
Random Walks with Random Projections

Purnamrita Sarkar

Machine Learning Department
Carnegie Mellon University
Pittsburgh, PA 15213
psarkar@cs.cmu.edu

Geoffrey J. Gordon

Machine Learning Department
Carnegie Mellon University
Pittsburgh, PA 15213
ggordon@cs.cmu.edu

Abstract

Random projections have been widely used for dimensionality reduction of high dimensional problems. In this paper we show how to compute some popular random walk based proximity measures (hitting and commute times, personalized pagerank) using random projections in undirected graphs. A number of important graph-based real world applications such as image segmentation, collaborative filtering in recommender networks, link prediction in social networks, fraud detection, and personalized graph search techniques rely on graph-theoretic measures of similarity. Random walk based proximity measures have been widely used for these applications. Hence we believe that the algorithm presented in this paper can benefit all these applications. First we show that discounted hitting and commute times are related to personalized pagerank. Next we generalize a recent result from Srivastava et. al. to computing personalized pagerank values by using random projections. We show that our formulation is equivalent to defining the position of a node in terms of its personalized pagerank from a set of basis nodes chosen randomly from the graph. Our preliminary experimental analysis opens up some very interesting questions, which we conclude with.

1 Introduction

Personalized page-rank, hitting and commute time between nodes are popular graph-based proximity measures which exploit structural properties of the graph. We will show that discounted hitting and commute time can be derived from personalized pagerank values. While personalized pagerank is effective at providing personalized recommendations, it is computationally expensive: to find the ranks of all nodes under a single starting distribution, we need to solve an $n \times n$ system of equations, where n is the number of nodes in the graph. We propose an algorithm which solves for a small number of linear systems ahead of time, and then uses these cached results to compute approximate personalized pagerank vectors for arbitrary starting distributions in real time. We are not the only ones to achieve the same goal. In previous work full personalization was achieved by [4], and [10]. The former simultaneously pre-computes personalized pagerank vectors via sampling from each node and stores them. The later improves the error-bounds of the sampling algorithm significantly by computing and storing sparse representations of personalization vectors from every node. Sarlós et al. proposed a deterministic algorithm for computing the personalized pagerank to every node by using rounding and sketching schemes. We intend to compare our algorithm with Sarlós et al. in future work.

Srivastava et al. showed [11] how the effective resistance between any pair of points in an undirected graph can be computed using random projections. Now we will show how to compute personalized pagerank values using random projections which will require only $O(\log n/\epsilon^2)$ dimensions per node in a n node graph, for accuracy ϵ . We show that this representation can also be interpreted as representing each node's location as a linear combination of personalized pagerank values from randomly picked nodes in the graph, with random coefficients. We demonstrate our results on random geometric graphs, and a co-authorship network.

2 Proximity Measures

We will first introduce some notation. Let A be the $n \times n$ adjacency matrix for an undirected graph $G = \{V, E\}$, such that the i, j^{th} entry denotes the weight on an edge between nodes i and j . For undirected graphs A is symmetric. For this paper we will only focus on undirected graphs. D is a diagonal matrix with weighted degrees of each node along its diagonal. L is the Laplacian matrix of the graph, and is defined as $D - A$. The graph Laplacian often comes in handy for connecting various undirected graphs with electrical networks. L is positive semi-definite. This can be seen from the fact that for any vector x , we can write $x^T L x$ as $\sum_{i,j \in E} w_{ij} (x_i - x_j)^2$, where w_{ij} is the weight of the edge between nodes i and j . P is the probability transition matrix which is obtained by row normalizing A , and the i, j^{th} entry denotes the probability of going to node j from node i in one step. We will define the Laplacian corresponding to the α discounted random walk as: $L_\alpha = D - (1 - \alpha)A$.

Personalized Pagerank: Personalized pagerank from a node i is defined as the stationary distribution corresponding to a stochastic process where a random surfer starts from node i , and at any time-step restarts the random walk with probability α and with probability $1 - \alpha$ moves to a neighboring node uniformly at random. Mathematically this is equivalent to the following equation, where r denotes the restart distribution, in this case, it is the indicator vector for i . We will denote personalized pagerank by v . For personalized pagerank at node j , w.r.t node i we use $v_i(j)$.

$$\begin{aligned} v &= \alpha r + (1 - \alpha)P^T v \\ v &= \alpha(I - (1 - \alpha)P^T)^{-1} r \end{aligned} \quad (1)$$

Intuitively personalized pagerank from node i to node j is simply a geometric sum over the occupancy probabilities, i.e. $\alpha \sum_{t=0}^{\infty} (1 - \alpha)^t P^t(i, j)$. For undirected graphs by using the reversibility of random walks we have: $v_i(j)/d(j) = v_j(i)/d(i)$. Also note that

$$\begin{aligned} v_i &= \alpha(I - (1 - \alpha)P^T)^{-1} e_i \\ &= \alpha(I - (1 - \alpha)AD^{-1})^{-1} e_i \\ D^{-1} v_i &= \alpha(D - (1 - \alpha)A)^{-1} \\ \frac{v_i(j)}{d(j)} &= \alpha L_\alpha^{-1}(i, j) \end{aligned} \quad (2)$$

α discounted hitting time $h_\alpha(i, j)$: The α discounted hitting time from node i to node j is defined as the expected time to hit node j from node i in a random walk which can stop with probability α at any step after it started. If a random walk never hits node j , then the α -discounted hitting time is the sum of the infinite series $\sum_{t=0}^{\infty} (1 - \alpha)^t = 1/\alpha$. Hitting time to oneself is zero. The α -discounted commute time is just $h_\alpha(i, j) + h_\alpha(j, i)$. We introduce the following result in this paper.

Theorem 2.1 *The α -discounted hitting time $h_\alpha(i, j)$ is related to personalized pagerank by:*

$$h_\alpha(i, j) = \frac{1}{\alpha} \left[1 - \frac{v_i(j)}{v_j(j)} \right]$$

Proof: Let h be hitting time to node j . We will ignore the subscript of α and j for this proof, since we are computing α discounted hitting time from all nodes i to j . We will denote the hitting time from node i to j by $h(i)$, which can be written as the average of the hitting times of its neighbors to j .

$$h(i) = \begin{cases} 1 + (1 - \alpha) \sum_k P(i, k) h(k) & \text{when } i \neq j \\ 0 & \text{otherwise} \end{cases}$$

We will use a trick similar to [8] to evaluate h . The above equation can be expressed by $f = \mathbf{1} + (1 - \alpha)P h - h$, where f is a scalar multiple of the unit vector e_j . Let f be ce_j . Let e_j be the indicator vector for node j . We can also write this as

$$h = (I - (1 - \alpha)P)^{-1} (\mathbf{1} - ce_j) \quad (3)$$

Where c is a scalar such that $h(j) = 0$. Now note that $(I - (1 - \alpha)P)^{-1}\mathbf{1} = \sum_{t=0}^{\infty}(1 - \alpha)^t P^t \mathbf{1}$, which simply translates to $1/\alpha \times \mathbf{1}$. The i, j^{th} entry of the matrix $(I - (1 - \alpha)P)^{-1}$ is $\sum_{t=0}^{\infty}(1 - \alpha)^t P^t(i, j) = \frac{v_i(j)}{\alpha}$. Hence we have

$$h(i) = \begin{cases} \frac{1}{\alpha} - c \frac{v_i(j)}{\alpha} & \text{when } i \neq j \\ \frac{1}{\alpha} - c \frac{v_j(j)}{\alpha} & \text{when } i = j \end{cases} \quad (4)$$

Since $h(j) = 0$, we have $c = 1/v_j(j)$. Hence $h(i) = \frac{1}{\alpha}[1 - \frac{v_i(j)}{v_j(j)}]$. \square

3 Random Projections and Personalized Pagerank

Since we have shown that many useful measures can be computed using personalized pagerank, computing personalized pagerank quickly would help computing these other measures as well. There has been a lot of work on computing personalized pagerank. To name a few the techniques involve hub-decomposition ([5]), storing fingerprints of random walks from every node ([4]), storing sparse representations of personalized pagerank from all nodes ([10]) etc. This approach is completely different from these in the sense that we relate personalized pagerank in undirected graphs to pairwise dot-products between position vectors of the nodes in the graph mapped to a high dimensional space using the graph laplacian. This naturally leads us to random projections for mapping the nodes to a much smaller dimensional subspace. This is similar in essence to the technique introduced by [11].

From equation 2 we see that personalized pagerank from node i to j is simply $\alpha d(j)L_{\alpha}^{-1}(i, j)$. Note that L_{α}^{-1} is a valid kernel, since it is positive semi-definite. Hence each entry of this matrix can be interpreted as the dot product between χ_i and χ_j , where χ_i is the projection under the kernel induced by the graph Laplacian, i.e. $\chi_i = L_{\alpha}^{-1/2}e_i$. This is true for any square root, but we will specify a particular one below. How do we take the square-root of L_{α}^{-1} ? This is possible because L_{α} can be written as $B_{\alpha}^T W_{\alpha} B_{\alpha}$. First, we will show how to obtain these matrices, i.e. B and W for the original graph Laplacian L using existing literature. Then we will derive B_{α} and W_{α} for the Laplacian with discount factor α , i.e. L_{α} .

B is the *signed edge-node incidence matrix* $m \times n$ matrix, where n is the number of nodes and m is the number of edges. We paraphrase the definition of B from [11]:

$$B(e, i) = \begin{cases} +1 & \text{if } i \text{ is } e\text{'s head} \\ -1 & \text{if } i \text{ is } e\text{'s tail} \\ 0 & \text{otherwise} \end{cases}$$

W is a $m \times m$ diagonal matrix, where the e, e^{th} entry denotes the weight on edge e .

In order to account for the α discount, we will slightly modify the weights. As a result now B_{α} is of size $(m + n) \times n$. For the self edge e from i to itself we assign 1 to $B_{\alpha}(e, i)$. For the other edges it is defined exactly as B .

The new W_{α} is of size $(m + n) \times (m + n)$. We will define W_{α} as follows:

$$W_{\alpha}(e, e) = \begin{cases} (1 - \alpha)w_e & \text{If } e \text{ is not a self edge, } w_e \text{ is the original weight on the edge } e \\ \alpha d(i) & \text{If } e \text{ is a self edge for node } i, d(i) \text{ is the weighted degree of } i \end{cases} \quad (5)$$

Hence $L_{\alpha}(i, j)$ becomes $-(1 - \alpha)A(i, j)$, if there is an edge between i, j , and zero otherwise. And $L_{\alpha}(i, i)$ becomes $\sum_e B_{\alpha}(e, i)B_{\alpha}(e, i)W_{\alpha}(e, e)$, which gets $(1 - \alpha)d(i)$ contribution from its original neighbors, and $\alpha d(i)$ from the newly introduced self-loop, leading to $d(i)$. This is exactly equal to $L_{\alpha} = D - (1 - \alpha)A$. As in [11] computing the square root of L_{α}^{-1} can be avoided as follows,

$$e_i^T L_{\alpha}^{-1} e_j = e_i^T L_{\alpha}^{-1} L_{\alpha} L_{\alpha}^{-1} e_j = e_i^T L_{\alpha}^{-1} B_{\alpha}^T W_{\alpha} B_{\alpha} L_{\alpha}^{-1} e_j = \chi_i^T \chi_j$$

χ_i is simply $B_{\alpha} W_{\alpha}^{1/2} L_{\alpha}^{-1} e_i$. In other words $\chi = L_{\alpha}^{-1} B_{\alpha}^T W_{\alpha}^{1/2}$ is the $n \times (m + n)$ matrix where the i^{th} row is the projection of node i into an $m + n$ dimensional space. We will use a variation [1] of the Johnson-Lindenstrauss lemma[6] to compute a low dimensional projection of χ .

Theorem 3.1 (Achlioptas [1]: Theorem 2.) Let S be an arbitrary set of n points in R^d , represented as an $n \times d$ matrix χ . Given $\epsilon, \beta > 0$, let $k_0 = \frac{4 + 2\beta}{\epsilon^2/2 - \epsilon^3/3} \log n$. Then for $k \geq k_0$, let Q be a $d \times k$ random matrix, where the i, j^{th} entries are random variables distributed as follows:

$$q_{ij} = \begin{cases} +1 & \text{with probability } 1/2 \\ -1 & \dots 1/2 \end{cases} \quad (6)$$

Let $E = \frac{\chi Q}{\sqrt{k}}$, and $f : \mathbb{R}^d \rightarrow \mathbb{R}^k$ map the i^{th} row of χ to the i^{th} row of E . With probability at least $1 - n^{-\beta}$, for all $\mathbf{x}, \mathbf{y} \in S$,

$$(1 - \epsilon) \|\mathbf{f}(\mathbf{x}) - \mathbf{f}(\mathbf{y})\|^2 \leq \|\mathbf{f}(\mathbf{x}) - \mathbf{f}(\mathbf{y})\|^2 \leq (1 + \epsilon) \|\mathbf{f}(\mathbf{x}) - \mathbf{f}(\mathbf{y})\|^2$$

How does the above translate to dot products? A simple derivation [9] shows that, under a Johnson-Lindenstrauss projection the following holds with high probability:

$$\mathbf{x}^T \mathbf{y} - \epsilon \|\mathbf{x}\|_2^2 \|\mathbf{y}\|_2^2 \leq \mathbf{f}(\mathbf{x})^T \mathbf{f}(\mathbf{y}) \leq \mathbf{x}^T \mathbf{y} + \epsilon \|\mathbf{x}\|_2^2 \|\mathbf{y}\|_2^2$$

This implies reconstruction of dot products lead to much more variance than that of distances. What is $\|\mathbf{x}\|_2^2$ in our case? This is nothing but the diagonal entries of L_α^{-1} . Hence $L_\alpha^{-1}(i, i) = \|\chi_i\|_2^2$. This is also equal to $PPV(i, i)/(\alpha d(i))$. For low degree nodes this quantity can be large.

3.1 Algorithm

Let Q be the random matrix of size $(m + n) \times k$ from theorem 3.1. We want to compute random projections of χ . In other words we are interested in computing the $n \times k$ dimensional χQ , i.e. $L_\alpha^{-1} B_\alpha^T W_\alpha^{1/2} Q$. Note that this boils down to solving k different linear systems for each dimension, i.e. solving for χ_i , in $L_\alpha \chi_i = B_\alpha^T W_\alpha^{1/2} Q_i$, where Q_i be the i^{th} column of Q . We generate $k = O(\log n / \epsilon^2)$ projections by solving linear systems. From there we can compute the i, j^{th} entry of L_α^{-1} by computing the dot product between χ_i and χ_j . Then to compute PPV from i to j , we just use $\alpha \times (\chi_i \cdot \chi_j) \times d(j)$. Recently there has been a lot of work on solving systems involving graph Laplacians. Authors in [12], and [11] present nearly linear-time algorithms for solving a linear system. [7] present a linear-work parallel algorithm for solving planar Laplacians. Computing $B_\alpha^T W_\alpha^{1/2} Q_i$ takes roughly $O(2m + n)$ time, since B_α has $2m + n$ non-zero entries. Solving the linear system using Spielman Teng solver would incur roughly $\tilde{O}(m + n)$ time. As a result we will have a $\tilde{O}((m + n) \log n / \epsilon^2)$ time algorithm to compute $O(\log n / \epsilon^2)$ dimensional embedding of each node in the graph.

Interpretation: Let us examine the projections a little deeper. Consider the k^{th} element of the vector $B_\alpha^T W_\alpha^{1/2} Q_i$. This boils down to sampling the neighboring edges of node k leading to a sum of $\sum_e B_\alpha(e, k) W_\alpha^{1/2}(e, e) Q_i(e) = c_{ik}^+ - c_{ik}^-$. The positive part c_{ik}^+ and the negative part $-c_{ik}^-$ comes from the same or opposite signs of B_α and Q_i . The entire vector $B_\alpha^T W_\alpha^{1/2} Q_i$ now can be written after normalization as $c_i^+ \mathbf{r}_i^+ - c_i^- \mathbf{r}_i^-$, where \mathbf{r}_i^+ and \mathbf{r}_i^- are distribution vectors, and c_i^+ and c_i^- are positive scalars. This ultimately leads to solving for $L_\alpha \chi_i = c_i^+ \mathbf{r}_i^+ - c_i^- \mathbf{r}_i^-$.

$$\begin{aligned} \chi_i &= c_i^+ L_\alpha^{-1} \mathbf{r}_i^+ - c_i^- L_\alpha^{-1} \mathbf{r}_i^- \\ D\chi_i &= \frac{c_i^+}{\alpha} PPV(\mathbf{r}_i^+) - \frac{c_i^-}{\alpha} PPV(\mathbf{r}_i^-) \\ \chi_i(k) &= \frac{c_i^+}{d_k \alpha} PPV(\mathbf{r}_i^+, k) - \frac{c_i^-}{d_k \alpha} PPV(\mathbf{r}_i^-, k) \end{aligned} \quad (7)$$

The interesting observation is that, the k^{th} coordinate of node i can be represented as a linear combination of the personalized pagerank from randomly picked nodes with random coefficients.

Can we do any better? So far we have seen that we can compute $O(\log n / \epsilon^2)$ dimensional embeddings for each nodes leads to small error with high probability. But can we do better? From the seminal result from [3], which has been also used by [9], we have a much weaker guarantee than Johnson-Lindenstrauss,

Lemma 3.2 (Alon et al: [3],[2]) For $0 < \epsilon \leq 1$, let Q be a random matrix ϵR , where R is a $1/\epsilon^2 \times d$ matrix with independent rows, such that each row contains 4-wise independent zero-mean $\{-1, +1\}$ random variables. Then for any $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$ we have that $E[(\mathbf{Q}\mathbf{u})^T(\mathbf{Q}\mathbf{v})] = \mathbf{x}^T\mathbf{y}$, and $\text{Var}[(\mathbf{Q}\mathbf{u})^T(\mathbf{Q}\mathbf{v})] \leq 2\epsilon^2\|\mathbf{x}\|_2^2\|\mathbf{y}\|_2^2$.

Note that this, along with Chebyshev's inequality leads to a weak guarantee on probability of success, i.e.

$$P[|(\mathbf{Q}\mathbf{u})^T(\mathbf{Q}\mathbf{v}) - \mathbf{x}^T\mathbf{y}| \geq 2\epsilon\|\mathbf{x}\|\|\mathbf{y}\|] \leq 1/2$$

. Using this [9] provided the result below for computing matrix products with low distortion in Frobenius norm.

Lemma 3.3 (Sarlos et al: [9]) Given a constant δ of probability of failure, if we instantiate $t = \log(1/\delta)$ independent copies of the tug-of-war matrix \mathbf{Q}_t , then

$$P[\min_{i=1\dots t} \|\mathbf{L}_\alpha^{-1} - (\chi\mathbf{Q}_t)(\chi\mathbf{Q}_t)^T\|_F \leq 2\epsilon(L_\alpha^{-1})_F] \geq 1 - \delta$$

However, in order to be able to use this we would need to have an idea about what is the best Q . How would we know that without knowing the true L_α^{-1} ? The authors provide another layer of random sampling for getting a low-error Q . We however have an easy way of picking Q . We can pick the one leading to the minimum number of negative entries. This means, we can get close in frobenius norm by solving only about $1/\epsilon^2$ linear systems, $\log(1/\delta)$ times. However if we want to get all the elements right, we would need to enforce this for all n^2 elements, which would require a smaller probability of error i.e. δ/n^2 , leading to $O(\log(n^2/\delta))$ trials, which again brings us back to similar complexity of Johnson-Lindenstrauss.

4 Experiments

We will present experiments on simulated random geometric graphs. We assign each node a random coordinate, and then connect the nodes that are close to one another (in Euclidian distance) with high probability. First we will show results on a graph with 300 nodes and 1900 edges. We sampled a set of 100 nodes and computed the mean absolute difference of the i^{th} row of L_α^{-1} and the reconstruction from $\chi\mathbf{Q}$. Note that dot products arising from the projections can be negative, whereas probabilities can never be. However we can easily *correct* this by thresholding. Using the mean and median of the absolute deviation of reconstructed vector from truth can be misleading because, personalized pagerank values are fairly sparse owing to the restart probability. Hence the *all-zero* vector would have very low mean and median errors as well. In order to avoid this and at the same time correct for the negative values, we examined the 50 largest positive values of the reconstructed and true vectors and computed mean and median over these entries only.

We used $k = 10000, 1000$ and 50 for our experiments. Although this looks unhelpful for small graphs with up-to about 10,000 edges, this is necessary in order to deal with the leading constant. The hope is that as we increase the graph size, k would grow slowly, thus letting us exploit the logarithmic asymptote. The top panel of figure 1(A) contains the histogram of mean and median errors. As for figure 1(B) we see that the error varies inversely with degree as we had conjectured in the previous section.

We also did experiments on two graphs with 1000 nodes. Figure 2 (A) contains results for a dense graph, whereas (B) contains results on a 1000 node graph with about half the edges. We can see that errors are in general larger in the sparser graph, which makes sense given that the mean degree is much smaller in the sparse graph. Finally we present the histogram of errors on a 17,000 node, 40,000 edge subgraph of Citeseer in figure 3. For $k = 1000$ the mean errors are roughly concentrated around 0.006. This means that $k = 1000$ leads to low error over the entire range of graph sizes we tested.

5 Conclusion and Future Direction

In this paper we showed how to use random projections for computing personalized pagerank, a popular graph-based proximity measure. These random projections can also be interpreted as a linear combination of personalized pagerank values from randomly picked nodes in the graph. We showed that discounted hitting and commute times can be expressed in terms of personalized pagerank. We presented preliminary experimental results on small random geometric graphs of size 300 and 1000,

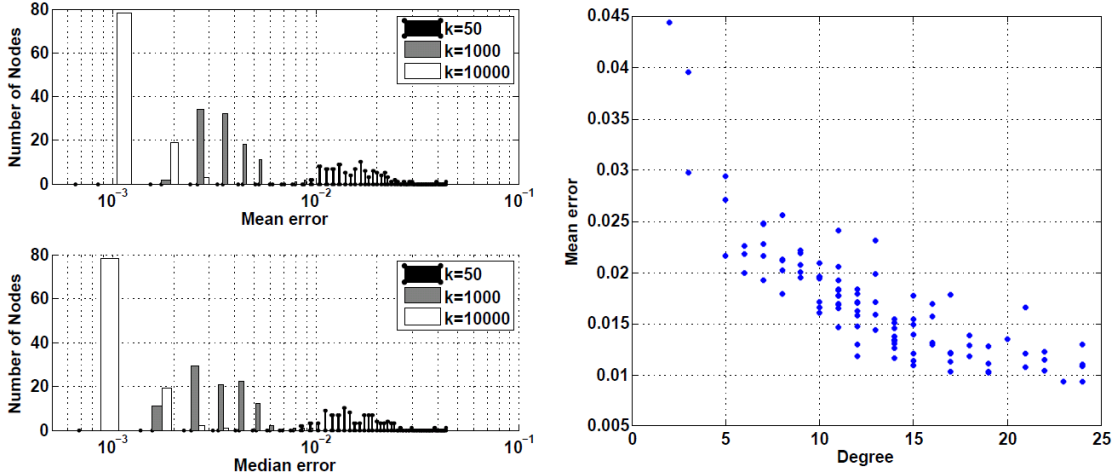


Figure 1: (A) Histogram of errors in a 300 node graph, Upper panel contains mean error and bottom panel contains median error. (B) Scatterplot of mean error for $k = 50$ with degree.

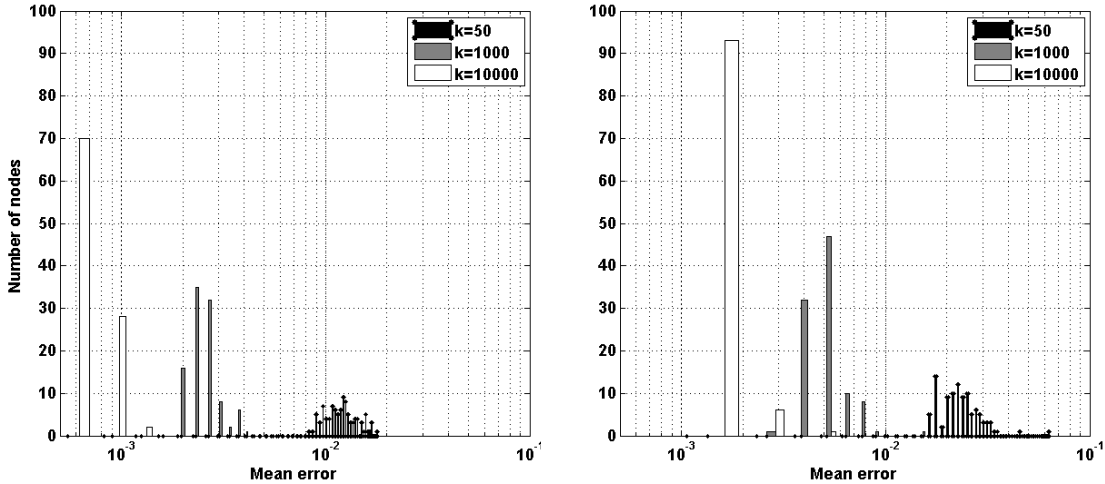


Figure 2: (A) Histogram of errors in a 1000 node graph in log scale. (B) Exactly as in A. but for a sparse graph with about half the number of edges, and 1000 nodes.

and a small 17,000 node real world co-authorship graph from the Citeseer domain. Our experiments bring us to an array of interesting questions. When do we start enjoying the logarithmic asymptote of $O(\log n/\epsilon^2)$? How does the error vary with the structure of the graph, for example how would this algorithm behave on a small-world social network vs. a network of neighboring pixels built from a 2D image? We are getting negative probabilities; is there a better projection technique to automatically get rid of negative numbers? How does this relate to compressed sensing?

Acknowledgments

This work was partially supported by the USDA (award 1040770), CDC (R01 – PH000028), and NSF (IIS – 0911032).

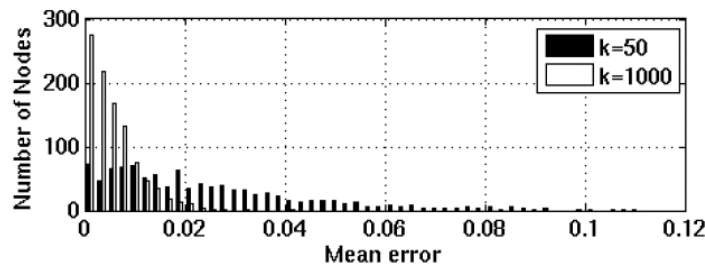


Figure 3: Citeseer error histogram with 1000 sampled nodes

References

- [1] D. Achlioptas. Database-friendly random projections. In *PODS '01: Proceedings of the twentieth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 274–281, New York, NY, USA, 2001. ACM.
- [2] N. Alon, P. B. Gibbons, Y. Matias, and M. Szegedy. Tracking join and self-join sizes in limited storage. In *PODS '99: Proceedings of the eighteenth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 10–20, New York, NY, USA, 1999. ACM.
- [3] N. Alon, Y. Matias, and M. Szegedy. The space complexity of approximating the frequency moments. In *STOC '96: Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 20–29, New York, NY, USA, 1996. ACM.
- [4] D. Fogaras, B. Rcz, K. Csalogny, and T. Sarls. Towards scaling fully personalized pagerank: Algorithms, lower bounds, and experiments. *Internet Mathematics*, 2004.
- [5] G. Jeh and J. Widom. Scaling personalized web search. In *Stanford University Technical Report*, 2002.
- [6] W. Johnson and J. Lindenstrauss. Extensions of lipschitz mappings into a hilbert space. In *In Conference in modern analysis and probability*, 1984.
- [7] I. Koutis and G. L. Miller. A linear work, $o(n^{1/6})$ time, parallel algorithm for solving planar laplacians. In *SODA '07: Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1002–1011, Philadelphia, PA, USA, 2007. Society for Industrial and Applied Mathematics.
- [8] L. Lovasz. Random walks on graphs: a survey. 1996.
- [9] T. Sarlós. Improved approximation algorithms for large matrices via random projections. In *FOCS '06: Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science*, pages 143–152, Washington, DC, USA, 2006. IEEE Computer Society.
- [10] T. Sarlós, A. A. Benczúr, K. Csalogány, D. Fogaras, and B. Rác. To randomize or not to randomize: space optimal summaries for hyperlink analysis. In *WWW '06*.
- [11] D. Spielman and N. Srivastava. Graph sparsification by effective resistances. In *Proceedings of the STOC'08*, 2008.
- [12] 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.