

# ARC Fellowship Application

Atish Das Sarma

Advisor - Richard J. Lipton

## Walk Fast Distributively and Learn despite Byzantine Failures

**Past Work.** We consider the problem of performing random walks efficiently in a distributed network; this has various applications including node sampling, malicious behavior detection, search etc. The model in distributed computing is that each node only has local access (to its neighbors) and in every round of communication, two adjacent nodes can communicate along the edge. The number of bits communicated per edge per round is restricted to  $O(\log n)$ . In all previous systems and applications, random walks of length  $l$  have always been done naïvely in  $O(l)$  rounds. In a paper to appear in PODC 2009, along with D. Nanongkai and G. Pandurangan, we showed a surprising result that one can perform such walks in just  $\tilde{O}(l^{2/3}D^{1/3})$  rounds, where  $D$  is the diameter of the graph (on undirected unweighted graphs). Even more recently, we worked together with P. Tetali and improved these results further. We show how to perform walks of length  $l$  in  $\tilde{O}(\sqrt{lD})$  rounds and complement it with a lower bound showing that  $\Omega(\frac{\sqrt{l}}{\log l})$  rounds are required. Thus, our results are almost tight for small diameter networks; this work is under submission.

**Criticism.** Due to the widespread applicability of random walks in distributed networks, our theoretical results can be very helpful in practice. However, in the current state, our algorithm has a few undesirable aspects from the practical standpoint. For example, one main complaint of the algorithm is, *all* nodes need to perform short local random walks even though eventually a walk is required from only a specific source node. While this does not violate the distributed computing model (since it allows certain communication along *each* edge in each round), this is undesirable as for a process as simple as sampling one destination, the entire network has to trigger computations. Eventually, the algorithm *stitches* few of these short random walks to form the walk of length  $l$  starting at the source. Most of the short walks performed go unused. This extra computation can be measured as Message Complexity, the number of messages exchanged throughout the system. What would be nice is for only few *relevant* nodes to initiate short walk computations, or, for a distributed mechanism for *maintaining* short walks from all nodes that can be used repeatedly. This way, for any single walk  $l$ -length request, only a small computation is required, and yet, we get our theoretically provable near-optimal  $\tilde{O}(\sqrt{l})$  round complexity. Another shortcoming of the results we prove (not necessarily of our techniques though) from a practical standpoint is that we do not consider failures. Network failures (both node and edges) occur all the time and it is very important for any distributed algorithm to be resistant to such network perturbations.

**Future Work.** One of the main extensions to this work that I am interested in is taking our theoretical techniques further, and adapting them as necessary, to actually make our improved bounds useful in practice. This would require improving our algorithms in several dimensions - (1) minimizing the computation for every walk request, (2) increasing resistance to node/edge failures, (3) look at specific applications and check for throughput bottle-necks other than the random walk component, and (4) optimize with more concern for even  $\log n$  and diameter  $D$  terms that are conveniently hidden currently in our theorems. I am also interested in theoretically extending our techniques to perform  $K$  walks, for large  $K$ , more efficiently. Notice that our upper bound is near-optimal only for 1 walk as our lower bound is also for performing a single walk. We have some work in this direction. Another theoretical problem that I am interested in is using our efficient

random walk algorithm as a subroutine to estimate eigenvalue gaps (second eigenvalue, and mixing time), connectivity structure (second eigenvector) of the transition matrix corresponding to the network graph. Below I describe few specific problems I plan to target during the coming semester.

- **Distributed maintenance of short walks,  $K$  walks, estimating spectral gap:** Can one store some pre-computed information with each node (such as the short walks in our algorithm) such that, at run time, whenever a walk of length  $l$  is desired from a source node  $s$ , the node sample can be obtained in  $\tilde{O}(\sqrt{l})$  rounds of communication and the total message complexity is also  $\tilde{O}(l)$ . This would then be near-optimal in message complexity and number of rounds and also facilitate sampling any number,  $K$ , of walks fast. A second step would be to prove that these  $K$  walks can be computed in parallel in less than  $K\sqrt{l}$  rounds; I believe the correct bound here is  $\tilde{O}(\sqrt{Kl})$  rounds (since one can think of  $K$  walks of length  $l$ , perhaps, as one walk of length  $Kl$ ). I would like to prove near-tight upper and lower bounds on round complexity and message complexity for performing  $K$  walks.

With a subroutine for efficiently sampling from several random walks, this naturally leads to a technique for estimating the spectral gap, or the mixing time of the network graph in a distributed manner. Samples from walks of length  $l$  are samples from the distribution induced at length  $l$ . These algorithms can be extended to sample from walks of length  $1, 2, 4, 8, \dots$ . One can then find an efficient way to compare the distance between distributions and estimate  $t$  such that the distribution at  $t$  and  $2t$  are *close*. This gives an estimate of the mixing time and thereby the spectral gap. Similar random walk techniques have been used for estimating the second eigenvector of the transition matrix; this gives important structural information on the topology of the network (such as sparse cuts etc.). It would be interesting to extend our distributed techniques to estimating these structural properties of the graph.

- **Approximate Sampling in presence of Byzantine failures:** In distributed networks such as peer to peer networks, node failures (network site going down), and edge failures (communication path failing) occur continuously. A natural way to model this theoretically is to assume that each communication edge (and/or node) fails independently and randomly with probability  $p$  during any round; this is usually called Byzantine failures. Notice that this could bias the random walk. I have two approaches in mind to extend our results and techniques to handle failures. First is to perhaps *wait* a bit at each step to make sure, with high probability, that the random walk step being taken is correct and prove that the number of rounds requires is still not too much. The second is to perhaps execute the algorithm as it is, and prove that the samples obtained are not far from sampling from the true distribution. In either case, some new ideas are required to extend these results. Extending random walk techniques to handle byzantine failures will greatly increase the applicability to real distributed systems.
- **Directed Graphs [Probably harder so will work on it after above two]:** I am interested in obtaining efficient algorithms for performing random walks and estimating the top eigenvector of directed network graphs. Directed network graphs arise in practice when there are one-way communication channels. The PageRank vector (top eigenvector) provides important information on the relative importance of nodes in the network. For undirected graphs, this is defined by the degree distribution, but there is no easy way of approximating this for directed graphs. The difficulty in approximating the pagerank vector distributively arises for (at least) two reasons: directed edges can cause far more congestion in the network than undirected edges, as a lot of random walks may be forced to go through a single edge. Second, mixing times can be exponentially large in the network. The first step would be obtaining an algorithm that can perform a single random walk sample from the distribution at length  $l$  in  $o(l)$  rounds.