2002)*, pages 685–690, 2002.
8. G. K. Zipf. *Psycho-Biology of Languages.* Houghton-Mifflin, 1935.