International journal on applications of graph theory in wireless ad hoc networks and sensor networks (GRAPH-HOC) Vol.6, No.1, March 2014

ENHANCED RANDOM WALK WITH CHOICE: AN EMPIRICAL STUDY

John Alexandris[1] and Gregory Karagiorgos[2]

[1] Software engineer, Mauromichaleon 32 Halandri, 15233, Athens, Greece
[2] Department of Computer Engineering,Technological Institute of Peloponnese, Branch of Sparta, Kladas, 23100 Sparta, Greece

ABSTRACT

The Random Walk with d Choice $RWC(d)$ is a recently proposed variation of the simple Random Walk that first selects a subset of d neighbor nodes and then decides to move to the node which minimizes the value of a certain parameter; this parameter captures the number of past visits of the walk to that node. In this paper, we propose the Enhanced Random Walk with d Choice algorithm $ERWC(d,h)$ which first selects a subset of d neighbor nodes and then decides to move to the node which minimizes a value H defined at every node; this H value depends on a parameter h and captures information about past visits of the walk to that node and - with a certain weight - to its neighbors. Simulations of the Enhanced Random Walk with d Choice algorithm on various types of graphs indicate beneficial results with respect to Cover Time and Load Balancing. The graph types used are the Random Geometric Graph, Torus, Grid, Hypercube, Lollipop and Bernoulli.

KEYWORDS

Random Walk, Power of Choice, Cover Time, Load Balancing, Wireless networks.

1. INTRODUCTION

Recently there is a growing interest in random walk-based algorithms, especially for a variety of networking tasks (such as searching, routing, self stabilization and query processing in wireless networks, peer-to-peer networks and other distributed systems [1,9]), due to its locality, simplicity, low overhead and inherent robustness to structural changes. Many wireless and mobile networks are subject to dramatic structural changes caused by sleep modes, channel fluctuations, mobility, device failures and other factors. Topology driven algorithms are inappropriate for such networks, as they incur high overhead to maintain up-to-date topology and routing information and also have to provide recovery mechanisms for critical points of failure. By contrast, algorithms that require no knowledge of network topology, such as random walks, are advantageous.

A random walk on a graph is the process of visiting the nodes of a graph in some sequential random order. The simple random walk starts at some fixed node (with uniform probability among all nodes in the network) and at each time step, it moves to a randomly chosen neighbor of the currently visited node. The simple random walk is a totally uncontrolled process, which often shows unwanted behavior, like frequent revisits of already covered nodes and substantial delays in visiting the most isolated regions within a network.

DOI:10.5121/jgraphoc.2014.6101

Motivated by the need to *improve random walk-based performance over graphs*, researchers have proposed various methodologies, to reduce the Cover Time of a random walk. A recently proposed methodology in [8] suggests that the use of multiple, parallel random walks starting from a fixed vertex of the graph will speed up the cover process. The authors prove that running many random walks in parallel yields a speed-up which is linear in the number of parallel walks. They also demonstrate that an exponential speed-up is sometimes possible, but that some natural graphs allow only a logarithmic speed-up.

A further methodology for reducing Cover Time on graphs is the use of the Power of Choice. The essential idea behind the power of choice, is to make some decision process more efficient, by selecting the best among a small number of randomly generated alternatives. The most basic results [22] about the power of choice are as follows. Suppose that n balls are placed randomly into n bins. Let the load of a bin be the number of balls in that bin after all balls have been thrown. What is the maximum load over all bins once the process is terminated? It is well known that, with high probability, the maximum load upon completion will be approximately $\frac{\log n}{\log \log n}$. We now state a surprising result proved in a seminal paper by Azar, Broder, Karlin, and Upfal [11]. Suppose that the balls are placed sequentially so that for each ball we choose 2 bins independently and uniformly at random and place the ball into the less full bin (breaking ties arbitrarily). In this case, the maximum load drops to $\frac{\log \log n}{\log 2} + O(1)$ with high probability. If each ball has $d \geq 2$ choices instead, then the maximum load will be $\frac{\log \log n}{\log d} + O(1)$ with high probability. Having two choices hence yields a qualitatively different type of behavior from the single choice case, leading to an exponential improvement in the maximum load; having more than two choices further improves the maximum load by only a constant factor.

Chen Avin and Bhaskar Krishnamachari in their paper ***The power of choice in Random Walks: An Empirical Study*** [6] proposed to combine random walk on a graph with the Power of Choice. The Random Walk with d Choice algorithm $RWC(d)$ works in such a way that it selects d neighbors uniformly at random and then chooses to step to the node u of the d neighbors that minimizes the fraction $\frac{c^t(u)+1}{\delta(u)}$, where $c^t(u)$ is the number of visits of the random walk at time t at node u and $\delta(u)$ is the degree of node u.

If the graph is regular, the walk steps to the least-visited neighbor; if not, the walk steps to the node that is farthest away from its stationary distribution $\pi(u)$. For the complete graph the analytical results show that the cover time of $RWC(d)$ is reduced by a factor of d, compared with the cover time of the simple random walk.

For general graphs the lack of Markov property suggests that the analytical results may be harder to obtain. The simulation-based study shows a consistent improvement in the cover time, cover time distribution and the load balancing at cover time for different graphs and different sizes. A surprising result is that for the 2-dimensional torus, choice seems to improve the cover time and the load on the most visited node by an unbounded factor. Specifically while the cover time of the n nodes torus is known to be $\Theta\left(n \log^2 n\right)$, the simulations showed that with $d = 2$, random

walk with choice has lower cover time than the simple random walk on the hyper-cube, that is known to have optimal cover time $\Theta(n \log n)$.

In this paper we propose the ***Enhanced Random Walk with d Choice*** $ERWC(d,h)$ algorithm. (A preliminary version of this work is in [4] and in [5]). We introduce, additionally to d, a new parameter $h > 1$. Let $H^t(v)$ be a value defined to node v at time t of the enhanced random walk algorithm depending on parameter h as follows:

$$H^t(v) = h * (\text{number of visits at node } v \text{ at time t}) + (\text{number of visits at neighbors of node } v \text{ at time } t)$$

The main idea behind $ERWC(d,h)$ is to steer the course of the random walk using not only the number of visits to the candidate node it is considering to move to (as it is the case with the $RWC(d)$, but a more comprehensive metric that captures the level of past passage through the broader region around the potential candidate node as well. This way, it is expected that the random walk will avoid highly visited regions and, thus, it will be likely to enter less visited regions incurring in less revisits to the nodes.

In order to represent the intensity of visits to a region, we introduce the idea of recording the "trail" of a random walk through a region by increasing a counter of a visited node by $h > 1$ and that of its neighbors by 1. The more frequent and the closer the passage of a random walk is to a certain node, the higher the accumulated value of its counter, H, will be expected to be. By randomly selecting d nodes, the random walk selects d potential directions to move to its next step. From these d possibilities, the random walk will consider first the subset of nodes never visited before and will select among them the one with the lowest value of H divided by the degree of the node; if the latter subset is empty, it will select the neighbor (direction) with the lowest value of H divided by the degree of the node. With these moving rules, the random walk is directed towards the unvisited nodes from the selected d, which are located in the least visited region (lowest value of H divided by the degree of the node); or else, it is directed towards the revisited node from the selected d, which is located in the least visited region.

At any point in time t the enhanced random walk algorithm selects d neighbors uniformly at random and then chooses to step to an unvisited node among them (if one exists). If all of the selected d nodes have already been visited the random walk chooses to step to the node u of the d neighbors than minimizes the fraction $\frac{H^t(u)}{\delta(u)}$, where $\delta(u)$ is the degree of node u.

Our result, based on simulations, is that the Enhanced Random Walk with d Choice algorithm outperforms the Random Walk with d Choice in the following ways:

1. For the Random Geometric, Grid and Torus graphs the performance improvement is in both Cover Time and Load Balancing at Cover Time.
2. For the Hypercube, Lollipop and Bernoulli graphs the performance improvement is only at Cover Time. For Lollipop graphs there is a small improvement in Load Balancing at Cover Time. For Hypercube and Bernoulli graphs the Load Balancing of the Enhanced

Random Walk with d Choice is worse by comparison with the Load Balancing of the Random Walk with d Choice.

The rest of the paper is organized as follows: Section 2 gives background information and definitions on graphs, random walk algorithms and the metrics of interest, associated with random walks on graphs. Section 3 formally introduces the Enhanced Random Walk with d Choice ($ERWC(d,h)$) algorithm. In Section 4 simulation results are presented for many types of graphs, including Random Geometric Graph, Grid, Torus, HyperCube, Lollipop and Bernoulli. Conclusions and Future Work are given in Section 5.

2. Background, Definitions and Metrics

Let $G(V,E)$ be an undirected graph with V the set of nodes and E the set of edges. Let $|V|=n$ and $|E|=m$. For $v \in V$ let $N(v)=\{u \in V \mid (uv) \in E\}$ be the set of neighbors of v and $\delta(v)=|N(v)|$ the degree of v. A δ - regular graph is a graph in which the degree of all nodes is δ.

2.1 Random Walk on Graphs

Simple Random Walk. The Simple Random Walk (SRW) is a walk where the next node to visit is chosen uniformly at random from the set of neighbors of a currently visited node. That is, when the walk is at node v the probability to move in the next step to node u is $P(v,u)=\frac{1}{\delta(v)}$ for $(v,u) \in E$ and 0 otherwise. If $\{v_t : t=0,1,2,....\}$ denotes the node visited by the SRW at step t then the walk can be described with a Markov chain. The Simple Random Walk is attractive due to its simplicity and robustness, yet it lacks in performance inducing high Cover Time and bad Load Balancing.

Random Walk with d Choice. The Random Walk with d Choice, $RWC(d)$ has been introduced in [6]. It is a walk whose next node to move is determined as follows: Let v denote the node reached by the walk at time t; let $c^t(v)$ be the number of visits to node v until time t; let $N(v)$ be the set of neighbor nodes connected to node v.

Algorithm 1 RWC(d) Algorithm: Upon visiting node v at time t, the $RWC(d)$:

1. Selects d nodes from $N(v)$ independently and uniformly at random
2. Steps to node u that minimizes $\frac{c^t(u)+1}{\delta(u)}$ (breaking ties in an arbitrary way)

The Random Walk with Choice has been shown in [6] to improve the Cover Time without losing the locality, simplicity and robustness of the random walk. This is an important goal, since it is directly related to the performance and energy usage of any random walk-based mechanism in a wireless network.

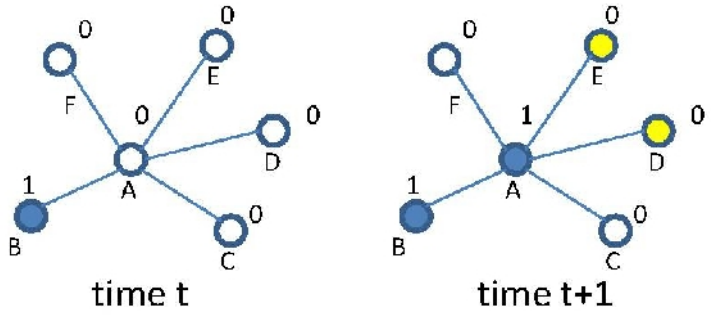

Figure 1: Execution of the $RWC(d)$ algorithm on a test graph

Figure1 shows graphically the execution of the $RWC(d)$ algorithm with $d = 2$. Snapshots of the algorithm execution are taken at time steps t and $t+1$ on an example graph. At time t the random walk visits node B and then moves to node A at time $t+1$. The associated values of $c^t(.)$ are shown. The yellow (shadowed) nodes represent two randomly selected candidate neighbors among which the selection for next movement of the random walk will be made.

2.2 Types of graphs used in simulation

We obtain simulations results for the types of graphs listed below:

1. *Random Geometric Graph,* $G(n,r)$. Random Geometric Graphs $G(n,r)$ result from placing n points uniformly at random on the unit square and connecting two nodes if and only if their Euclidean distance is at most r. Let r_{con} be the critical radius to guarantee connectivity w.h.p. It is known that r_{con} increases as $\mathrm{O}\left(\sqrt{\frac{\log n}{\pi n}}\right)$ [19]. Recently, it has been proven that, when $r = \Theta(r_{con})$ then w.h.p. $G(n,r)$ has optimal Cover Time $\mathrm{O}(n \log n)$ and optimal Partial Cover Time $\mathrm{O}(n)$ [7] for the simple random walk. The random geometric graphs have been widely used to model link connectivity and protocol behavior in randomly deployed wireless networks.
2. *Grid,* $G(n_1,n_2)$. A 2-dimensional Grid $G(n_1,n_2)$ is a graph in which the vertices are arranged on a rectangular 2 dimensional array with dimensions n_1 and n_2. The total number of nodes n is $n_1 * n_2$. It is known to have non-optimal Cover Time $\Theta(n \log^2 n)$ for the simple random walk [14].
3. *Torus,* $T(n_1,n_2)$. A 2-dimensional torus $T(n_1,n_2)$ is a graph in which the vertices are arranged on a Grid $G(n_1,n_2)$ with the additional property that the vertices on opposite sides of the boundaries of the Grid are connected. The number of nodes n is $n_1 * n_2$. It is known that has non-optimal Cover Time $\Theta(n \log^2 n)$ for the simple random walk [14].

4. *Hypercube graph,* H_n. The Hypercube graph with n nodes can be constructed by labeling these nodes as $0,1,2,...,2^n-1$ and connecting them if their Hamming distance equals 1. Hypercube graph is a $\log_2 n$-regular graph. H_n is known to have optimal Cover Time $\Theta(n\log n)$ for simple random walks [23].
5. *Lollipop,* $L(n_1,n_2)$. The Lollipop graph can be created by joining a complete graph K_{n_1} to a boundary vertex of a path graph P_{n_2}. The number of nodes n is n_1+n_2. It is known to have the worst case Cover Time of $\Theta(n^3)$ for the simple random walk [7].
6. *Bernoulli Graph,* $B(n,p)$. The Bernoulli graph $B(n,p)$ (a.k.a. Erdős-Rényi graph) is a random graph with n nodes in which each edge (out of $\binom{n}{2}$ possible edges) is chosen independently at random with an edge probability $p(n)$ [10]. These graphs were first offered by Gilbert [18] and have been thoroughly investigated since the seminal work of Erdős and Rényi [15,12]. It has been shown that $G(n,r)$ and $B(n,p)$ graphs have closely related critical connectivity thresholds for the radius and the edge probability. In particular if $\pi r^2 = p = \frac{\log n + \gamma_n}{n}$ then both graphs are connected w.h.p. if and only if $\gamma_n \to +\infty$ and disconnected w.h.p. if and only if $\gamma_n \to -\infty$ [24,19,15,12]. We denote by p_{con} the critical edge probability to guarantee connectivity w.h.p.

2.3 Metrics

In this subsection, we present a set of performance metrics associated with random walks on graphs. These metrics are later used to validate the performance of the proposed algorithm.
The metrics of interest are:

1. *Cover Time* (CT). The Cover Time C_g of a graph G is the maximum (over all starting nodes v) expected time taken by a random walk on G to visit all nodes in G. Formally, for $v \in V$ let C_v be the expected number of steps for the simple random walk starting at v to visit all nodes in G, and the Cover Time of G is $C_g = \max_{v\in V} C_v$. The Cover Time of graphs have been widely investigated [21,2,14,13,3,25,7,20]. It has been shown by Feige in [16,17] that for simple random walks $(1+o(1))n\log n < C_g < (1+o(1))\frac{4}{27}n^3$. Results for the Cover Time of specific graphs vary from optimal cover time $\Theta(n\log n)$ associated with the complete graph, to the worse case $\Theta(n^3)$ associated with the lollipop graph. The best known cases correspond to dense, highly connected graphs; on the other hand, when connectivity decreases and bottlenecks exist in the graph, the Cover Time increases.
2. *Maximum Node Load* (NNL). We use it to measure Load Balancing at Cover Time. For every node $v \in V$ let NL_v be the expected number of visits by the random walk to node v of the undirected graph $G(V,E)$ at Cover Time. It is obvious that $NL_V \geq 1 \ \forall v \in V$ .The Maximum Node Load is the infinity norm of the expected number of visits to every node

at Cover Time. Formally: $MNL = \|NL\|_\infty = \max_{v \in V} |NL_v|$. It is a metric representing how well the visits of the random walk are distributed over the nodes of the graph. The reason we use Maximum Node Load value as a metric of Load Balancing is that the decrease of Maximum Node Load value leads to better Load Balancing. This is an important metric primarily for nodes cooperating in wireless networks due to energy consumption limitations of these nodes. Each visit to a node is associated with a fixed energy amount required to handle transmission and reception of the random walk agent. At the same time, it is evident that the Maximum Node Load metric is representative of the energy depletion in the network. It is expected that the more uniformly distributed the visits of the random walk over the nodes of the network, the more uniform is energy expenditure on behalf of each node.

3. Enhanced Random Walk with d Choice.

In this section we first introduce the $ERWC(d,h)$ algorithm. Next we show, by using an example, an advantage of the $ERWC(d,h)$ over the $RWC(d)$, and finally we give a method to estimate a good h value, which is a significant parameter that dominates the behaviour of the $ERWC(d,h)$.

3.1 Description of the ERWC(d,h).

In the Enhanced Random Walk with d Choice, $ERWC(d,h)$, the next node to move to is determined as follows: Let $h > 1$ be an integer. Let v be the node visited by the walk at time t; let $c^t(v)$ be the number of visits to node v until time t; let $N(v)$ be the set of neighbor nodes to node v.

Now we define $H^t(v)$ as follows: $H^t(v) = \left(c^t(v) * h\right) + \sum_{u \in N(v)} c^t(u)$

Algorithm 2 ERWC(d,h) Algorithm: Upon visiting node v at time t, the $ERWC(d,h)$:

1. Selects d nodes from $N(v)$ independently and uniformly at random. Let $M(v,d)$ denote the selected set of neighbors of v, $M(v,d) \subseteq N(v)$.
2. Modify $M(v,d)$ as follows:
   - $M(v,d) = \begin{cases} u, & \forall u \in M(v,d) \text{ and } c^t(u) = 0 \\ M(v,d) \text{ remains the same otherwise} \end{cases}$
3. Steps to node $u \in M(v,d)$ that minimizes $\frac{H^t(u)}{\delta(u)}$ (breaking ties in an arbitrary way)
4. $c^t(u) = c^t(u) + 1$, $H^t(u) = H^t(u) + h$, and $H^t(k) = H^t(k) + 1$, $\forall k \in N(v) - \{u\}$.

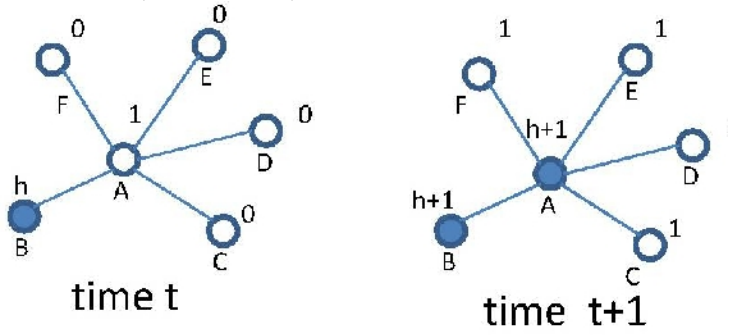

Figure 2: Execution of the $ERWC(d,h)$ algorithm on a test graph.

Figure 2 shows graphically the execution of the $ERWC(d,h)$ algorithm at two consecutive time steps t, $t+1$ n an example graph. The random walk visits node B at time t and moves to node A at time $t+1$. The respective H values for all nodes of the graph before and after the walk visits node A are given according to the $ERWC(d,h)$ algorithm. Each node visited by the random walk increases its H value by h, whereas the neighbors of that node increase their H by 1.

3.2 An advantage of the ERWC(d,h).

Now we show, by using an example, an advantage of the $ERWC(d,h)$ algorithm next movement selection over the $RWC(d)$ algorithm.

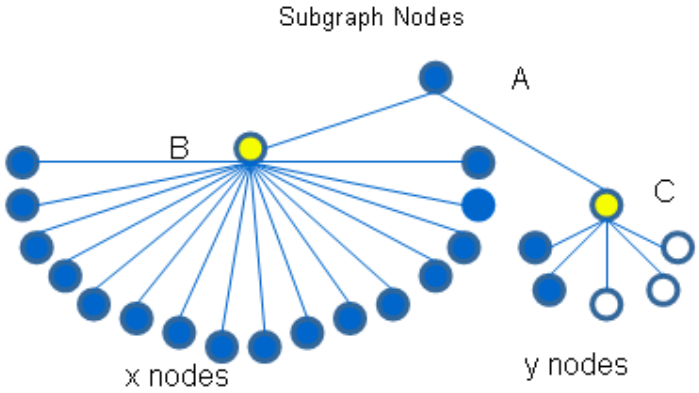

Figure 3: Subgraph of a graph.

Figure 3 shows graphically a subgraph of a graph in which all nodes with white interior color are unvisited and nodes B, C have 0 of 17 and 3 of 6 unvisited neighbors respectively. Let us assume that the random walk is at node A at time t. The yellow (shadowed) nodes B and C represent two candidate neighbors among which the selection for next movement of the random walk will be made. Let us also assume for the number of visits at nodes A, B and C that $c^t(A)=1$, $c^t(B)\geq 1$ and $c^t(C)\geq 1$. We denote by $x_i, i=1,2,...,16$ all neighbors of B except

node A and by $y_j, j=1,2,...,5$ all neighbors of B except node A. To make some calculation we assume that $h=2$, $c^t(x_i)=1,\ i=1,2,...,16$ and that only two of the y_j nodes are visited once. It is obvious that the next move should be to node C because all nodes in the neighborhood of B are visited and 3 of 6 nodes in the neighborhood of C are unvisited. Now we examine in our example, the conditions under which the $ERWC(d,h)$ steps to C while at the same time the $RWC(d)$ steps to B, in order to show that the next movement selection of the $ERWC(d,h)$ is better than that of the $RWC(d)$.

A. We assume that $c^t(B)>c^t(C)$, and $c^t(C)\geq 1$

i. RWC(d). The $RWC(d)$ ignores the information that all the neighborhood of B is visited and surely steps to B if $\frac{c^t(B)+1}{\delta(B)}<\frac{c^t(C)+1}{\delta(C)}\Leftrightarrow c^t(B)<\left(c^t(C)+1\right)\left(\frac{\delta(B)}{\delta(C)}\right)-1$

In our example we have $\frac{\delta(B)}{\delta(C)}=\frac{17}{6}=2.83$

Therefore the walk surely steps to B when $c^t(B)<\left(c^t(C)+1\right)2.83-1=2.83c^t(C)+1.83$

Now by using the above inequality we can make some calculations:

1. If we set, for example, $c^t(C)=1$ then it follows that $c^t(B)<4.66$ thus $c^t(B)\leq 4$. By assumption we have that $c^t(B)>c^t(C)\geq 1$ thus $c^t(B)>c^t(C)\geq 1$. Therefore if $2\leq c^t(B)\leq 4$ the walk surely moves to node B.
2. If we set $c^t(C)=2$ then it follows that $c^t(B)<7.49$ thus $c^t(B)\leq 7$. Therefore if $3\leq c^t(B)\leq 7$ the walk surely moves to node B.

ii. ERWC(d,h). Are there any cases in which the $ERWC(d,h)$ surely moves to C while at the same time the $RWC(d)$ steps surely to B? We showed, for example, that when $2\leq c^t(B)\leq 4$ and $c^t(C)=1$ the $RWC(d)$ steps surely to B. We have to examine the behavior of the $ERWC(d,h)$ for the same values of $c^t(B)$ and $c^t(C)$. We start first with $c^t(B)=2$ to make some calculations and after that we examine the case of $c^t(B)=3,4$. The $ERWC(d,h)$ will surely move to C if $\frac{H^t(C)}{\delta(C)}<\frac{H^t(B)}{\delta(B)}$.

In our example $\delta(C)=6$ and $\delta(B)=17$.

$$H^t(C)=(c^t(C)*h)+c^t(A)+\sum_{j=1}^{5}c^t(y_i)=5,\ H^t(B)=(c^t(B)*h)+c^t(A)+\sum_{i=1}^{16}c^t(x_i)=21$$

Thus $\frac{5}{6} < \frac{21}{17}$ and therefore the $ERWC(d,h)$ will surely move to C in this case. In our example all of the $x_i, i = 1,2,...,16$ nodes are visited once. We can have a better estimate about the minimum number of the x_i nodes that need to be visited once by the $ERWC(d,h)$ in order to surely step to C when $c^t(B) = 2$. Let denote by X the number of the x_i nodes that are visited once, $1 \le X \le 16$. We have $H^t(B) = (c^t(B) * h) + c^t(A) + \sum_{i=1}^{X} c^t(x_i) = 5 + X$

Then the $ERWC(d,h)$ surely moves to C if $\frac{5}{6} < \frac{5+X}{17} \Leftrightarrow X > 9.16$. Thus if $X \ge 10$ the $ERWC(d,h)$ will always move to C.

It is easy to check that for the other values of $c^t(B) = 3,4$ the $H^t(B)$ value increases, so the $ERWC(d,h)$ will surely move to C for lesser minimum values of X.

B. We assume that $c^t(B) \le c^t(C)$, and $c^t(B) \ge 1$. In a similar way we can prove that in this case the $RWC(d)$ walk will never step to C in contrast with the $ERWC(d,h)$ which will step to C under certain conditions.

We showed by using this example, that there are many cases in which the $ERWC(d,h)$ will move to C, while at the same time the $RWC(d)$ always moves to B, leading the $ERWC(d,h)$ to better next movement selection in comparison with the $RWC(d)$.

Our main goal in this paper is to reduce the Cover Time, so it is important to find out a good h value that minimize the $ERWC(d,h)$ Cover Time for all graph types instances with the same number of nodes.

3.3 Estimation of a good h value that minimizes the Cover Time for the ERWC(d,h).

The $ERWC(d,h)$ algorithm uses the H value at graph nodes, which highly depends on the h parameter, in order to make decisions about subsequent random walk movements.

In order to calculate $H^t(u)$ we have to calculate first a good value for h, which minimize the $ERWC(d,h)$ Cover Time for all graph types instances with the same number of nodes.

Solving such a problem requires us to find an analytical function for describing $ERWC(d,h)$ Cover Time with respect to parameter h, choice d and graph type instance. However, such an analytical function is hard to find.

Instead we propose an experimental way to find a good h value for the $ERWC(d,h)$ that minimizes the Cover Time in a specific range of h values.

Given a graph instance $G(V,E)$ and a random walk algorithm, let NS_v be the number steps needed by the random walk algorithm to cover G starting at node v, let $c(u)$ be the number of visits to node u, for each $u \in V$, when the random walk algorithm covers G.

The Cover Time is $CT = \max_{v \in V}(NS_v)$ and the Maximum Node Load is $MNL = \max_{u \in V}(c(u))$ at Cover Time.

Now we determine a simple method for finding a good h value for the $ERWC(d,h)$ algorithm which minimize Cover Time.

Let d be the choice, $n = |V|$, and N be the number of graph instances. Given a graph type and fixed d, n, N. We calculate for each $h = 2,3,....,100$ using N graph instances G_i, $i = 1,2,...,N$ the mean Cover Time $mCT_h = \sum_{i=1}^{N}(CT_i)/N$. From these mean Cover Times we find a good h value as the value resulting in lowest mean Cover Time i.e. $mCT_{hgood} = \min_{h=2,...,100}(mCT_h)$.

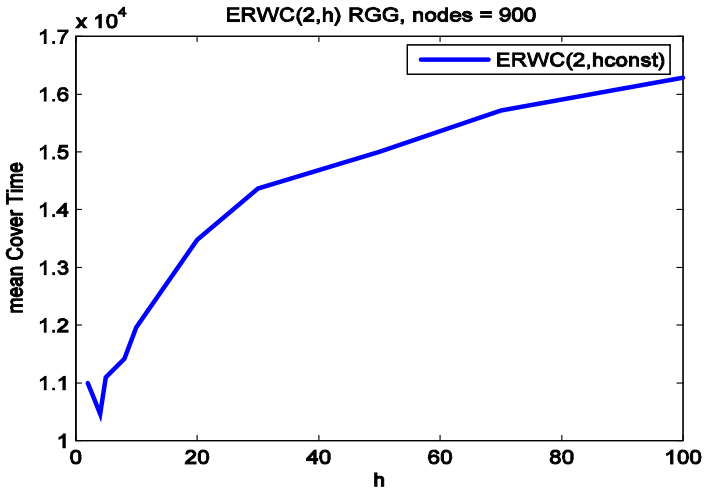

Figure 4: Mean Cover Time for different h on RGG G(900, $2r_{con}$)

Figure 4 shows an example of finding a good value for h on $N = 200$ instances of Random Geometric Graphs with $n = 900$ and $r = 2r_{con}$. The mean Cover Time of the $ERWC(d = 2,h)$ algorithm on this graph is shown for all values of $h \in [2,100]$. A good value for h is 4 that is clearly in the region of low h values because large values of h lead to possible bottlenecks and to the increase of Cover Time.

Graph Type	Number of Nodes	Good h Value
RGG, $2r_{con}$	900	4
Torus	900	3
Grid	900	2

Bernoulli, $2p_{con}$	900	7
Hypercube	256	3
Lollipop	100	3

Table 1: Results for good h that minimize the Cover Time on various types of graphs

Table 1 shows good h values calculated for all graph types and sizes used in this paper.

4 Experimental results

In this section, we present comparative results for the performance of the $ERWC(d,h)$ and $RWC(d)$ algorithms on various types of graphs (Random Geometric Graph, Torus, Grid, Hypercube, Lollipop, Bernoulli) with respect to Cover Time and Load Balancing at Cover Time.

We obtain experimental results in the following way: Let d be the choice, $n=|V|$, N be the number of graph instances and h be a previously calculated good value if the algorithm is the $ERWC(d,h)$. Given a graph type, an $ERWC(d,h)$ or $RWC(d)$ algorithm and fixed d, n, N, h then for each graph instance G_i, $i=1,...,N$ we obtain the CT_i, and MNL_i, using the same N graph instances of a graph type.

In the next step we compare these results in order to establish the performance improvement of the $ERWC(d,h)$ algorithm over the $RWC(d)$ algorithm.

To compare the Cover Time results CT_i we use the following statistical values:

$$CT_{\min}=\min_{i=1,...,N}(CT_i),\ CT_{\max}=\max_{i=1,...,N}(CT_i),\ CT_{mean}=\sum_{i=1}^{N}(CT_i)/N,$$

$$CT_{median}=\underset{i=1,...,N}{median}(CT_i),CT_{std}=\sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(CT_i-CT_{mean}\right)^2}$$

Furthermore, to compare the Maximum Node Load results we use the same statistical values.

We get simulation results for $ERWC(d,h)$ and $RWC(d)$, using the appropriate good h value for the enhanced random walk algorithm, on:

1. *Random geometric graphs,* $G(n,r)$ for $n=100,400,900,1600$, radius $r=1.1r_{con}, 2r_{con}, 3r_{con}$ and for $d=2,3$.
2. *Grid graphs* $G(n_1,n_2)$ with n nodes, where $n_1=n_2=\sqrt{n}$ and $n=n_1*n_2$ for $n=100,400,900,1600$ and for $d=2,3$.
3. Torus graphs $T(n_1,n_2)$ with n nodes, where $n_1=n_2=\sqrt{n}$ and $n=n_1*n_2$ for $n=100,400,900,1600$ and for $d=2,3$.
4. *Hypercube graphs* H_n with n nodes, for $n=256,1024$ and for $d=2,3$.

5. *Lollipop graphs* $L(n_1, n_2)$ with n nodes, where $n_1 = n_2 = n/2$ and $n = n_1 + n_2$ for $n = 100, 400, 900, 1600$ and for $d = 2$.
6. *Bernoulli graphs*, $B(n, p)$ for $n = 100, 400, 900, 1600$, radius $p = 1.1p_{con}$, $2p_{con}$, $3p_{con}$ and for $d = 2, 3$.

In this paper we present only a representative subset of the results in order to show the performance improvement of the $ERWC(d, h)$ algorithms over the $RWC(d)$ algorithms.

4.1 Experimental results for RGG

This subsection presents simulation results for $ERWC(d = 2, h = 4)$ and $RWC(d = 2)$ algorithms on Random Geometric Graphs $G(n, 2r_{con})$ with $n = 900$ and $N = 500$.

We obtain our results CT_i, MNL_i, where $i = 1, 2, ..., N$, for both algorithms using the same N instances of Random Geometric Graphs.

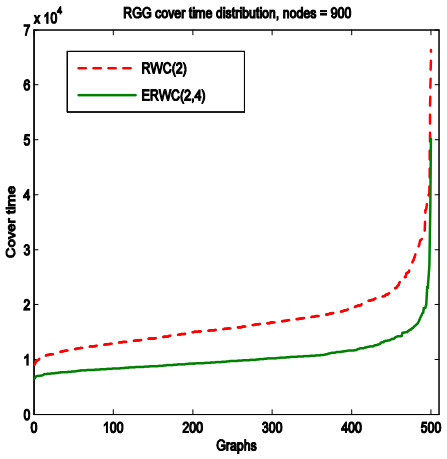

Figure 4.1: Results for the Cover Time metric for $ERWC(2, 4)$ and $RWC(2)$ on RGG

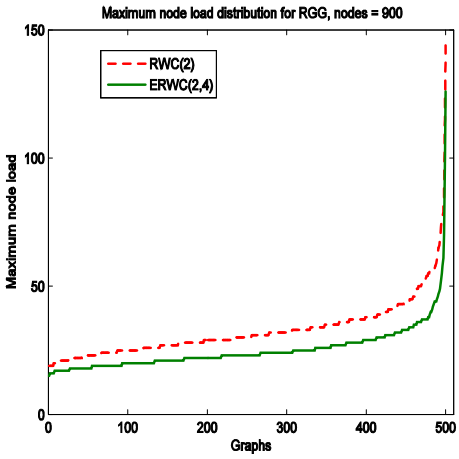

Figure 4.2: Results for the Maximum Node Load metric for $ERWC(2, 4)$ and $RWC(2)$ on RGG

Figure 4.1 is a diagram of sorted CT_i results for the Cover Time metric. It can be seen that $ERWC(2, 4)$ algorithm outperforms $RWC(2)$ throughout the whole range of graph instances used in our simulations. One can thus verify that a random walk using the $ERWC(2, 4)$ algorithm requires a lower average number of steps to fully cover the RGG.

This is an important result because it can be directly associated with reduced energy expenditures required by a wireless network to support the random walk.

Figure 4.2 shows the sorted MNL_i results for the Maximum Node Load at Cover Time for both algorithms. We can see that the $ERWC(2,4)$ algorithm is significantly better than the $RWC(2)$ algorithm with respect to Maximum Node Load. The numerical results, along with percentage reductions for both algorithms are presented in Table 2.

Statistics	$CT_{\min}$	$CT_{\max}$	CT_{mean}	CT_{median}	CT_{std}
$RWC(2)$	9036	66352	16896	15751.0	5737.5
$ERWC(2,4)$	6505	50195	10437	9711.5	3432.0
Reduction (%)	28%	24%	38%		
	$MNL_{\min}$	$MNL_{\max}$	MNL_{mean}	MNL_{median}	MNL_{std}
$RWC(2)$	19	144	32.766	30	11.566
$ERWC(2,4)$	15	126	25.138	23	8.850
Reduction (%)	21%	12.5%	23.2%		

Table 2: Statistics for CT and MNL on RGG

These results indicate a 38% reduction in required steps for Cover Time and 23.2% reduction of the MNL_{mean} value in favour of $ERWC(2,4)$ against $RWC(2)$ algorithm.

Table 2 shows also percentage improvements associated with other measures, such as the 28% reduction for $CT_{\min}$, 24% reduction for $CT_{\max}$, 21% for $MNL_{\min}$ and 12.5% for $MNL_{\max}$. These results confirm the improved performance of random walk with the $ERWC(d,h)$ algorithm over the random walk with the $RWC(d)$ on $RGGs$.

4.2 Experimental results for Torus graphs

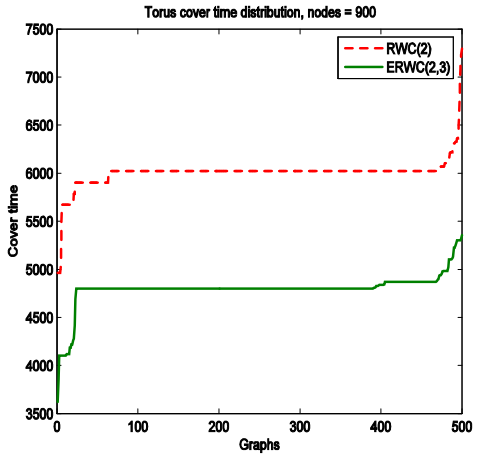

Figure 4.3: Cover Time results for $ERWC(2,3)$ and $RWC(2)$ on Torus

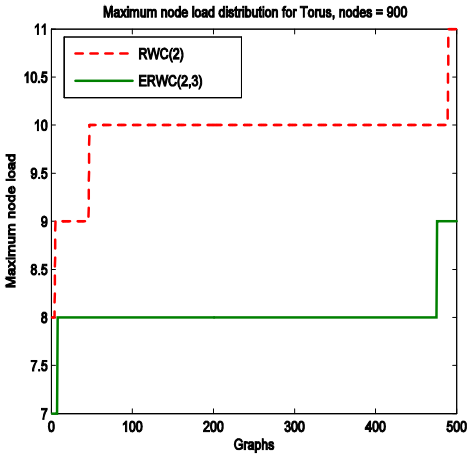

Figure 4.4: Maximum Node Load results for $ERWC(2,3)$ and $RWC(2)$ on Torus

Figure 4.3 and Figure 4.4 present the sorted CT_i, MNL_i results, where $N = 500$ and $i = 1,2,...,N$. These results were obtained for the algorithms $ERWC(2,3)$ and $RWC(2)$, using the same N instances of Torus graphs with $n = 900$. It can be seen in Figure 4.3 that the $ERWC(2,3)$ algorithm outperforms the $RWC(2)$ algorithm for all Torus graphs instances with respect to Cover Time. The flat line part of the results for Torus graphs are identical values for CT due to the symmetry of the Torus graph. There are also reductions for the Maximum Node Load metric. Numerical values of these results and the corresponding percentage reductions are shown in Table 3.

Statistics	$CT_{\min}$	$CT_{\max}$	CT_{mean}	CT_{median}	CT_{std}
$RWC(2)$	4965	7297	6008.9	6024	165.76
$ERWC(2,3)$	3621	5358	4799.4	4801	170.95
Reduction (%)	27%	26.5%	20%		
	$MNL_{\min}$	$MNL_{\max}$	MNL_{mean}	MNL_{median}	MNL_{std}
$RWC(2)$	8	11	9.92	10	0.36
$ERWC(2,3)$	7	9	8.03	8	0.25
Reduction (%)	12.5%	18%	19.6%		

Table 3: Statistics for CT and MNL on Torus Graphs.

The $ERWC(2,3)$ algorithm achieves a significant 20% average reduction in required steps to Cover Time compared with $RWC(2)$ for the Torus graph. Furthermore, the average reduction with respect to MNL metric measures is 19.6% indicating improved Load Balancing characteristics. One can thus verify performance improvements for the $ERWC(d,h)$ based random walk proposed in this paper.

4.3 Experimental results for Grids

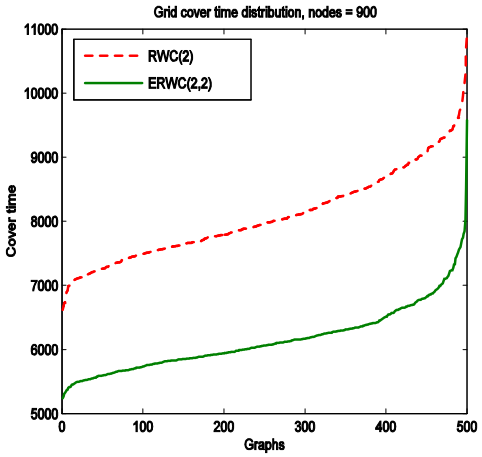

Figure 4.5: Cover Time results for $ERWC(2,2)$ and $RWC(2)$ on Grid

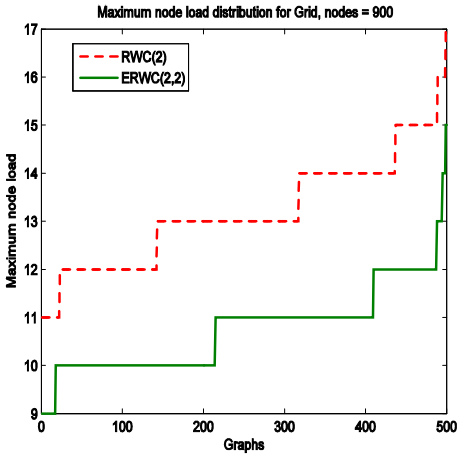

Figure 4.6: Maximum Node Load results for $ERWC(2,2)$ and $RWC(2)$ on Grid

Figure 4.5 and Figure 4.6 show the sorted simulation results CT_i, MNL_i, where $N = 500$ and $i = 1, 2, ..., N$, for the $ERWC(2,2)$ and $RWC(2)$ based random walks on Grids with $n = 900$. The results were collected for both algorithms, using the same N instances of Grid graphs. Performance improvements for the $ERWC(d,h)$ algorithm can be seen for all different instances of Grids associated with the simulations.

Statistics	$CT_{\min}$	$CT_{\max}$	CT_{mean}	CT_{median}	CT_{std}
$RWC(2)$	6619	10975	8088.7	7976.5	705.53
$ERWC(2,2)$	5239	9576	6155	6061.5	524.28
Reduction (%)	20%	12.7%	23.9%		
	$MNL_{\min}$	$MNL_{\max}$	MNL_{mean}	MNL_{median}	MNL_{std}
$RWC(2)$	11	17	13.19	13	1.13
$ERWC(2,2)$	9	15	10.76	11	0.90
Reduction (%)	18%	11.7%	18.4%		

Table 4: Statistics for CT and MNL on Grid Graphs.

Table 4 shows numerical results for both algorithms on Grids. There is a 23.9% reduction in mean required steps for Cover Time when the $ERWC(2,2)$ algorithm is used. The MNL_{mean} is also reduced by 18.4%. Furthermore, there is a 20% reduction for $CT_{\min}$, a 12.7% reduction for $CT_{\max}$, a 18% reduction for $MNL_{\min}$ and a 11.7% for $MNL_{\max}$. From these results we conclude that there is performance improvement both for the Cover Time and the Load Balancing on Grids.

4.4 Experimental results for Bernoulli graphs

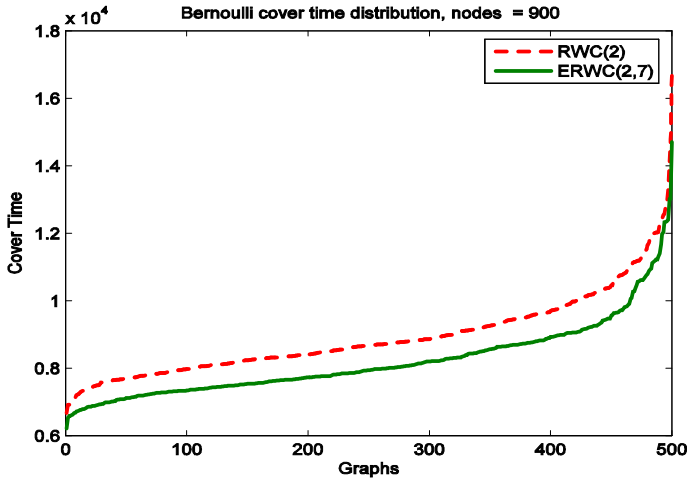

Figure 5: Cover Time distribution for $ERWC(2,7)$ and $RWC(2)$ on Bernoulli Graphs

Figure 5 shows the sorted CT_i results where $N = 500$ and $i = 1,2,...,N$ obtained by running the $ERWC(d = 2, h = 7)$ and $RWC(d = 2)$ algorithms on the same N graph instances of a Bernoulli graph $B(n, 2p_{con})$ with $n = 900$.

The results indicate that $ERWC(2,7)$ algorithm outperforms $RWC(2)$ over all Bernoulli instances used in the simulations.

Statistics	$CT_{\min}$	$CT_{\max}$	CT_{mean}	CT_{median}	CT_{std}
$RWC(2)$	6672	16896	8930.9	8668.5	1246.6
$ERWC(2,7)$	6209	14702	8210.3	7936.0	1144.1
Reduction (%)	6.93%	12.98%	8.06%		

Table 5: Statistics for CT on Bernoulli graphs.

Table 5 shows the numerical values of results for our simulations on Bernoulli graphs. There is a CT_{mean} improvement of 8.6% in favor of $ERWC(d,h)$. Further reductions in Cover Time are 6.93% for $CT_{\min}$ and 12.98% for $CT_{\max}$. The $ERWC(2,7)$ algorithm is thus shown to be an improved version of the $RWC(2)$ algorithm on Bernoulli graphs with respect to Cover Time.

The Load Balancing of $ERWC(2,7)$ is getting worse for Bernoulli graphs by comparison with $RWC(2)$. For example the MNL_{mean} get worse by 22%, so we do not present any other results about the Maximum Node Load metric.

4.5 Experimental results for Hypercube graphs

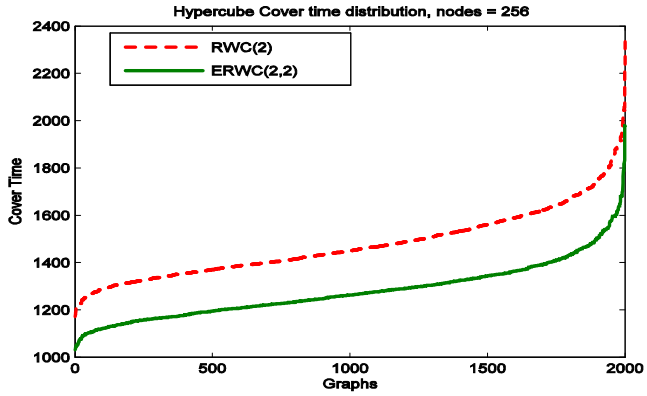

Figure 6: Cover Time distribution for $ERWC(2,2)$ and $RWC(2)$ on HyperCube Graphs

Figure 6 shows the sorted CT_i results, where $N = 2000$ and $i = 1, 2, ..., N$ obtained by running the $ERWC(d = 2, h = 2)$ and $RWC(d = 2)$ algorithms on the same N graph instances of a Hypercube graph with $n = 256$.

The results indicate that $ERWC(2,2)$ algorithm outperforms $RWC(2)$ over all Hypercube instances used in the simulations.

Statistics	$CT_{\min}$	$CT_{\max}$	CT_{mean}	CT_{median}	CT_{std}
$RWC(2)$	1171	2334	1476.0	1449	145.82
$ERWC(2,2)$	1031	1978	1279.5	1263	120.14
Reduction (%)	11.9%	15.2%	13.3%		

Table 6: Statistics for CT on Hypercube graphs.

Table 6 shows the numerical values of the results for our simulations on Hypercube graphs. There is a CT_{mean} improvement of 13.3% in favor of $ERWC(d,h)$. Further reductions in Cover Time are 11.9% for $CT_{\min}$ and 15.2% for $CT_{\max}$. The $ERWC(2,2)$ algorithm is thus shown to be an improved version of the $RWC(2)$ algorithm on Hypercube graphs with respect to Cover Time.
The Load Balancing of $ERWC(2,2)$ is getting worse for Hypercube graphs by comparison with $RWC(2)$. For example the MNL_{mean} get worse by 11%, so we do not present any other results about the Maximum Node Load metric.

4.6 Experimental results for Lollipop graphs

The last graph type used for comparing $ERWC(d = 2, h = 2)$ and $RWC(d = 2)$ algorithms is the Lollipop graph with $n = 100$.

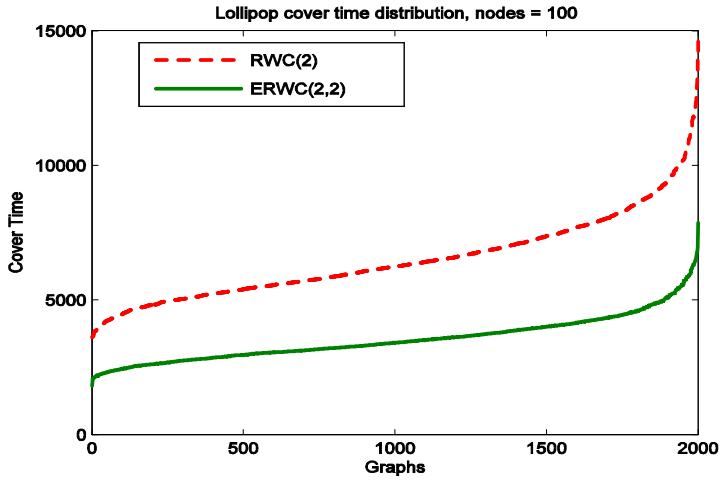

Figure 7: Cover Time results for $ERWC(2,2)$ and $RWC(2)$ on Lollipop Graphs

Figure 7 shows the sorted CT_i results, where $N = 2000$ and $i = 1, 2, ..., N$. These results where obtained for both algorithms using the same N graph instances of Lollipop graphs.

Statistics	$CT_{\min}$	$CT_{\max}$	CT_{mean}	CT_{median}	CT_{std}
$RWC(2)$	3603	14605	6510.9	6237	1557.90
$ERWC(2,2)$	1813	7868	3544.7	3410	827.02
Reduction (%)	49.6%	46.1%	45.5%		

Table 7: Statistics for CT on Lollipop graphs.

The reduction in CT_{mean} in favor of $ERWC(2,2)$ measures 45.5%. Furthermore, Table 7 shows reductions of 46.9% at $CT_{\min}$ and 46.1% at $CT_{\max}$.

These results show a significant performance improvement in Cover Time of $ERWC(2,2)$ algorithm over the $RWC(2)$ algorithm on Lollipop graphs.
The performance improvement in Load Balancing on Lollipop graphs is small. For example there is a 2.99% improvement with respect to MNL_{mean}, so we do not present any other results about the Maximum Node Load metric.

5 Conclusions

In this work, the Enhanced Random Walk with Choice algorithm is introduced as an improved version of the Random Walk with Choice algorithm. The $ERWC(d,h)$ algorithm is formally defined and compared against $RWC(d)$ via simulations over six different graph types, including Random Geometric Graphs, Torus, Grids, Hypercube, Lollipop, and Bernoulli graphs. Comparative results are presented for both $ERWC(d,h)$ and $RWC(d)$ algorithms, which are based on the Cover Time metric and Maximum Node Load metric. Our simulation results indicate significant savings in Cover Time up to 38% for Random Geometric Graphs, 20% for Torus

graphs, 23.9% for Grids, 13.3% for Hypercube, 45.5% for Lollipop and 8.06% for Bernoulli graphs. Furthermore, the Load Balancing properties of the $ERWC(d,h)$ algorithm are better than those of $RWC(d)$ with reductions for MNL in the order of 23.2% for Random Geometric Graphs, 19.6% for Torus graphs and 18.4% for Grids.

Our plans for future work include running simulations on other types of graphs to verify performance improvement of the $ERWC(d,h)$ algorithm over $RWC(d)$. Furthermore, the analytical description of Cover Time for both algorithms is an open issue for future research.

ACKNOWLEDGEMENTS

The second author gratefully acknowledges support of the project Autonomic Network Architecture (ANA), under contract number IST-27489, which is funded by the IST FET Program of the European Commision.

Part of this work by Dr. Gregory Karagiorgos was carried out while he was in the Department of Informatics and Telecommunications, University of Athens

REFERENCES

[1] M. Alanyali, V. Saligrama, and O. Sava, A random-walk model for distributed computation in energy-limited network. In Proc. of 1st Workshop on Information Theory and its Application, San Diego, 2006.

[2] D.J. Aldous, Lower bounds for covering times for reversible Markov chains and random walks on graphs. J.Theoret. Probab., 2(1): 91-100, 1989

[3] R. Aleliunas, R. M. Karp, R. J. Lipton, L. Lovász, and C. Rackoff, Random walks, universal traversal sequences, and the complexity of maze problems. In 20th Annual Symposium on Foundations of Computer Science (San Juan, Puerto Rico, 1979), pages 218-223. IEEE, New York, 1979.

[4] John Alexandris and Gregory Karagiorgos, Enhanced Random Walk with Choice: An Empirical Study. In the 2012 International Conference of Wireless Networks (ICWN'12) (London), pages 1263-1268. Vol II Newswood Limited IAENG, ISBN: 978-988-19251-3-8.

[5] John Alexandris, Gregory Karagiorgos and Ioannis Stavrakakis, Enhanced Random Walk with Choice: An Empirical Study. In arxiv.org 1007.3157v1 [cs.DS].

[6] Chen Avin and Bhaskar Krishnamachari, The Power of Choice in Random Walks: An Empirical Study, The 9th ACM/IEEE International Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems, (MSWiM), Malaga, Spain, October 2006.

[7] Chen Avin and Gunes Ercal, On The Cover Time of Random Geometric Graphs. In Proceedings, Automata, Languages and Programming, 32nd International Colloquium, ICALP-05, pages 677-689, 2005.

[8] Noga Alon, Chen Avin, Mchail Koucky, Gady Kozma, Zvi Lotker and Mark R. Tuttle (2011). Many Random Walks Are Faster Than One. Combinatorics, Probability and Computing, 20, pp 481-502 doi:10.1017/S0963548311000125

[9] Chen Avin and C. Brito, Efficient and robust query processing in dynamic environments using random walk techniques. In Proc. of the third international symposium on Information processing in sensor networks, pages 277-286, 2004.

[10] Chen Avin (2008). Distance Graphs: From Random Geometric Graphs to Bernoulli Graphs and Between. DIALM-POMC '08 Proceedings of the fifth international workshop on Foundations of mobile computing, ACM New York, NY, USA doi:10.1145/1400863.1400878

[11] Y. Azar, A. Broder, A. Karlin, and E. Upfal, Balanced allocations. In Proceedings of the 26th ACM Symposium on the Theory of Computing, pages 593-602, 1994.

[12] B. Bollobás: Random Graphs. Academic Press, Orlando, FL (1985)

[13] A. Broder and A. Karlin, Bounds on the cover time. J. Theoret. Probab., 2: 101-120, 1989.

[14] A. K. Chandra, P. Raghavan, W. L. Ruzzo, and R. Smolensky, The electrical resistance of a graph captures its commute and cover times. In Proc. of the twenty-first annual ACM symposium on Theory of computing, pages 574-586. ACM Press, 1989.
[15] P. Erdős, A. Rényi: On random graphs. Publicationes Mathemticae (Debrecen) 6 (1959) 290–297
[16] U. Feige, A Tight Upper Bound on the Cover Time for Random Walks on Graphs. Random Struct. Alg. 6 (1995), 51-54.
[17] U. Feige, A Tight Lower Bound on the Cover Time for Random Walks on Graphs, Random Struct. Alg. 6 (1995), 433-438.
[18] E.N. Gilbert: Random graphs. The Annals of Mathematical Statistics 30 (1959) 1141–1144
[19] P. Gupta and P.R. Kumar, Critical power for asymptotic connectivity in wireless networks. In Stochastic Analysis, Control, Optimization and Applications: A Volume in Honor of W.H.Fleming, 1998, 547-566.
[20] Johan Jonasson and Oded Schramm, On the Cover Time of Planar Graphs in Electronic Communications in Probability 5 (2000) 85-90.
[21] P. Matthews, Covering problems for Brownian motion on spheres. Ann. Probab., 16(1): 189-199, 1988.
[22] Michael Mitzenmacher, The Power of Two Choices in Randomized Load Balancing, PhD Thesis, 1996.
[23] R. Motwani and P. Raghavan, Randomized algorithms, Cambridge University Press, 1995.
[24] M.D. Penrose: The longest edge of the random minimal spanning tree. The Annals of Applied Probability 7 (1997) 340–361.
[25] D. Zuckerman, A technique for lower bounding the cover time, In Proc. of the twenty-second annual ACM symposium on Theory of computing, pages 254-259. ACM Press, 1990.

Authors

John Alexandris received the B.Sc. degree in Physics from the National and Kapodistrian University of Athens, Greece, in 1990. He is working as a software engineer. His research interests are: Design and Analysis of Algorithms, Random Graphs and Random Walks.

Gregory Karagiorgos received the B.Sc. (Maitrise) degree in Micro-Informatique Appliquee Aux Sciences Humaines from Universite de Paris VIII-SAINT-DENIS, France, in 1986. He received the Ph.D. degree in Computer Science from the National and Kapodistrian University of Athens, Greece, in 2002. He is an assistant professor in the Department of Computer Engineering at Technological Institute of Peloponnese since March 2010. His research interests are: Design and Analysis of Algorithms, Random Walks and Combinatorial Optimization.