

# Research Statement

Purnamrita Sarkar

January 15, 2009

## 1 Overview

A broad range of graph-based real world applications, collaborative filtering in recommender networks, link prediction (e.g. predicting future links from the current snapshot of a graph) in social networks, fraud detection, personalized graph search techniques and graph visualization over time requires a deep understanding of the underlying graph structure. These networks can consist of millions of entities, and hence scalability is a critical concern for designing algorithms in these settings.

My research focus is on understanding and analyzing the dynamics of entity relationships in large networks. This consists of two parts. The first involves answering questions about a given graph: which graph-theoretic measure is best suited for a given learning task? How can we compute these measures efficiently on massive networks? What are the properties of the graph that we can exploit for designing efficient algorithms? Since a lot of existing algorithms use random walks in one form or the other, I have worked on accelerating random walk based approaches, in particular for computing proximity measures in graphs. Currently I am working towards developing new algorithms with formally proven properties for ranking problems on graphs of up to a billion nodes on a single CPU machine. The second question is how to model the friendships drifting over time? I have also worked on tractable generative models which capture the evolution of large social networks.

## 2 Proximity search in graphs

Link prediction in social networks, personalized graph search techniques, fraud detection and collaborative filtering in recommender networks are important practical problems that greatly rely on graph theoretic measures of similarity. Given a node in a graph we would like to ask which other nodes are most similar to this node. Ideally we would like this similarity measure to capture the graph structure such as having many common neighbors or having several short paths between two nodes. This kind of structural information can be easily quantified using random walks on graphs: diffusion of information from one node to another.

Random walks on graphs is an extremely well studied domain of mathematics. However, most random walk based proximity measures are computationally impractical for large graphs. For example computing expected commute time [1] between pairs of nodes in a graph becomes intractable when the underlying graph is very large. These proximity measures are also handicapped by their dependence on long-range paths. I have devised algorithms [11] which utilize the short term behavior of random walks on graphs and quickly compute all *interesting* pairs of approximate nearest neighbors in graphs without examining all pairs. These are evaluated on both simulated and real graphs of size up to 100,000 entities on a single CPU machine, which was previously impossible. My aim is to be able scale up to billion node graphs on a single CPU machine. The focus of my thesis is design and analysis of algorithms for proximity search in graphs, which is a core primitive at the heart of many graph-based learning problems, and I will discuss it in detail in the following sections.

### 2.1 Random Walks on Graphs

The **hitting time** from node  $i$  to node  $j$  is the expected number of hops to hit node  $j$  in a random walk starting at node  $i$ . The **commute time** is the expected time to hit node  $j$  and come back to node  $i$ . These measures give a natural definition of distance in a graph. They are robust to noise,

which (for undirected graphs) is a direct consequence of the analogy between these measures and electrical properties of graphs as shown by [4].

For *undirected graphs* the hitting and commute times between pairs of nodes can be computed in closed form using the pseudo-inverse of the graph laplacian  $L$ . Most clustering applications either require extensive computation to produce pairwise proximity measures, or employ heuristics for restricting the computation to subgraphs. See, for example [9], and [2]. In short, it is only tractable to compute these measures on graphs with a few thousand nodes for most purposes. Unfortunately these techniques are *not applicable to directed graphs*, which is a severe restriction since the underlying graphs of many real world problems are directed in nature. In contrast, our approach does not require the underlying graph to be undirected.

The proximity measures described above are impaired by their sensitivity to long range paths [8, 2]. In addition, popular entities often obtain high rankings under these metrics, making them relatively ineffective for personalized search problems [2]. We therefore propose truncated versions of these measures in order to exploit the short term behavior of random walks.

## 2.2 Finding Closest Commute Time Neighbors in Large Graphs

We develop an algorithm [11] to compute  $\epsilon$ -approximate nearest neighbors in truncated hitting time in large graphs. The main idea is to break the graph into overlapping neighborhoods for each node. The neighborhood for each node is initialized with its direct neighbors and is expanded so as to prune away nodes which are *provably* not potential nearest neighbors. The algorithmic framework and provides upper and lower bounds on hitting times from the nodes in the neighborhood to the starting node. These bounds get tighter with the expansion of the neighborhood. After caching the information for each node, one can quickly answer approximate nearest neighbor queries in truncated commute time. The evaluation was based on link prediction tasks on 100,000 node real graphs obtained from Citeseer on a single CPU machine. Our algorithm performed consistently better than other existing algorithms in the social networks/link prediction domain. Empirical results and preliminary theoretical analysis indicate near-linear runtime.

## 2.3 Fast Incremental Proximity Search in Large Graphs

In this project [12] we investigate two aspects of ranking problems on large graphs. First, we augment the deterministic pruning algorithm mentioned in section 2.2 with sampling techniques to compute approximately correct rankings with high probability under random walk based proximity measures *at query time*. Second, we prove some surprising locality properties of these proximity measures by examining the short term behavior of random walks. The proposed algorithm can answer queries *on the fly without caching any information about the entire graph*. We present empirical results on a 600,000 node author-word-citation graph from the Citeseer domain on a single CPU machine where the average query processing time is around 4 seconds. We present quantifiable link prediction tasks. On most of them our techniques outperform Personalized Pagerank, a well-known diffusion based proximity measure.

## 2.4 Fast Dynamic Reranking in Large Graphs

So far we have considered ranking problems. In this project (under submission) we design algorithms for quickly *reranking* search results based on a small set of results labeled as relevant or irrelevant by the user. Note that this is slightly different from traditional semi-supervised learning problems since the labeled set of nodes vary from user to user and as a result we want an algorithm which will be extremely fast in order to be able to incorporate user-feedback on the fly.

We present a graph theoretic measure for discriminating irrelevant results from relevant results using a few labeled examples provided by the user. The key intuition is that nodes relatively closer (in graph topology) to the relevant nodes than the irrelevant nodes are more likely to be relevant. We present a simple sampling algorithm to evaluate this measure at specific nodes of interest, and an efficient branch and bound algorithm to compute the top k nodes from the entire graph under this measure. On quantifiable prediction tasks the introduced measure outperforms other diffusion-based proximity measures which take only the positive relevance feedback into account. On the entity-relation graph built from the authors and papers of the entire DBLP citation corpus (1.4 million

nodes and 2.2 million edges) our branch and bound algorithm takes about 1.5 seconds to retrieve the top 10 nodes w.r.t. this measure with 10 labeled nodes.

### 3 Probabilistic Modeling of Dynamic Networks

I am also interested in designing tractable generative models for large dynamic networks [10]. I have explored two facets of social network modeling. First, we generalized a successful static model of relationships into a dynamic model that accounts for friendships drifting over time. Second, we showed how to make it tractable to learn such models from data, as the number of entities  $n$  gets large. The generalized model associates each entity with a point in  $p$ -dimensional Euclidean latent space. The points can move as time progresses but large moves in latent space are improbable. Observed links between entities are more likely if the entities are close in latent space. We showed how to make such a model tractable (sub-quadratic in the number of entities) such that amortized time per entity is only  $O(\log n)$ . Our algorithm scaled easily to a real world graph of 11,000 nodes, which is two orders of magnitude larger than the graphs that could be analyzed by prior statistical network modeling algorithms.

### 4 Ongoing and Future Work

The last few sections open us to a rather challenging and interesting array of questions which involve machine learning with very large graphs. For example: how can we let end-users with machines having weak memory and processing power use our algorithms? How does one identify the right measure for a given learning task? In settings where human intervention is needed to make an informed decision, how can we minimize it as much as possible? I believe that answering these questions will enable us to efficiently utilize the massive source of networked data that is being generated from corporate and public sources every day. We can answer these questions by unifying our knowledge in diverse areas like graph theory, data mining, information theory and database systems. I will address each of them separately.

#### 4.1 Scalable Algorithms on Graphs: Where is the limit?

The existing algorithms for query processing on massive graphs assume that the graph can be loaded into main memory. What if I want to process an arbitrary query on a graph too large to fit into main memory? Note that this gives rise to two constraints: (a) the query is arbitrary, and (b) the entire graph cannot be loaded into main memory. Existing algorithms relax one or both of these constraints. For example streaming algorithms [13] can process graphs too big for memory, but they require a few passes of the data in order to process an arbitrary query. On the other hand algorithms like [6] can answer an arbitrary query by caching sketches for each node in a preprocessing step, but the assumption is that the graph can be fit into main memory. I believe that one of the important future directions is to characterize the tradeoffs and limitations of algorithms in this setting.

One may counter that the out-of-memory constraint is not realistic: after all, clever graph compression techniques could be applied on the large graphs. However there are several important settings where this might not work: the graph could be simply be too large, as is the case with the Web graph. Decompression might lead to an unacceptable increase in query response time. Also consider the problem of searching personal information networks [3], which requires integrating the users personal information with information from the Web, often needs to be performed on the user's own machine for preserving privacy [5]. These machines are general purpose machines, with many other applications vying for memory space. Hence one must try to use as little memory as possible for a search application which is only used occasionally.

The first step towards this goal will be to design an algorithm that allows one to simulate a random walk on an external memory graph while minimizing I/O costs. This is of tremendous importance since most random walk based metrics (e.g. personalized pagerank, hitting and commute times, simrank [7]) can be estimated by sampling paths in a graph. In our ongoing work we are developing an algorithm which will organize the graph on disk such that one can simulate a random walk without requiring too many disk seeks. The key intuition, similar to some recent work in the theoretical community [14] is that if one is able to cluster a graph such that the clusters have small conductance then a random

walk is more prone to stay within one cluster. However, how does the number of seeks depend on the number of cross-edges, the number of nodes in each page, the size of the graph, and other properties of the graph? We want to identify the formal relationship between these elements and use that to partition large graphs into page-sized chunks, with corresponding guarantees on runtime performance.

## 4.2 Graph-based Learning Tasks: Automatically Identifying the Right Measure

A lot of research has been directed towards new metrics and clever approximation algorithms. However the goodness of an algorithm is often identified with the quality of the approximation. This brings us to a very fundamental question: what is the best measure for a given task on a given class of graphs? The answer to this would have a tremendous impact on our intuition about real world graphs, and would enable automatic identification of the right measure for a task at hand on a given graph. It would also save end-users the trouble to understand and compute every metric separately for a task.

## 4.3 Maximizing the Utility of User-feedback

Personalized search being the holy grail, algorithms for reranking and ranking refinement which incorporate user-feedback are gaining popularity. Under this setting the user is asked to label a set of results. Ideally we would want to exploit this feedback loop to the utmost. This brings us to graph-based active learning problems. We want to find a small subset of nodes, such that knowing their labels will maximize the overall labeling accuracy. While there is a rich literature on active learning, the traditional techniques are not aimed at graph data-sets. Since in a graph, the relationship between the datapoints, i.e. the nodes are known explicitly, there is significant scope for improvement by exploiting the graph structure.

## References

- [1] David Aldous and James Allen Fill. *Reversible Markov Chains*. 2001.
- [2] M. Brand. A Random Walks Perspective on Maximizing Satisfaction and Profit. In *SIAM '05*, 2005.
- [3] Soumen Chakrabarti, Jeetendra Mirchandani, and Arnab Nandi. Spin: searching personal information networks. In *SIGIR '05: Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 674–674, New York, NY, USA, 2005. ACM.
- [4] A. K. Chandra, P. Raghavan, W. L. Ruzzo, and R. Smolensky. The electrical resistance of a graph captures its commute and cover times. In *STOC'89*, pages 574–586, 1989.
- [5] Bhavana Bharat Dalvi, Meghana Kshirsagar, and S. Sudarshan. Keyword search on external memory data graphs. *Proc. VLDB Endow.*, 1(1):1189–1204, 2008.
- [6] D. Fogaras, B. Rcz, K. Csalogny, and Tams Sarls. Towards scaling fully personalized pagerank: Algorithms, lower bounds, and experiments. 2004.
- [7] Glen Jeh and Jennifer Widom. Simrank: A measure of structural-context similarity. In *ACM SIGKDD, Proceeding of the International Conference on Knowledge Discovery and Data Mining*, 2002.
- [8] David Liben-Nowell and Jon Kleinberg. The link prediction problem for social networks. In *CIKM '03*, 2003.
- [9] M. Saerens, F. Fouss, L. Yen, and P. Dupont. The principal component analysis of a graph and its relationships to spectral clustering, 2004.
- [10] Purnamrita Sarkar and Andrew Moore. Dynamic social network analysis using latent space models. *SIGKDD Explorations: Special Edition on Link Mining*, 2005.
- [11] Purnamrita Sarkar and Andrew Moore. A tractable approach to finding closest truncated-commute-time neighbors in large graphs. In *Proc. UAI*, 2007.
- [12] Purnamrita Sarkar, Andrew W. Moore, and Amit Prakash. Fast incremental proximity search in large graphs. In *Proceedings of the 25th International Conference on Machine Learning*, 2008.
- [13] Atish Das Sarma, Sreenivas Gollapudi, and Rina Panigrahy. Estimating pagerank on graph streams. *PODS*, 2008.
- [14] D. Spielman and S. Teng. Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems. In *Proceedings of the STOC'04*, 2004.