

Random Walks and Cover Times

Project Report

Aravind Ranganathan

ECES 728 Internet Studies and Web Algorithms

1. Objectives:

We consider random walk based broadcast-like operation in random networks. First, we verify if partial covering is indeed better than complete covering. Then, we analyze two types of random networks and determine the best suited random walk covering technique for each type of networks. Type-A networks are similar to wired networks where an edge between two nodes indicates a direct exclusive connection between the nodes. Type-B networks on the other hand are similar to wireless networks where any message broadcast by a node is simultaneously available to all of its neighbors. Our primary objectives are:

1. Compare complete covering and partial covering
2. Simulate Unbiased and p-biased Random walks
3. Determine which random walk covering technique is best suited for each type of networks

2. Network Construction:

We consider randomly generated networks that are neither too small, nor too big (typically having about 1000 nodes). Ad-hoc techniques can be used for smaller networks while it has been shown that partial cover time (PCT) works better for much larger networks^[1]. We now define some terms that are widely used in the following sections:

2.1 Definitions:

Cover Time (CT): It is the expected number of steps required by a random walk to visit all the nodes in the network.

Partial Cover Time (PCT): It is the expected number of steps required by a random walk to visit a constant fraction of the nodes in the network, say 80%.

Type-A Networks: They are similar to wired networks where an edge between two nodes indicates a direct non-shared connection between the nodes.

Type-B Networks: They are similar to wireless networks where any message broadcast by a node is simultaneously received by all its neighbors.

Unbiased random walk: The sink node starts the broadcast and it randomly selects one of its neighbors to send the packet. Whenever a node gets the packet, it uniformly randomly selects one of its neighbors and passes the packet to the selected neighbor. A node that gets the packet is said to have been “visited”.

p-biased random walk: Every node keeps track of which of its neighbors have been visited. Whenever a node receives a packet, it chooses one of its unvisited neighbors with a probability $((p/d_u)+((1-p)/d))$ where ‘d’ is its number of neighbors and ‘d_u’ is its number of unvisited neighbors.

3. Random walks in Type-A and Type-B Networks:

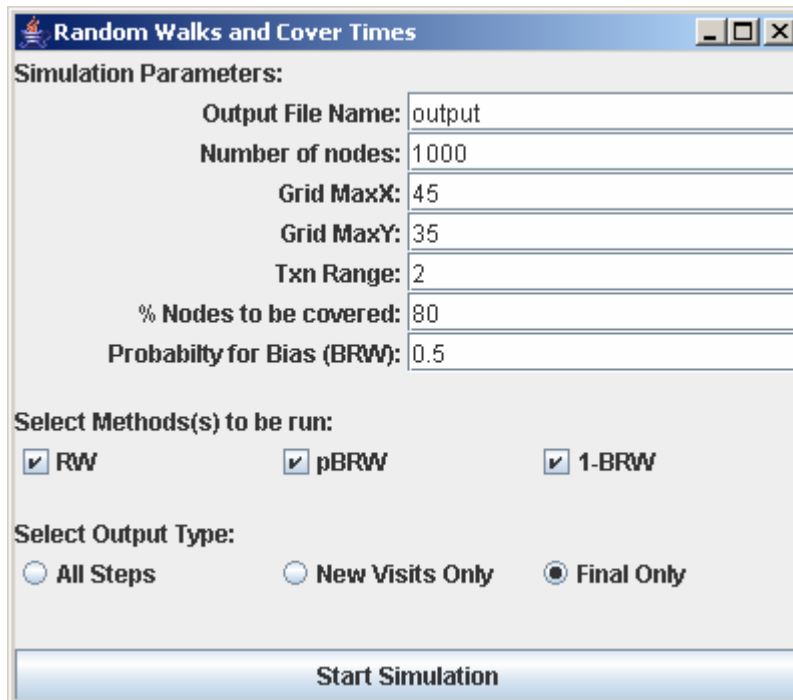
A typical example of a Type-A network would be a wired network (like the internet) while a wireless sensor network would be an example for a Type-B network.

Unbiased random walks do not have any adverse effect on Type-A networks. However, it causes a lot of unwanted congestion in Type-B networks. This is because of the “hidden terminal” and the “exposed terminal” problems associated with such networks where whenever a node transmits a packet, all its active neighbors receive the data and hence cannot transmit / receive data of their own until the first node finishes its transmission.

A p-biased random walk requires that each node maintain not only a list of neighbors but also mark which of its neighbors have already been visited. In Type-A networks, this would imply that each node send an “I HAVE BEEN VISITED” message to all of its neighbors, whenever it receives a packet. We can reduce the amount of these redundant packets by allowing each node to send the “I HAVE BEEN VISITED” packet to all its neighbors except the node from which it got the packet and the node to which it is going to pass the packet as it would be redundant. This would mean that each node with ‘d’ neighbors would have to send ‘d-2’ additional packets. This increases the congestion in the network. However, it is not the case with Type-B networks as whenever a node transmits a packet, all its neighbors automatically know that the node was visited.

4. Simulations Setup:

The simulations were coded in Java using Eclipse 3.0 IDE. The input parameters were obtained in the following screen:



The screenshot shows a Java dialog box titled "Random Walks and Cover Times". It contains several input fields and checkboxes. The "Simulation Parameters" section includes: "Output File Name" (output), "Number of nodes" (1000), "Grid MaxX" (45), "Grid MaxY" (35), "Txn Range" (2), "% Nodes to be covered" (80), and "Probability for Bias (BRW)" (0.5). The "Select Method(s) to be run:" section has three checked checkboxes: "RW", "pBRW", and "1-BRW". The "Select Output Type:" section has three radio buttons: "All Steps", "New Visits Only", and "Final Only" (which is selected). A "Start Simulation" button is at the bottom.

Parameter	Value
Output File Name	output
Number of nodes	1000
Grid MaxX	45
Grid MaxY	35
Txn Range	2
% Nodes to be covered	80
Probability for Bias (BRW)	0.5

Select Method(s) to be run:

- RW
- pBRW
- 1-BRW

Select Output Type:

- All Steps
- New Visits Only
- Final Only

Start Simulation

Fig 4.1. Input Parameters for the simulation

The default values specify that a network of 1000 nodes be deployed in a 45x35 unit² grid where nodes at an Euclidian distance of less than 2 units are connected and that the simulation be run till 80% of the nodes are covered for all the 3 techniques – unbiased random walk, 0.5-biased random walk and 1-biased random walk. The output graph shows only the final status of the random walk for each of the three techniques.

Once the simulation is started, a random graph is first created. Since the random graph may have disconnections, if the size of the biggest component is more than 90% of the size of the graph, a new graph with just the biggest component is chosen as the final graph on which the simulations are run. Hence there is a possibility that the final graph has only 90% of the number of nodes specified in the simulation. However, for the values we chose (typically the default values), most of graphs turned out to be 98% or more connected.

The following figures show the disconnections in a sample initial random graph with 100 nodes and the final graph with only the connected 93 nodes belonging to the biggest component. The red node indicates a randomly chosen sink node in the biggest component.

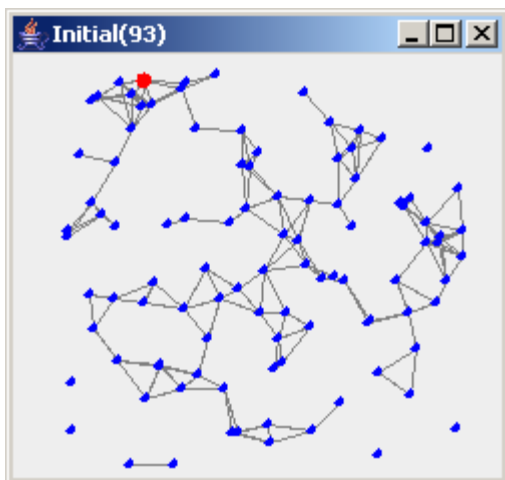


Fig 4.2.a: Initial random graph with 7% disconnections

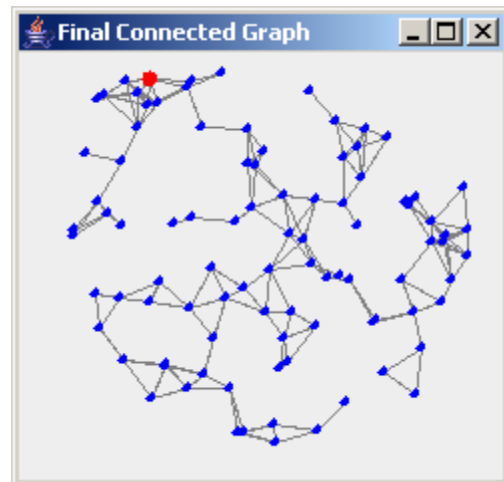


Fig 4.2.b: The same graph after the disconnections were removed

Now, based on the input parameters, the simulation was run for the same graph for different covering techniques and the cover times (or the partial cover times).

4.1 Sample Output:

The outputs obtained for a sample simulation run are shown in the following figures:

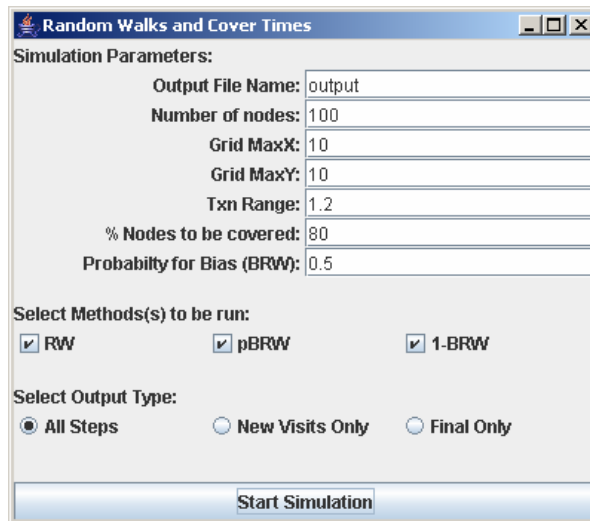


Fig 4.3.a: Input Parameters for the simulation

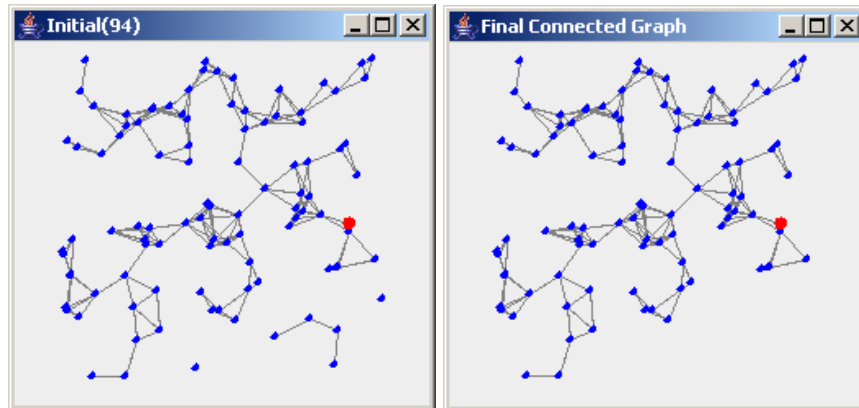


Fig 4.3.b: Initial random graph with 6% disconnections

Fig 4.3.c: The same graph after the disconnections were removed

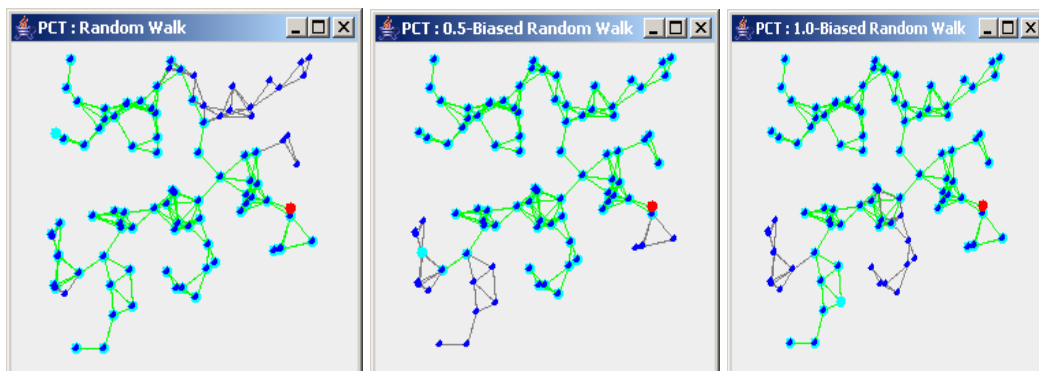


Fig 4.3.d: Graphs indicating Random Walk Path, Visited Nodes for each technique.

Note: Visited nodes have cyan background and visited edges are in green.
The Sink node is in red, the final node is completely in cyan.

4.2 Interesting Observation in Unbiased vs. 1-biased random walks:

After running the simulations with the Output Type option set to “All steps” (see Fig 4.1), we observed an interesting dissimilarity in the way new nodes were visited in an unbiased random walk and that of a 1-biased random walk.

We noticed that in an unbiased random walk, nodes nearer to the sink node were visited prior to those far away from the sink node. In other words, nodes closer to the sink node seemed to have a higher probability of being visited and were almost always visited while nodes farthest away from the sink were among the last of the nodes to be visited.

However, in the case of 1-biased random walks, it was quite the contrary. The walk was observed to go away from the sink in any direction until it hit a boundary after which, it would come towards the sink and then branch out in some other direction.

The observations make intuitive sense. In a 1-biased random walk, initially, a node has more neighbors going away from the sink and hence the walk tends to move away from the sink until it hits a boundary when it is forced to choose a path back towards the sink. These observations have been captured below:

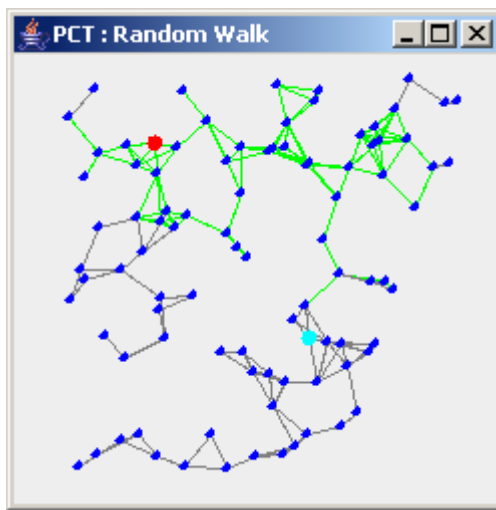


Fig 4.3.a: Unbiased Random walk

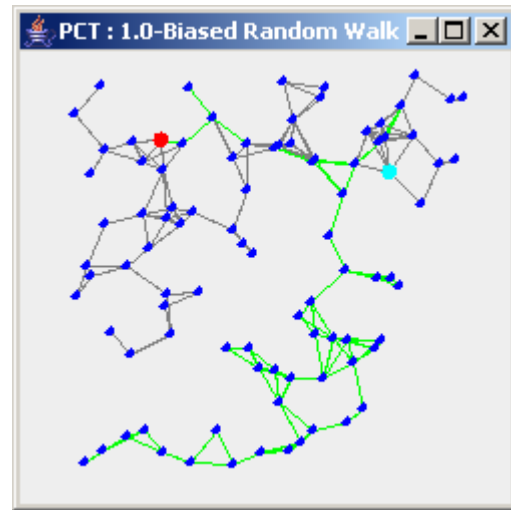


Fig 4.3.b: 1-biased Random walk

Note that the 1-biased random walk has covered the longest path away from the sink while the unbiased random walk has been covering more nodes closer to the sink. The edges of the random walk are in green.

We also noticed that the node covering of a p -biased random walk with $0 < p < 1$ lies in-between the two and is dependent on the value of ‘ p ’.

5. Results:

5.1. Cover time vs. Partial Cover Time:

The simulation was run with default values for 25 different graphs and the average cover time (for 100% cover) and partial cover time (for 80% covering) for different random walk techniques were calculated. The results are as follows:

	Average Cover Time	
	100% Cover	80% Cover
RW	62304.64	8166.92
0.25-BRW	43459.16	4913.76
0.5-BRW	29646.12	3273.36
0.75-BRW	25882.44	2967.96
1.0-BRW	23185.60	2317.48

Table 5.1: Cover Times for 100% vs. 80% Covering

It can be seen that it takes only about 1/9th the total cover time for 100% covering to cover 80% of the nodes. Thus, partial cover time for 80% covering is about 9 times faster than 100% cover time.

5.2. Unbiased and p-biased Random walks:

For the same 25 graphs, we computed the minimum, maximum and the average PCT (for 80% covering) for unbiased random walks as well as for p-biased random walks for p=0.25, 0.5, 0.75 and 1. The following graph shows the results:

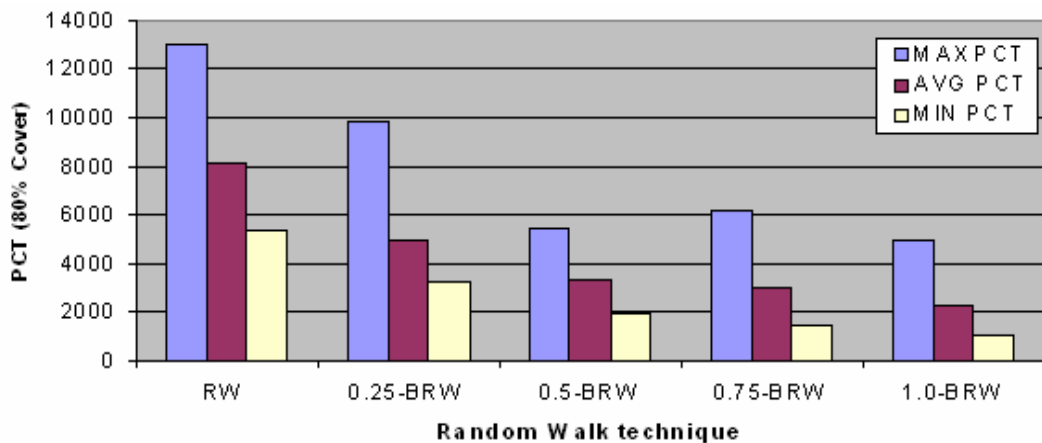


Fig 5.1: Average, Minimum, Maximum PCT for different Techniques

Clearly p-biased random walk performs increasingly better as more and more bias is introduced. A 1-biased random walk takes about 4 times fewer steps than an unbiased random walk. As discussed earlier in section 3, this will contribute to increased congestion in Type-A networks as each node has to send additional “I

HAVE BEEN VISITED” messages to its neighbors. The following graph looks at the average number of additional messages in the network in order to maintain the status of a node’s neighbors.

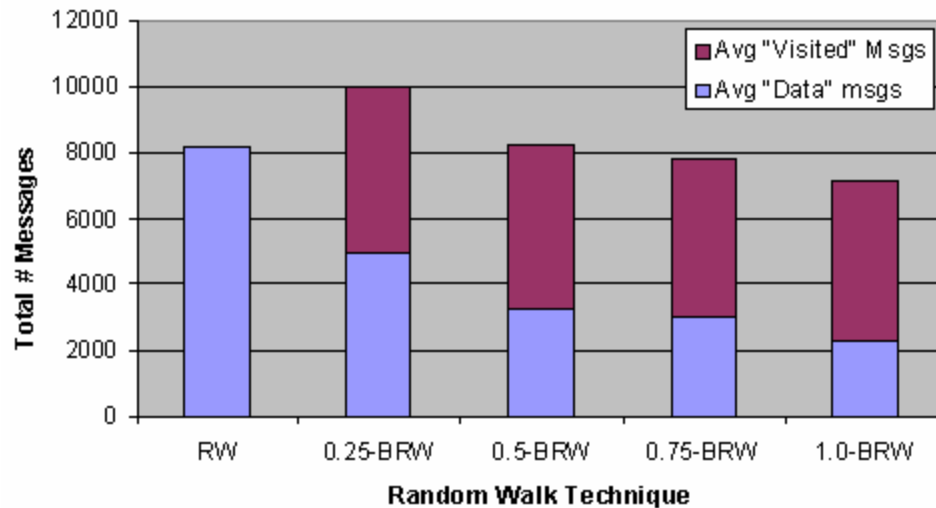


Fig 5.2: Average total messages in the network

Surprisingly, 1-biased random walk has fewer total messages than an unbiased random walk. The size of an “I HAVE BEEN VISITED” message can be safely assumed to be no greater than that of the data. Also, these additional messages are not needed for Type-B networks.

Hence, overall, 1-biased random walk scheme performs better.

6. Conclusions:

The simulation results are in accordance with the expected results. Simulations show that:

1. Complete covering is much slower than partial covering. It takes about 9 times more steps to cover the last 20% of the nodes than the first 80% of the nodes.
2. For Type-A Networks:
 - a. 1-biased random walk performs much better even with the increased congestion in the network.
3. For Type-B Networks:
 - a. 1-biased random walk performs the best and doesn’t even cause increase in network congestion.

Overall, 1-biased random walk with partial covering is the best technique to be used for both Type-A as well as Type-B networks.

7. References:

- [1] “Efficient and robust query processing in dynamic environments using random walk techniques” - Chen Avin and Carlos Brito

