Multiplicative approximations of random walk transition probabilities

Michael Kapralov* Rina Panigrahy†

June 17, 2011

Abstract

We study the space and time complexity of approximating distributions of l-step random walks in simple (possibly directed) graphs G. While very efficient algorithms for obtaining *additive* ϵ-approximations have been developed in the literature, non non-trivial results with multiplicative guarantees are known, and obtaining such approximations is the main focus of this paper. Specifically, we ask the following question: given a bound S on the space used, what is the minimum threshold $t > 0$ such that l-step transition probabilities for all pairs $u, v \in V$ such that $P^l_{uv} \geq t$ can be approximated within a $1 \pm \epsilon$ factor? How fast can an approximation be obtained?

We show that the following surprising behavior occurs. When the bound on the space is $S = o(n^2/d)$, where d is the minimum out-degree of G, no approximation can be achieved for non-trivial values of the threshold t. However, if an extra factor of s space is allowed, i.e. $S = \tilde{\Omega}(sn^2/d)$ space, then the threshold t is *exponentially small in the length of the walk* l and even very small transition probabilities can be approximated up to a $1 \pm \epsilon$ factor. One instantiation of these guarantees is as follows: any almost regular directed graph can be represented in $\tilde{O}(ln^{3/2+\epsilon})$ space such that all probabilities larger than n^{-10} can be approximated within a $(1 \pm \epsilon)$ factor as long as $l \geq 40/\epsilon^2$. Moreover, we show how estimates of all such probabilities can be found faster than matrix multiplication time.

Ours results can be related to some of the known fast algorithms for approximating matrix products in terms of Frobenius or spectral norm – we essentially use a variant of row/column sampling to give a fast algorithm for obtaining a significantly more precise approximation to A^l for a special class of matrices A.

For undirected graphs, we also give small space oracles for estimating P^l_{uv} using a notion of bicriteria approximation based on approximate distance oracles of Thorup and Zwick [STOC'01].

*Institute for Computational and Mathematical Engineering, Stanford University, Stanford, CA. Email: kapralov@stanford.edu. Research supported in part by NSF award IIS-0904325. Part of the work was done when the author was an intern at Microsoft Research Silicon Valley.

†Microsoft Research Silicon Valley, Mountain View, CA. Email: rina@microsoft.com

1 Introduction

Large scale graphs are now a widely used tool for representing real world data. Many modern applications, such as search engines or social networks, require supporting various queries on large-scale graphs efficiently. An important operation is calculating some measure of distance between two nodes in a graph. Random walks recently emerged as an important tool for measuring distance between nodes in such graphs. PageRank[22] and its personalized version is one of the popular random walk based measures of relatedness of nodes in a graph. Personalized PageRank uses the distribution of a short random walk of exponentially distributed length started at a source node to measure proximity between the source and other nodes in the graph. Methods for computing personalized PageRank have received significant attention recently. Very efficient algorithms are known for computing and updating (personalized) Pagerank vectors in various computation models such as random access and streaming (see, e.g. [17, 18, 1, 26, 27, 2]). The techniques that have been used range from methods from linear algebra to Monte Carlo simulation and dynamic programming. It is hard to do justice to the large body of work on this topic in limited space, and we refer the reader to the great surveys [19, 5] and recent papers on Pagerank computation (see, e.g. [2]) for a more complete discussion of prior work on Pagerank. The approximation obtained via these methods is usually an *additive* ϵ-approximation for some $\epsilon > 0$. With this measure of approximation, it is in general sufficient to perform $O(\log n/\epsilon)$ random walks of small length to obtain an approximation. While additive approximations are well-studied, no non-trivial multiplicative $(1 \pm \epsilon)$-factor approximations are known, and obtaining such approximations using small space and time is the main focus of this paper.

Given a simple (possibly directed) graph $G = (V, E)$ and a bound S on the space we can use, we would like to obtain $(1 \pm \epsilon)$-factor approximations of l-step transition probabilities P^l_{uv} from u to v, *for all pairs* $u, v \in V$ *such that* $P^l_{uv} \geq t$, for the smallest possible threshold value t. We show that the following surprising behavior occurs. If the minimum out-degree of the graph G is d, then no $(1 \pm \epsilon)$-factor approximation can be achieved for any reasonable value of t if the space S available is below $o(n^2/d)$. However, if the available space is a factor $s > 1$ larger than $\tilde{\Omega}(n^2/d)$, then $(1 \pm \epsilon)$-factor approximation can be achieved for probabilities larger than $t \geq \frac{1}{d} s^{-(\epsilon/4)(l-1)}$. Thus, increasing the amount of space available by a factor of $s > 1$ allows one to approximate transition probabilities that are *exponentially small in the length of the walk* l. One instantiation of such guarantees is that any regular graph can be represented in $\tilde{O}(ln^{3/2+\epsilon}/\epsilon^2)$ space so that $(1 \pm \epsilon)$-approximations to transition probabilities P^l_{uv} can be recovered for all pairs $u, v \in V$ such that $P^l_{uv} \geq n^{-10}$ as long as $l \geq 40/\epsilon^2$.

These bounds are nearly-tight: we show that the space complexity of obtaining $(1 \pm \epsilon)$-approximations to P^l_{uv} for all pairs such that $P^l_{uv} \geq \frac{1}{d_u}(s/2)^{-(l-1)}$ is $\Omega(sn^2/d)$. Additionally, our techniques yield fast algorithms for calculating very precise approximations to l-step random walk transition probabilities for special classes of graphs. For example, it follows that an $(1\pm\epsilon)$-factor approximation to P^l_{uv} for all pairs such that $P^l_{uv} \geq n^{-10}$, say, can be obtained in time $\tilde{O}(n^{2+(1+\epsilon)\frac{\omega-2}{\omega-1}})$, where $\omega \geq 2$ is the matrix multiplication constant, for any (almost) regular graph. Thus, we show that very precise approximations can be obtained strictly faster than matrix multiplication if $\omega > 2$. A very interesting question raised by our work is whether our techniques can be used to obtain similar approximations to the problem of multiplying two matrices with non-negative entries faster than matrix multiplication time. We note that besides being of intrinsic theoretical interest, multiplicative approximation guarantees for all transition probabilities above a very small threshold may be useful, for example, in applications to probabilistic model checking, where one is interested in the probability that a Markov chain reaches an 'error state' in a given number of steps ([24]).

We also consider the problem of obtaining approximate oracles for random walk transition probabilities for undirected graphs. While almost-optimal tradeoffs for the problem of approximating distances have been

obtained in the large body of work on spanners ([7, 8, 11, 29, 31, 4, 30, 23]), we are not aware of any known results on small space oracles for approximating random walk transition probabilities. Our lower bounds on the space complexity of multiplicative approximation suggest that a weaker notion of approximation is required if close to linear space is desired. A natural candidate notion is a relaxation of the length of the random walk similar to the approximations obtained in the well-known distance oracles of Thorup and Zwick[29]. Surprisingly, it turns out that such relaxation is too strong for our problem. However, we show that a more relaxed notion of bicriteria approximation can be used to obtain small space oracles based on the techniques of [29].

1.1 Related work

Approximate matrix multiplication It is interesting to compare our results on multiplicative approximation to the line of work on approximate matrix computations (e.g. [9, 15, 10, 25, 21, 6, 20]), which develops extremely efficient approximate randomized algorithms for several problems in numerical linear algebra. A lot of these algorithms can be used in the streaming model, requiring close to optimal space an only a small constant number of passes over the data. These algorithms are also very general and apply to arbitrary matrices. There is a vast literature on this topic, and we refer the reader to some of the latest papers for a more complete list of references (see, e.g. [20]). All of the developed algorithms for approximating matrix products yield approximation guarantees in term of the Frobenius or spectral norm. To highlight the relation to our results, we now compare our algorithm to the earlier algorithm of [9], which is also based on column/row sampling. In [9] the authors show that for any $s > 1$ there exists a randomized algorithm that given two matrices $A, B \in \mathbb{R}^{n\times n}$ outputs a matrix P such that

$$\mathbf{E}[||P - A \cdot B||_F^2] \leq \frac{1}{s}||A||_F^2||B||_F^2 \tag{1}$$

in time $\tilde{O}(n^2 s)$ by carefully sampling rows of A and columns of B. On the other hand, our multiplication approximation algorithm computes the l-th power of a matrix A, giving entrywise approximation guarantees. Let A be the random walk transition matrix of an (almost)regular graph. We show how to calculate $(1 \pm \epsilon)$-factor approximation *to every sufficiently large entry of* A^l in time $\tilde{O}(n^{2+(1+\epsilon)\frac{\omega-2}{\omega-1}})$ for any sufficiently large l. It should be noted that the best speedups that our techniques currently yield are for a special class of graphs. On the other hand, we obtain a significantly more precise estimate than (1) faster than matrix multiplication time. Since $||A||_F$ can in general be very large, there is no way of setting the parameter s in the algorithm of (1) that would allow to obtain such precision faster than matrix multiplication.

It is interesting to note that the approach we take also uses row and column sampling. However, the process is somewhat different: we show that for each pair $u, v \in V$ such that the (u, v) entry of A^l is sufficiently large *there exists* j, $1 \leq j \leq l-1$ such that sampling rows of A^j and columns of A^{l-j} at an appropriate rate allows one to get a $1 \pm \epsilon$ approximation for A^l_{uv}. It seems plausible that similar techniques could be used to obtain good approximations to the product of two matrices with non-negative entries faster than matrix multiplication time. This currently remains a very interesting open problem.

Another connected line of work concerns efficiently representing various notions of distance between nodes of a weighted undirected graph, which we now describe.

Distance oracles. A large body of work on spanners and emulators provides almost optimal tradeoffs for the case when the distance measure of interest is the shortest path distance. A *spanner* is a subgraph H of the target graph G that approximates the shortest path metric on G in some sense. Several notions

of spanners have been considered in the literature. A *multiplicative spanner* is a subgraph H of G such that $\delta_G(u,v) \leq \delta_H(u,v) \leq t\delta_H(u,v)$ for some constant $t > 1$ called the *stretch* of H. The landmark paper of [29] provides almost optimal tradeoffs between the size of H and the stretch, assuming a widely believed girth conjecture of Erdős. In particular, [29] construct stretch $2k-1$-spanners with $O(n^{1+1/k})$ edges. Moreover, their construction has $O(k)$ query time, thus yielding an approximate distance oracle. Very recently, [23] gave an oracle that returns a path of length at most $2\delta_G(u,v)+1$ and uses $O(n^{5/3})$ space for unweighted graphs. Efficient algorithms for constructing spanners in the streaming model have been proposed in [12, 3].

Additive spanners are subgraphs H of G that approximate shortest path distances in G up to an additive error. Additive spanners with sublinear (in the distance) additive approximation error have been constructed for unweighted graphs in [30]: the authors show how to construct a subgraph H of G such that if $\delta_G(u,v) = d$, then $d \leq \delta_H(u,v) \leq d + O(d^{1-1/(k-1)})$. Another useful notion of approximation is obtained if the condition of H being a subgraph of G is removed. In particular, H is an *emulator* for G is H is a (possibly weighted) graph on the same set of nodes that approximates the shortest path metric of G. Approximation/space tradeoffs known for emulators are significantly better than for additive spanners: the authors of [30] construct emulators H with $O(n^{1+1/(2^k-1)})$ edges such that if $\delta_G(u,v) = d$, then $d \leq \delta_H(u,v) \leq d + O(d^{1-1/(k-1)})$. One disadvantage of these constructions of additive spanners and emulators, however, is that even though the constructed graph H approximates distances in G well, no fast method for answering distance queries using H is known.

Effective resistance. Other distance measures besides shortest path distance are of value in many real-world scenarios. For example, in networks that can be modeled by preferential attachment graphs, whose diameter is logarithmic in the size of the network, a notion of distance that takes the number of paths between two nodes is more desirable. One such notion is commute time, which has been considered as a measure of distance in social networks ([13, 14]). The results of [28] show that a data structure that returns $(1+\epsilon)$-approximate answers in $O(\log n/\epsilon^2)$ time and takes $O(n \log n/\epsilon^2)$ space can be produced in $\tilde{O}(m)$ time. This algorithm can also be implemented in the semi-streaming model with at most a polylogarithmic loss in efficiency and the space required.

$s-t$ mincuts. The minimum $s-t$ cut is another important measure of node proximity. It is known [16] that all min-cuts in a weighted undirected graph can be represented in $O(n)$ space by storing the Gomory-Hu tree T of the graph. Additionally, this algorithm can be implemented in the semi-streaming model with at most a polylogarithmic loss in efficiency and the space required.

1.2 Our Results and Techniques

Directed graphs

For a graph $G = (V, E)$ and a pair of nodes $u, v \in V$ we denote the probability of reaching v from u in l steps of a simple random walk on G by P^l_{uv}.

Our main result is

Theorem 1 *Let $G = (V, E)$ be a (possibly directed) graph with minimum out-degree bounded below by d. For any $s \geq 1$ there exists a data structure that allows returning $1 \pm \epsilon$-factor approximations to P^l_{uv} for all $u, v \in V$ such that $P^l_{uv} \geq (1/d)s^{-(\epsilon/4)(l-1)}$ using space $\tilde{O}\left(\frac{lsn^2}{\epsilon^2 d}\right)$. Moreover, whp for any u, v the*

approximation to P^l_{uv} that is returned does not overestimate P^l_{uv} by more than a $1+\epsilon$ factor. The query time is $\tilde{O}(lns/(\epsilon^2 d))$ and the preprocessing time is $\tilde{O}(l \cdot \min\{n^\omega, lsn^3/(\epsilon^2 d)\})$.

Remark 2 *Note that the preprocessing time is $o(n^\omega)$ for $d = \Omega(sn^{3-\omega})$ and $l = \tilde{O}(1)$.*

Note that the threshold above which we can provide $1 \pm \epsilon$-approximations to P^l_{uv} depends exponentially on the length of the walk l. Thus, even for moderate values of s Theorem 1 potentially allows to approximate transition probabilities for all pairs $u, v \in V$. One instantiation of Theorem 1 is as follows:

Corollary 3 *Let $G = (V, E)$ be a regular graph. Then all l-step transition probabilities $P^l_{uv} \geq n^{-10}$ can be approximated up to $1 \pm \epsilon$ for any $\epsilon > 0$ in space $\tilde{O}(ln^{3/2+\epsilon}/\epsilon^2)$ as long as $l \geq 40/\epsilon^2$.*

Proof: Set $s = n^\epsilon$. If $d = \Omega(n^{1/2+\epsilon})$, then Theorem 1 can be used to compress the representation to $\tilde{O}(ln^{2+\epsilon}/\epsilon^2 d) = \tilde{O}(ln^{3/2+\epsilon}/\epsilon^2)$ space. Otherwise one can store the entire graph in $\tilde{O}(n^{3/2+\epsilon})$ space. ■

Remark 4 *Perhaps the most natural interpretation of the guarantees given by our algorithm is as follows. Given the compressed representation of the graph, if for a query (u, v) the value $\hat{P}^l_{uv}$ returned by the algorithm is large (i.e. satisfies $\hat{P}^l_{uv} \geq n^{-10}$), then it is guaranteed to be within $(1 \pm \epsilon)$ factor of the true transition probability. Otherwise, the true transition probability is very small, i.e. below n^{-10}.*

Note that it is entirely possible that all l-step transition probabilities in a regular graph are larger than n^{-10}, in which case one can approximate each entry of A^l up to an $1 \pm \epsilon$ factor, where A is the random walk transition matrix of G.

Interestingly, our algorithm yields a method for approximating sufficiently large l-step transition probabilities in a regular graph faster than matrix multiplication:

Corollary 5 *$P^l_{uv} \geq n^{-10}$ can be approximated up to $1 \pm \epsilon$ for any $\epsilon > 0$ in time $\tilde{O}(ln^{2+(1+\epsilon)(\omega-2)/(\omega-1)})$ as long as $l \geq 40/\epsilon^2$*

We now show that the upper bound given by Algorithm 2 is of the right form:

Theorem 6 *Any algorithm that gives a constant factor approximation to $P^l_{uv} \geq \frac{1}{d_u}(s/2)^{-(l-1)}$ in an undirected graph $G = (V, E)$ with minimum degree d must use $\Omega\left(\frac{sn^2}{l^2 d}\right)$ space.*

Undirected graphs

The lower bound given by Theorem 6 suggests that near-linear space cannot be achieved independently of the minimum degree of the graphs if a constant factor approximation to P^l_{uv} is desired. In light of the result of [29] on $2k-1$-approximate distance oracles for undirected graphs in space $O(n^{1+1/k})$ it is natural to conjecture that one can output a constant factor approximation to the probability of reaching v from u in $(2j-1)t$ steps for some $1 \leq j \leq k$ in $O(n^{1+1/k})$ space. Perhaps surprisingly, we show that such approximation cannot be achieved for random walk transition probabilities:

Lemma 7 *Any algorithm that given a weighted graph $G = (V, E)$ for any pair of nodes (u, v) outputs an estimate $\hat{p}(u, v)$ such that*

$$P^l_{u,v} \leq \hat{p} \leq (1+\epsilon) \max_{1 \leq j \leq 2k-1} P^{jl}_{u,v}$$

for any constant $\epsilon > 0$ must use $\Omega(n^2/2^{kl})$ space.

Lemma 7 motivates more relaxed bicriteria approximation. In particular, when queried about the value of $P^l_{u,v}$, the algorithm may return an approximation to the logarithm of $P^{jl}_{u,v}$ for any $1 \le j \le 2k-1$. The following theorem shows that a somewhat more powerful notion of approximation is possible:

Theorem 8 *Let $G = (V, E)$ be a weighted undirected graph such that $1/\gamma \le d(u)/d(v) \le \gamma$ for all $u, v \in V$ for some constant $\gamma \ge 1$. There exists an $O(k^3 n^{1+1/k} \log n)$ space and $O(k^3 \log n)$ query time oracle that given a pair $u, v \in V$ outputs a value $\hat{p}$ such that*

$$\gamma^{-1} P^l_{u,v} \le \hat{p} \le 4 \max_{1 \le j \le k} \left(P^{(2j-1)l}_{u,v} / n^{(j-1)/k} \right)^{1/(2j-1)}.$$

The preprocessing time is bounded by $\tilde{O}(kln^\omega)$, where $\omega \ge 2$ is the matrix multiplication constant.

Note that the difference between the statement of Theorem 8 differs from the notion of bicriteria approximation outlined above in the extra normalization term $n^{(j-1)/k}$, which only improves the guarantee. The algorithm is based on approximate distance oracles of Thorup and Zwick.

Theorem 8 applies to γ-regular graphs. For general graphs, we approximate the symmetrized quantity $S_t(u, v) = \sqrt{P_t(u, v) P_t(v, u)} = \sqrt{d(u)/d(v)} P_t(u, v)$:

Theorem 9 *Let $G = (V, E)$ be a weighted undirected graph. There exists an $O(k^3 n^{1+1/k} \log n)$ space and $O(k^3 \log n)$ query time oracle that given a pair $u, v \in V$ outputs a value $\hat{p}$ such that*

$$S^l_{u,v} \le \hat{p} \le 4 \max_{1 \le j \le k} \left(S^{(2j-1)l}_{u,v} / n^{(j-1)/k} \right)^{1/(2j-1)}.$$

The preprocessing time is bounded by $\tilde{O}(kln^\omega)$, where $\omega \ge 2$ is the matrix multiplication constant.

This approximation ratio is optimal up to an $O(\log n)$ factor in the following sense:

Theorem 10 *Any algorithm that outputs an estimate $\hat{p}$ such that*

$$P^l_{u,v} \le \hat{p} \le (1+\epsilon) \max_{1 \le j \le k} \left(P^{(2j-1)l}_{u,v} / n^{(j-1)/k} \right)^{1/(2j-1)}$$

for a constant $\epsilon > 0$ must use $\Omega(n^{1+1/k}/2^{lk})$ space.

2 Directed graphs

2.1 Algorithm

In this section we give an algorithm for approximating random walk transition probabilities in simple graphs $G = (V, E)$. Most proofs from this section are deferred to Appendix A due to space considerations.

Given a bound $\tilde{O}(lsn^2/d)$ on the available space, where $s \ge 1$, the preprocessing Algorithm 1 samples $O(\log n/\epsilon)$ sets of *centers* $C_t \subseteq V$, by including each vertex into C_t uniformly at random with probability $r = \frac{4s \log n}{\epsilon^2 (1-\epsilon/2) d}$. For each center $c \in C_t$ the algorithm computes and stores P^j_{uc} and P^j_{cu} for all $u \in V$ *and all* $1 \le j \le l-1$:

1: **for** $t = 1, \ldots, 4 \log n/\epsilon$ **do**
2: Choose $r = \frac{4s \log n}{\epsilon^2 (1-\epsilon/2) d}$ centers C_t uniformly at random.
3: For each $c \in C_t$ and for each $j, 1 \le j \le l-1$ calculate and store P^j_{uc}, P^j_{cu} for all $u \in V$.
4: **end for**

Algorithm 1: Preprocessing algorithm

At query time, given a pair of vertices $u, v \in V$, for each of the $O(\log n/\epsilon)$ sets of centers C_t and for each $1 \leq j \leq l-1$ Algorithm 2 calculates an estimate

$$\hat{p}_{t,j} := \frac{1}{r} \sum_{c \in C_t} P^j_{uc} P^{l-j}_{cv}. \tag{2}$$

Finally, the algorithm sets

$$\hat{p}^{min}_j := \min_{1 \leq t \leq 4 \log n/\epsilon} \hat{p}_{t,j} \tag{3}$$

and returns the largest of $\hat{p}^{min}_j$, $1 \leq j \leq l-1$ as its estimate.

1: **for** $t = 1, \ldots, 4 \log n/\epsilon$ **do**
2: $\quad \hat{p}_{t,j} \leftarrow \frac{1}{r} \sum_{c \in C_t} P^j_{uc} P^{l-j}_{cv}$.
3: **end for**
4: $\hat{p} \leftarrow \max_{j=1,\ldots,l-1} \min_{t=1,\ldots,4 \log n/\epsilon} p_{t,j}$
5: **return** $\hat{p}$

Algorithm 2: Estimation algorithm

The estimator in (2) is unbiased, i.e. $\mathbf{E}[\hat{p}_{t,j}] = P^l_{uv}$ for all t and j since the set of centers was chosen uniformly at random. The question that we consider is for which $u, v \in V$ and for what sampling rate r is $\hat{P}^l_{uv}$ is tightly concentrated around its expectation.

It is easy to construct examples that show that in general there need not be any useful concentration for nontrivial sampling probabilities if $1 \pm \epsilon$ factor approximation of P^l_{uv} is desired for all pairs $u, v \in V$. Moreover, one cannot in general guarantee tight concentration for any *fixed* j under reasonable conditions on u and v. However, in this section we will show that in fact for each $u, v \in V$ such that P^l_{uv} is sufficiently large the estimate $\hat{p}_{t,j}$ from (2) will not underestimate by more than a $1-\epsilon$ factor whp *for at least one choice of* j *between* 1 *and* $l-1$ *and for all* t. The main technical part of our analysis is analysing conditions under which $\hat{p}_{t,j}$ does not underestimate P^l_{uv}, since overestimation can be easily dealt with by independent repetition.

We first give the intuition behind the analysis. We assume that the graph is undirected and strictly regular for simplicity. Then P^l_{uv} for a pair $u, v \in V$ is just the number of l-step walks from u to v divided by d^l, where d is the degree. We need to relate the number of l-step walks from u to v to the probability that the random sampling in (2) underestimates by more than $1-\epsilon$ factor. Fix j between 1 and $l-1$ and for a node $c \in V$ let

$$\alpha(j, c) = \frac{P^j_{uc} P^{l-j}_{cv}}{P^l_{uv}}$$

be the fraction of l-step walks from u to v that pass through c at step j. By Bernstein's inequality the probability of underestimating by more than a factor of $1-\epsilon$ can be related to the variance of the sequence $\alpha(j, c), c \in V$. Thus we need to relate the variance of $\alpha(j, \cdot)$ to the number of walks from u to v. In order to do that, we consider a uniformly random l-step walk $P = (X_1, \ldots, X_{l-1})$ from u to v, where X_j is the (random) j-th vertex in the walk. The number of paths is thus equal to the entropy $H(P)$. However, by a well-known property of the entropy function we have that

$$H(P) = H(X_1, \ldots, X_{l-1}) \leq \sum_{j=1}^{l-1} H(X_j | X_{j-1}, \ldots, X_1) \leq \sum_{j=1}^{l-1} H(X_j | X_{j-1}). \tag{4}$$

We now note that the distribution of X_j is given by $\alpha(j,\cdot)$ defined above. Thus, it is sufficient to bound

$$\min_{(X,Y),X\propto\alpha(j,\cdot),(X,Y)\in E} H(X|Y) \tag{5}$$

in terms of ϵ and the variance of $\alpha(j,\cdot)$. Here the minimization is over all random variables (X,Y) taking values in $(V\times V)\cap E$ (corresponding to the j-th and $(j-1)$-st vertex on the random walk from u to v) such that the distribution of X is $\alpha(j,\cdot)$. Given such a bound, we conclude that if (2) underestimates by more than a factor $1-\epsilon$ *for all* j *between* 1 *and* $l-1$, then there must be very few paths between u and v.

We have given the sketch of the proof for regular graphs, where the regularity allows the use of the entropy function for bounding P^l_{uv}. In the more general case of graphs of minimum degree d we use the relative entropy, or Kullback-Leibler divergence, with respect to the inverse degree sequence of the graph. We now give the details of the proof.

Denote the set of l-step walks in G from u and v by $\mathcal{P}^l_{uv}\subseteq V^{l-1}$. We write $P=(v_1,\ldots,v_{l-1})\in\mathcal{P}^l_{uv}$ to denote the path $(u,v_1,\ldots,v_{l-1},v)$. Let $\mu^l:\mathcal{P}^l_{uv}\to[0,1]$ be the natural distribution on paths $P=(v_1,\ldots,v_{l-1})\in\mathcal{P}^l_{uv}$ defined as

$$\mu((v_1,\ldots,v_{l-1}))=\begin{cases}\frac{1}{P^l_{uv}}\frac{1}{d_u}\prod_{j=1}^{l-1}\frac{1}{d_{v_j}} & \text{if } (v_1,\ldots,v_{l-1})\in\mathcal{P}^l_{uv}\\ 0 & \text{o.w.}\end{cases}$$

Note that this is indeed a distribution, i.e. $\sum_{P\in\mathcal{P}^l_{uv}}\mu(P)=1$.

Also, for $P=(v_1,\ldots,v_{l-1})\in V^{l-1}$ define

$$\eta^l((v_1,\ldots,v_{l-1}))=\frac{1}{d_u}\prod_{j=1}^{l-2}\frac{1}{d_{v_j}}.$$

For a vertex $v\in V$ define $\eta(v)=\frac{1}{d_v}$. We note that η^l is in general not a distribution.

Recall that for two distributions on a set X the relative entropy of p with respect to q is defined as

$$D(p||q)=\sum_{x\in X}p_x\log(q_x/p_x). \tag{6}$$

Note that (6) is defined even when q is only guaranteed to be non-negative. Note that the usual definition uses $\log(p_x/q_x)$, but we use this one in order to get an inequality of the form (4).

We will use the following:

Claim 11

$$P^l_{uv}\le\frac{1}{d}e^{D(\mu||\eta)}.$$

Proof: Indeed,

$$D(\mu||\eta)=\sum_{P\in\mathcal{P}^l_{uv}}\mu(P)\log(\eta(P)/\mu(P))=\sum_{P\in\mathcal{P}^l_{uv}}p(P)\log(d_{v_{l-1}}P^l_{uv})\ge\log(dP^l_{uv}).$$

■

For two random variables X,Y taking values in V define

$$D^*(X|Y||\eta)=\sum_{y\in V}p(y)\sum_{x\in V}p(x|y)\log(\eta(y)/p(x|y)). \tag{7}$$

We will use the following

Lemma 12 *Let $(X_1, \ldots, X_{l-1})$ be a random variable taking values in V^{l-1}. Let $X_0 = u$. Then one has*

$$D((X_1, \ldots, X_{l-1})||\eta^l) \leq \sum_{j=1}^{l-1} D^*(X_j|X_{j-1}||\eta)$$

Note that this is the equivalent of (4) in the sketch of the proof for the regular case given above.

Definition 13 *For a distribution x on V define*

$$\bar{D}(x) = \max_{(X,Y):X \propto x,(X,Y)\in E} D^*(X|Y||\eta).$$

This is the equivalent of (5) in the regular case. We will use the following lemma:

Lemma 14 *Let x be a distribution on V. Sort elements of x so that $x_1 \geq x_2 \geq \ldots \geq x_n$. Then for any $j, 1 \leq j \leq n$ one has*

$$\bar{D}(x) \leq \left(\sum_{i<j} x_i \right) \log \frac{1}{dx_j}.$$

Lemma 15 *(Bernstein's inequality) Let $X = \sum_{i=1}^n X_i$, where $|X_i| \leq M$ almost surely. Then*

$$\mathbf{Pr}\left[X - \mathbf{E}[X] < \epsilon \mathbf{E}[X]\right] < \exp\left(-\frac{\epsilon^2 \mathbf{E}[X]^2}{\sum_{i=1}^n \mathbf{E}X_i^2 + \epsilon M \mathbf{E}[X]/3} \right)$$

We now prove the main lemma in the analysis:

Lemma 16 *If our sample underestimates by a factor larger than $1-\epsilon$ with probability larger than $1-n^{-4}$, then for every j between 1 and $l-1$ we have*

$$\bar{D}(\alpha(j,\cdot)) \leq -\eta \log(\epsilon^2(1-\eta)s/4)$$

for some $\eta \in [\epsilon/4, \epsilon/2]$.

Proof:

Let $x_1 \geq \ldots \geq x_n$, $\sum_i x_i = 1$ be the distribution $\alpha(j,\cdot)$ (we numbered vertices of G). Let $X_i = \mathbf{Ber}(s/d, x_i)$. We consider two cases. First suppose that $x_1 \geq \epsilon/4$. Then we have $\bar{D}(x) \leq -\epsilon/4 \log d$ and we are done. We now assume that $x_1 \leq \epsilon/4$.

Denote by x^ϵ the subsequence $x_{j^\epsilon} \geq \ldots \geq x_n$ such that $\sum_{i=j^\epsilon}^n x_i \leq 1 - \epsilon/4$, where j^ϵ is the maximum possible. Since $x_1 \leq \epsilon/4$, we have that $\sum_{i=j^\epsilon}^n x_i \geq 1 - \epsilon/2$.

By Bernstein's inequality

$$\begin{aligned}\mathbf{Pr}\left[X^\epsilon - \mathbf{E}[X^\epsilon] < \delta \mathbf{E}[X^\epsilon]\right] &< \exp\left(-\frac{\delta^2 (s\log n/d)^2}{(s\log n/d)||x^\epsilon||_2^2 + x_{j^\epsilon}(s\log n/3d)} \right) \\ &= \exp\left(-\frac{\delta^2 \log n}{(d/s)||x^\epsilon||_2^2 + x_{j^\epsilon}(d/3s)} \right)\end{aligned}$$

Thus, if our sampling underestimates by a factor larger than $1-\epsilon$, we must have that

$$||x^\epsilon||_2^2 + x_{j^\epsilon}/3 \geq \delta^2 s/(4d).$$

Let $\eta := 1 - \sum_{i \geq j^\epsilon} x_i$. We have that Since $||x^\epsilon||^2 \leq (1-\eta)x_{j^\epsilon}$, we have $||x^\epsilon||^2 + x_{j^\epsilon}/3 \leq (4/3-\eta)x_{j^\epsilon}$, so $||x^\epsilon||^2 + x_{j^\epsilon}/3 \geq \delta^2 s/(4d)$ implies that $x_{j^\epsilon} \geq \delta^2(s/d)/(16/3 - 4\eta)$. Thus, by Lemma 14 we have

$$\bar{D}(x) \leq -\eta \log(\epsilon^2(1-\eta)s/4),$$

where we chose $\delta = \epsilon/4$. ■

It now follows by Claim 11, Lemma 16 and Lemma 12 that if a pair u, v is underestimated by more than a factor of $1 - \epsilon$ with probability at least $1 - n^{-4}$, then

$$P^l_{uv} \leq \frac{1}{d}(\epsilon^2(1-\epsilon/2)s/4)^{-(\epsilon/4)(l-1)}. \tag{8}$$

It remains to note that with probability at least $1 - n^{-2}$ one has $\hat{p}^{min}_j \leq (1+\epsilon)P^l_{uv}$ for all u, v. Since by the previous estimate with probability at least $1-1/n$ for each u, v that satisfy (8) one has $\hat{p}_{t,j} \geq (1-\epsilon)P^l_{uv}$, we have proved Theorem 1, which is the main result of this section.

2.2 Lower bounds

The following theorem, proved in Appendix A, shows that the upper bound given by Algorithm 2 is of the right form:

Theorem 17 *Any algorithm that gives a constant factor approximation to $P^l_{uv} \geq \frac{1}{d_u}(s/2)^{-(l-1)}$ in an undirected graph $G = (V, E)$ with minimum degree d must use $\Omega\left(\frac{sn^2}{l^2 d}\right)$ space.*

Also, the ϵ-factor in the exponent is not an artifact of our analysis:

Lemma 18 *For any $\epsilon < 1/2$ there exist graph $G = (V, E)$ with minimum degree d on which Algorithm 2 underestimates P^l_{uv} by more than $1 - \epsilon$ factor for pairs (u, v) such that $P^l_{uv} \geq \frac{1}{d_u}(s/2)^{-\epsilon(l-1)}$*

Proof: Consider $(l-1)/\epsilon$ sets $V_{i,j}$, $i = 1, \ldots, 1/\epsilon, j = 1, \ldots, (l-1)$, $l-1$ sets V^*_j, $j = 1, \ldots, (l-1)$ and two special vertices u and v. The vertex set of the graph G will be $V = \{u, v\} \bigcup_{j=1}^{l-1} V^*_j \cup \left(\bigcup_{i=1}^{1/\epsilon} V_{i,j}\right)$. A set V_{ij} contains one vertex if $i + j \equiv 2 (\mathrm{mod}\ \lfloor 1/\epsilon \rfloor)$ and contains d/s vertices otherwise (see Fig. 1). For each $j = 1, \ldots, l-2$ and all i the nodes in $V_{i,j} \cup V_{i,j+1}$ form a bipartite clique, and the remaining $d - d/s$ edges from each node in $V_{i,j}$ go to V^*_j. Note that a uniform sample misses the nodes in V_{ij} for $i+j \equiv 2 (\mathrm{mod}\ \lfloor 1/\epsilon \rfloor)$ with a constant probability, which can be boosted by introducing many pairs $u, v \in V$. Thus, the uniform sampling algorithm does not give a better than $1 - \epsilon$ approximation. However, one sees that $P^l_{uv} \geq \frac{1}{d_u}(s/2)^{-\epsilon(l-1)}$.

■

3 Bicriteria approximation for undirected graphs

We now give an algorithm for approximating P^l_{uv} for undirected graphs based on the techniques of [29]. The algorithm is based on the idea of sampling a hierarchy of centers and approximating transition probabilities by probabilities of going through the centers. However, unlike in Algorithm 2, one cannot guarantee concentration of the sample around its mean since we would like to use close to linear amount of space. Another notion of approximation is still possible: we show that our estimate never underestimates P^l_{uv} and at the same time cannot exceed a quantity that depends on the transition probability P^{jl}_{uv}, where $1 \leq j \leq (2k-1)l$.

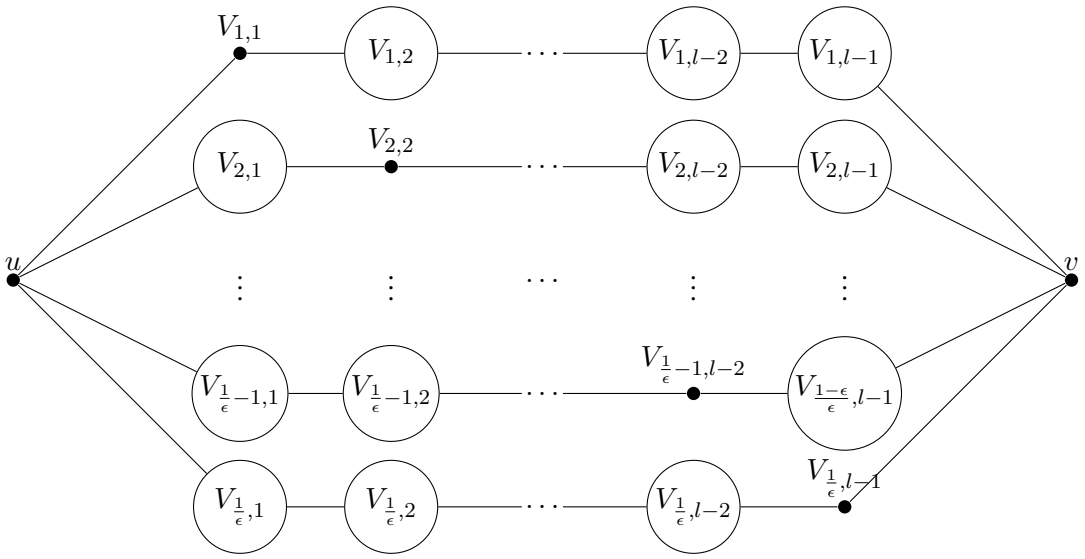

Figure 1: Tight example for algorithm analysis (the sets V_j^* are not shown)

We start by giving intuition about the algorithm. The algorithm chooses a hierarchy of centers $A_i, i = 0, \ldots, k$, where $A_0 = V$ and A_{i+1} is obtained by sampling elements of A_i independently with probability $n^{-1/k}$, so that $\mathbf{E}[A_i] = n^{1-i/k}$. For each vertex $u \in V$ and for each $j = 0, \ldots, k$, the algorithm stores the vertices in A_j that are more likely to be reached in jl steps from u than any vertex in A_{j+1}. When $P^l_{u,v}$ is queried, the algorithm attempts to find a center $c \in A_j$ for some j such that both $P^{(j-1)l}_{u,c}$ and $P^{jl}_{c,v}$ are available. The search for the centers is organized in a way that ensures that for the center c^* that is found the probability of reaching v from u in $(2j-1)l$ steps via c^* can be related to $P^l_{u,v}$ (the left inequality in Theorem 8). This is achieved via methods similar to [29].

As noted in section 2, choosing random centers and approximating the probability of a $(2j-1)l$-step transition from u to v by the sum of $P^{(j-1)l}_{u,c} P^{jl}_{c,v}$ over centers c is exact in expectation. However, the lack of concentration as well as the fact that we only use one center for approximation does not allow us to conclude that out estimate is within a constant multiplicative factor of the true probability of $(2j-1)l$-step transition. However, with some additional care we ensure that our estimate does not exceed the expectation significantly, thus yielding the right inequality in Theorem 8.

We first consider the case of graphs with vertex degrees differing by at most a constant factor, i.e. the case in which our lower bounds hold, and then extend the results to a symmetrized transition probability for general graphs.

We now describe the algorithm. Let $A^s_0 = V, s = 0, \ldots, 6k^2 \log n$ and let A^s_j be obtained by sampling elements of A^s_{j-1} with probability $n^{-1/k}$, $j = 1, \ldots, k$. For a vertex $u \in V$ and a set $S \subseteq V$ we define $P^{jl}_{u,S} = \max_{s \in S} P^{jl}_{u,s}$. For a vertex u define $B^s_j(u)$ to be

$$B^s_j(u) = \{v \in A^s_j : P^{(j+1)l}_{u,v} > P^{(j+1)l}_{u,A^s_{j+1}}\}.$$

Also, let $p^s_j(u) := v$ such that $P_{jl}(u,v) = P_{jl}(u, A^s_{j+1})$. We prove the following

Theorem 8 *Let $G = (V, E)$ be a weighted undirected graph such that $1/\gamma \leq d(u)/d(v) \leq \gamma$ for all $u, v \in V$. There exists an $O(k^3 n^{1+1/k} \log n)$ space and $O(k^3 \log n)$ query time oracle that given a pair*

$u, v \in V$ *outputs a value* $\hat{p}$ *such that*

$$\gamma^{-1} P_{u,v}^{l} \leq \hat{p} \leq 4 \max_{1 \leq j \leq k} \left(P_{u,v}^{(2j-1)l} / n^{(j-1)/k} \right)^{1/(2j-1)}.$$

We note that the same approximation guarantees hold for the symmetrized quantity $S_{uv}^{l} = \sqrt{P_{uv}^{l} P_{vu}^{l}}$ (see Appendix B).

References

[1] K. Avrachenkov, N. Litvak, D. Nemirovsky, and N. Osipova. Monte carlo methods in pagerank computation: When one iteration is sufficient. *SIAM J. Numer. Anal.*, 45, 2007.

[2] B. Bahmani, A. Goel, and A. Chowdhury. Fast incremental and personalized pagerank. *http://arxiv.org/abs/1006.2880*, 2010.

[3] S. Baswana. Streaming algorithm for graph spanners - single pass and constant processing time per edge. *Inf. Process. Lett.*, 106:110114, 2008.

[4] S. Baswana and S. Sen. A simple linear time algorithm for computing $(2k-1)$-spanner of $o(n^{1+1/k})$ size for weighted graphs. *ICALP*, 2003.

[5] P. Berkhin. A survey on pagerank computing. *Internet Mathematics*, 2, 2005.

[6] K. Clarkson and D. Woodruff. Numerical linear algebra in the streaming model. *STOC*, 2009.

[7] D. Cohen. Fast algorithms for constructing t-spanners and paths with stretch t. *SIAM Journal on Computing*, 28:210236, 1999.

[8] D. Dor and U. Halperin, S.and Zwick. All pairs almost shortest paths. *SIAM Journal on Computing*, 29, 2000.

[9] P. Drineas and R. Kannan. Fast monte carlo algorithms for approximate matrix multiplication. *FOCS*, 2001.

[10] P. Drineas, R. Kannan, and M. Mahoney. Fast monte carlo algorithms for matrices i: Approximating matrix multiplication. *SIAM J. Computing*, 36, 2006.

[11] M.L. Elkin and D. Peleg. $(1+\epsilon, \beta)$-spanner constructions for general graphs. *STOC*, 2001.

[12] Joan Feigenbaum, Sampath Kannan, Andrew McGregor, Siddharth Suri, and Jian Zhang. Graph distances in the streaming model: the value of space. *SODA*, 2005.

[13] A. Firat, S. Chatterjee, and M. Yilmaz. Genetic clustering of social networks using random walks. *Computational Statistics & Data Analysis*, 51, 2007.

[14] F. Fouss, A. Pirotte, J.-M. Renders, and M. Saerens. Random-walk computation of similarities between nodes of a graph with application to collaborative recommendation. *IEEE Transactions on Knowledge and Data Engineering*, 19, 2007.

[15] A. Frieze, R. Kannan, and S. Vempala. Fast monte-carlo algorithms for finding low-rank approximations. *JACM*, 2004.

[16] R. E. Gomory and T.C. Hu. Multi-terminal network flows. *J. Soc. Indust. Appl. Math.*, 1961.

[17] G. Jeh and J. Widom. Scaling personalized web search. *WWW*, 2003.

[18] S. D. Kamvar, T. H. Haveliwala, C. D. Manning, and G. H. Golub. Extrapolation methods for accelerating pagerank computations. *WWW*, 2003.

[19] A. N. Langville and C. D. Meyer. Deeper inside pagerank. *Internet Mathematics*, 1, 2004.

[20] A. Magen and A. Zouzias. Low rank matrix-valued chernoff bounds and approximate matrix multiplication. *SODA*, 2011.

[21] N. H. Nguyen, T. T. Do, and T. D. Tran. A fast and efficient algorithm for low-rank approximation of a matrix. *STOC*, 2009.

[22] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. *Technical report, Stanford InfoLab*, 1999.

[23] M. Patrascu and L. Roditty. Distance oracles beyond the thorup-zwick bound. *FOCS*, 2010.

[24] J. Rutten, M. Kwiatkowska, G. Norman, and D. Parker. *Mathematical Techniques for Analyzing Concurrent and Probabilistic Systems,* P. Panangaden and F. van Breugel (eds.), volume 23 of *CRM Monograph Series*. American Mathematical Society, 2004.

[25] T. Sarlos. Improved approximation algorithms for large matrices via random projections. *FOCS*, 2006.

[26] T. Sarlós, A. A. Benczúr, K. Csalogány, D. Fogaras, and B. Rácz. To randomize or not to randomize: space optimal summaries for hyperlink analysis. *WWW*, 2006.

[27] A. Das Sarma, S. Gollapudi, and R. Panigrahy. Estimating pagerank on graph streams. *PODS*, 2008.

[28] D.A. Spielman and N. Srivastava. Graph sparsification by effective resistances. *STOC*, pages 563–568, 2008.

[29] M. Thorup and U. Zwick. Approximate distance oracles. *STOC*, 2001.

[30] M. Thorup and U. Zwick. Spanners and emulators with sublinear distance errors. *SODA*, 2006.

[31] D. Woodruff. Lower bounds for additive spanners, emulators and more. *FOCS*, 2006.

A Proofs from section 2

We now give
Proof of Lemma 12: The proof is by induction on l.

Base:$l = 2$ Since $X_0 = u$, $D^*(X_1|X_0||\eta) = D(X_1||\eta^1)$.

Inductive step:$l \to l+1$ First note that for any $j > 0$ for any $x \in V^{j-2}, x', y \in V$ one has

$$\eta^j(x, x', y) = \eta(x')\eta^{j-1}(x, x').$$

Using this fact, we get

$$\begin{aligned}
D(X_1,\ldots,X_{l-1}||\eta^l) &= \sum_{x\in V^{l-3},x',y\in V} p(x,x',y)\log((\eta(x')\eta^{l-1}(x,x'))/p(x,x',y)) \\
&= \sum_{x'\in V} p(x') \sum_{x\in V^{l-3}} p(x|x') \sum_{y\in V} p(y|x,x') \\
&\quad \Big[\log(\eta(x')/p(y|x,x')) + \log(\eta^{l-1}(x,x')/p(x,x'))\Big] \\
&= \sum_{x'\in V} p(x')\log\eta(x') \\
&\quad + \sum_{x'\in V} p(x') \sum_{x\in V^{l-3}} p(x|x') \left[\sum_{y\in V} p(y|x,x')\log(1/p(y|x,x'))\right] \\
&\quad + \sum_{x'\in V,x\in V^{l-3}} p(x,x')\log(\eta^{l-1}(x,x')/p(x,x')) \\
&= S_1 + S_2 + S_3.
\end{aligned}$$

By convexity of the entropy function, we have

$$S_2 \le \sum_{x'\in V} p(x') \sum_{y\in V} p(y|x')\log(1/p(y|x')),$$

so

$$S_1 + S_2 \le D^*(X_{l-1}|X_{l-2}||\eta).$$

By the inductive hypothesis, $S_3 \le \sum_{j=1}^{l-2} D^*(X_j|X_{j-1}||\eta)$, so

$$D((X_1,\ldots,X_{l-1})||\eta^l) \le \sum_{j=1}^{l-1} D^*(X_j|X_{j-1}||\eta).$$

■

Proof of Lemma 14: Consider any pair $(X,Y), X \propto x, (X,Y) \in E$. Denote the distribution by $p(x,y)$. Denote the set of vertices corresponding to $x_i, i \ge j$ by I_j. For a set $S \subset V$ we will write $p(S)$ to denote the probability assigned to S by x.

We have

$$\begin{aligned}
D^*(X|Y||\eta) &= \sum_y p(y) \sum_x p(x|y)\log(1/(p(x|y)d_y)) \\
&\le \sum_y p(y) \sum_{x\in I_j} p(x|y)\log(1/(p(x|y)d)) \\
&\quad + \sum_y p(y) \sum_{x\in V\setminus I_j} p(x|y)\log(1/(p(x|y)d_y)) \\
&\le \sum_y p(y) \sum_{x\in I_j} p(x|y)\log(1/p(x|y)) + p(I_j)\log(1/d) \\
&\le p(I_j)H(x|_{I_j}) + p(I_j)\log(1/d) \le p(I_j)\log(1/(dx_j)),
\end{aligned}$$

where we used the convexity of the entropy function and the fact that all out-degrees are at least d. ■

Proof of Corollary 5: Set $s = n^\epsilon$. We need to calculate l-step transition probabilities from $\tilde{O}(n^{1+\epsilon}/d)$ nodes. This takes $O(nd)$ time per node. After doing that we end up with two matrices X and Y of dimension $n \times n^{1+\epsilon}/d$ and $n^{1+\epsilon}/d \times n$ respectively. We need to evaluate XY. To do so, we write

$$X = [X_1 \dots X_k]$$
$$Y = [Y_1 \dots Y_k]^T,$$

where $k = O(d/n^\epsilon)$ and each of X_j, Y_j is $(n^{1+\epsilon}/d) \times (n^{1+\epsilon}/d)$. Multiplying $X_i Y_j$ takes $O(n^{(1+\epsilon)\omega}/d^\omega)$ times. Since there are $(d/n^{1+\epsilon})^2$ such pairs, the total time taken is

$$\left(\frac{d}{n^\epsilon}\right)^2 (n^{(1+\epsilon)\omega}/d^\omega) = n^{(1+\epsilon)\omega-2\epsilon}/d^{\omega-2}$$

This works well when d is large. When d is small, sparse matrix-vector products can be used to calculate the distribution in $O(n^2 d)$ time. Choosing the better of the two, we obtain $O\left(n^{2+(1+\epsilon)(\omega-2)/(\omega-1)}\right)$ runtime. ■

Proof of Theorem 17: Construct the graph $G = (V, E)$ as follows. Let

$$V = V_0 \cup \left(\bigcup_{j=1}^{l-1} V_l^* \cup \left(\bigcup_{i=1}^{sn/d} V_{ij}\right)\right),$$

where $|V_{ij}| = d/s$, $|V_j^*| = d$ and $|V_0| = n$. Thus, the graph G has $O(nl)$ vertices.

$V_{i,j-1} \cup V_{i,j}$ form a bipartite clique for each i and j. For each j each vertex in V_{ij} is connected to $d - d/s$ nodes in V_j^*. Finally, for each $u \in V_0$ and $i = 1, \dots, sn/d$ a fair coin is flipped to determine if u is connected to all vertices in V_{i1} or to none.

Now choose $u \in V_{il}$ and $v \in V_0$. All l-step paths go through vertices of $V_{i,l}, V_{i,l-1}, \dots, V_{i1}$ in that order, and the probability of reaching u from v is $\frac{1}{d_u}(s/2)^{-(l-1)}$ if V_{i1} is connected to v and 0 otherwise. Thus, a constant factor approximation allows one to determine presence of edges between V_0 and V_{i1}, and hence necessarily takes $\Omega(sn'^2/l^2 d)$ space, where $n' = nl$ is an upper bound on the number of vertices in G. ■

B Analysis of Algorithm 4

B.1 Lower bounds

We show that the space used by Algorith 4 is optimal in the sense that any algorithm that provides the guarantees of Theorem 8 necessarily uses $\Omega(n^{1+1/k})$ space.

Our lower bounds are based on the following observation. Consider a random $G_{n,p}$ graph with a path $P_v = (u_1, \dots, u_{l-1})$ attached to every vertex v, which we identify with u_l (the paths belonging to different vertices are disjoint). It can be shown that for any vertex w that is connected to u_l by an edge the probability of reaching v from u_1 in j steps for any j between l and kl is significantly larger than the probability of reaching a vertex w that is not connected to u_l by a direct edge. This is because the probability of reaching w without leaving the set of vertices $\{u_0, \dots, u_l, w\}$ dominates the probability of reaching w by going through vertices of the $G_{n,p}$ graph other than u_l and w since the random walk on $G_{n,p}$ mixes fast. Thus, shortest paths are the most probable paths in this setting. In what follows we make this intuition precise.

We start by proving Lemma 7:

Proof of Lemma 7: Consider a family of graphs $G = (V, E)$ defined as follows. Let $V = \bigcup_{j=1}^{l} V_j$, where $|V_j| = n$. There is a matching between the nodes of V_j and V_{j+1} for $j = 1, \ldots, l-1$, each edge having weight n, and a random graph with each edge appearing independently with probability p on the vertices of V_l. Edges with both endpoints in V_l have weight 1. Note that whp the degrees of vertices of G differ by at most a factor of $(2p+1)/(p/2)$, i.e. G is γ-regular with $\gamma = 4 + 2/p$ whp.

For a vertex $u_1 \in V_1$ let u_j be the vertex in V_j that is matched to u_{j-1}, for $2 \leq j \leq l$ For a vertex $v \in V_l, v \neq u_l$ we will estimate $P^{kl}_{u_1,v}$ in two cases: (1) $(u_l, v) \in E$ and $(u_l, v) \notin E$.

1. $(u_l, v) \in E$ In this case one has $P^{lk-1}(u_1, u_l) \geq 2^{-lk+1}$. Hence, whp

$$P^{lk}(u_1, v) \geq 2^{-lk+1} \frac{p/2}{(1+2p)} \frac{1}{pn} = 2^{-lk+1} \frac{1}{(2+4p)n}.$$

2. $(u_l, v) \notin E$ One has whp $P^{lk}_{u_1,v} \leq \frac{2p}{1+p/2} \max_{w \in V} P^2_{w,v} = \max_{w \in V_l} P^2_{w,v}$, where the term $\frac{2p}{1+p/2}$ appears due to the transition of the walk out of u_l, and the maximum can be taken over nodes in V_l since $(u_l, v) \notin E$).

We now upper bound $P^2_{w,v}$. Fix $w \in V_l$. The expected number of common neighbors of w and v is np^2, and is at most $2np^2$ with high probability. Hence, with high probability $P^2_{w,v} \leq 2np^2 \frac{1}{(np)^2}$.

Finally, we get that with high probability one has $P^{lk}_{u_1,v} \leq \frac{4p}{(1+p/2)n}$.

Now suppose that an algorithm A outputs an estimate $\hat{p}$ such that

$$P^l_{u,v} \leq \hat{p} \leq (1+\epsilon) \max_{1 \leq j \leq k} P^{jl}_{u,v}$$

for some $\epsilon > 0$. By the estimates above one has that if $(u_l, v) \in E$, then $P^l_{u_1,v} \geq 2^{-lk+1} \frac{1}{(2+4p)n}$, and when $(u_l, v) \notin E$, then $P^{kl}_{u_1,v} \leq 4p/(1+p/2)n$. Hence, setting $p = \Omega(2^{-lk})$, we get that the output of the algorithm can be used to determine the presence of any edge $(u, v), u, v \in V_l$. Hence, the algorithm must use $\Omega(pn^2) = \Omega(n^2/2^{-lk})$ space.

■

We now give

Proof of Theorem 10: We use the same construction as in the proof of Lemma 7 with $p = \alpha n^{-1+1/k}$, where $\alpha > 0$ is a parameter that will be chosen later. The weight of edges in the attached paths is $\alpha n^{1/k}$. Using the same arguments as before, we have $P^{lk}_{u_1,v} \geq 2^{-lk+1} \frac{1}{2\alpha n^{1/k}}$ whp.

Also, we have that

$$P^{lk}_{u_1,v} \leq \max_{1 < j \leq k} P^j_{u_l,v}.$$

Fix $w, v \in V_l$. Consider a j-step walk for $2 \leq j \leq k$ that traverses $2 \leq s \leq j$ distinct edges $W \in \mathcal{W}_{s,j}, W = (w = w_0, w_1, \ldots, w_j = v)$ *and does not use the edge* (u_l, v). The size of $\mathcal{W}_{s,j}$ is at most $j^{2(j-s)} n^{s-1}$, and the probability that all edges of $W \in \mathcal{W}_{s,j}$ are present is $\alpha^s n^{(-1+1/k)s}$. The probability of taking such a walk is at most $2\alpha^{-j}/n$ with high probability (since the degree of every node is within a factor of $(1 \pm \ln 2/lk)$ of expectation with high probability). Hence, with high probability over the graph one has for all j

$$\mathbf{E}\left[\max_{1<j\leq k} P^j_{u_l,v}\right] \leq k \sum_{s=1}^{j} |\mathcal{W}_{s,j}| \alpha^s n^{-(1-1/k)s} \alpha^{-j} (2/n)$$

$$\leq k \sum_{s=1}^{j} j^{2(j-s)} n^{s-1} \alpha^s n^{-(1-1/k)s} (2/n) = k \sum_{s=1}^{j} \alpha^s j^{2(j-s)} n^{-1+s/k} \alpha^{-j} (2/n) \leq 3k/n$$

By Markov's inequality and the fact that the probability that we bounded is independent of the presence or absence of a direct edge (since the edge (u_l, v) was excluded), at least for half of the graphs and for at least half of the pairs (u_0, v) such that $(u_l, v) \in E$ one has $\max_{1<j\le k} P^j(u_l, v) \le 12k/n$.

Now consider a pair (u_1, v) such that $(u_l, v) \in E$. Then $P_l(u, v) \ge 2^{-lk+1}\frac{1}{2\alpha n^{1/k}}$, so A outputs $\hat{p} \ge 2^{-lk+1}\frac{1}{2\alpha n^{1/k}}$. On the other hand, for a pair (u_1, v) such that $(u_l, v) \notin E$ one has $P^l_{u_1,v} = 0$, while $P^{lj}_{u_1,v} \le 12k/n$, so we have

$$\max_{1\le j\le k}\left(P^{jl}_{u,v}/n^{(j-1)/k}\right)^{1/(2j-1)} \le 12k \max_{1\le j\le k}\left(n^{-1-(j-1)/k}\right)^{1/(2j-1)}$$

$$= 12k \max_{1\le j\le k}\left(n^{-1-(j-1)/k}\right)^{1/(2j-1)} = 12k \max_{1\le j\le k} n^{-(1/k)(k+j-1)/(2j-1)} = 12kn^{-1/k}.$$

Thus, algorithm A outputs $\hat{p} \le 12Ckn^{-1/k}$, so after choosing a sufficiently small constant $\alpha = \Omega(2^{-lk})$, we get that the presence or absence of an edge between at least half of pairs of vertices can be determined from the output of A for at least half of the graphs. Thus, A must use $\Omega(n^{1+1/k}/2^{lk})$ space. ■

B.2 Upper bound

for $s \leftarrow 1$ to $6k^2 \log n$ **do**
  $A_0^s := V$
  for $j = 1$ to k **do**
    Let A_j^s contain each element of A_{j-1}^s independently with probability $n^{-1/k}$
  end for
  for every $v \in V$ and $j \leq k$ **do**
    $p_j^s(v) \leftarrow \text{argmax}_{w \in A_{j+1}^s} P_{v,w}^{(j+1)l}$.
    Let $B_j^s(u) \leftarrow \{v \in A_j^s : P_{u,v}^{(j+1)l} > P_{u,A_{j+1}^s}^{(j+1)l}\}$.
  end for
end for

Algorithm 3: Preprocessing algorithm

for $i \leftarrow 1$ to k **do**
  $Q_i \leftarrow \emptyset$, $R_i \leftarrow \emptyset$
  for $s \leftarrow 1$ to $6k^2 \log n$ **do**
    $q_{si} \leftarrow \max_{w \in A_i^s} P_{u,w}^{il} \cdot P_{w,v}^{(i+1)l}$.
    $R_i \leftarrow R_i \cup \{q_{si}\}$
  end for
  $z_i \leftarrow (6k^2 \log n/(4i^2))$-th largest element in R_i.
end for
for $s \leftarrow 1$ to $6k^2 \log n$ **do**
  $w \leftarrow v, i \leftarrow 0$
  while $w \notin B_i^s(u)$ **do**
    $w \leftarrow p_i^s(u)$
    $(u, v) \leftarrow (v, u)$
    $i \leftarrow i + 1$
  end while
  $Q_i \leftarrow Q_i \cup \{P_{u,w}^{(i+1)l} \cdot P_{w,v}^{(i+2)l}\}$
end for
for $i \leftarrow 0$ to $k - 1$ **do**
  $Q_i^* \leftarrow \{x \in Q_i | x \leq z_i\}$
  if $Q_i^* \neq \emptyset$, let $r \leftarrow \min\{r_s \in Q_i^*\}$ and return $\hat{p} \leftarrow r^{1/(2i-1)}$.
end for

Algorithm 4: Probability estimation

In this section we give the analysis of Algorithm 4 omitted in section 3. We first consider γ-regular graphs, and then give an approximation of the symmetrized quantity $S_{u,v}^l = \sqrt{P_{uv}^l P_{vu}^l}$.

We first bound the space used by the data structure:

Lemma 19 *The expected space required by Algorithm 4 is* $O(k^3 n^{1+1/k} \log n)$.

Proof: The size of the list stored at each vertex is bounded by $n^{1/k}$ in expectation for each $j = 0, \ldots, k$. To show this, we fix u and consider the list of vertices $v \in A_j$ in decreasing order of the probability $P_{u,v}^{(j+1)l}$. Each vertex $v \in A_j$ is included in A_{j+1} with probability $n^{-1/k}$. Thus the size of the list is dominated by

a geometric random variable with rate $1 - n^{-1/k}$. Since $6k^2 \log n$ such lists are stored at each vertex, the bound follows. ■

The proof of Theorem 8 follows from the following lemmas.

Lemma 20 *Let $G = (V, E)$ be a γ-regular graph. Let $u, v \in V$. Let $\hat{p}$ be the output of the algorithm on (u, v). Then*

$$\hat{p} \geq P^l_{u,v}/\gamma.$$

Proof: Fix s and consider the **while** loop in the estimation algorithm (lines 11–15). We show by induction on $i = 0, \ldots, k$ that a **while** loop invariant is

$$P^{(i+1)l}_{u,w} \geq (P^l_{u,v})^{\lceil (i+1)/2 \rceil} (P^l_{v,u})^{\lfloor (i+1)/2 \rfloor}.$$

Base: $i = 0$ $P^{(i+1)l}_{u,w} \geq (P^l_{u,v})^{\lceil (i+1)/2 \rceil} (P^l_{v,u})^{\lfloor (i+1)/2 \rfloor} = P^l_{u,v}$ since $w = v$.

Inductive step: $i \to i + 1$ Since $w \notin B^s_i(u)$, one has $P^{(i+1)l}_{u,p^s_i(u)} \geq P^{(i+1)l}_{u,w}$. Hence,

$$P^{(i+2)l}_{v,p^s_i(u)} \geq P^l_{v,u} P^{(i+1)l}_{u,p^s_i(u)} \geq P^l_{u,v} P^{(i+1)l}_{u,w}.$$

By the inductive hypothesis, $P^{(i+1)l}_{u,w} \geq (P^l_{u,v})^{\lceil (i+1)/2 \rceil} (P^l_{v,u})^{\lfloor (i+1)/2 \rfloor}$, so

$$P^{(i+2)l}_{v,p^s_i(u)} \geq (P^l_{u,v})^{\lceil (i+1)/2 \rceil} (P^l_{v,u})^{\lfloor (i+1)/2 \rfloor + 1}.$$

Upon setting $w \leftarrow p^s_i(u)$ and switching u and v, we get

$$P^{(i+2)l}_{u,w} \geq (P^l_{v,u})^{\lceil (i+1)/2 \rceil} (P^l_{u,v})^{\lfloor (i+1)/2 \rfloor + 1} = (P^l_{v,u})^{\lfloor (i+2)/2 \rfloor} (P^l_{u,v})^{\lceil (i+2)/2 \rceil}.$$

since $\lfloor j/2 \rfloor + 1 = \lceil (j+1)/2 \rceil$ and $\lceil j/2 \rceil = \lfloor (j+1)/2 \rfloor$ for all integer j.

We can now finish the proof of the lemma. We get that if $w \in B^s_i(u)$,

$$P^{(i+1)l}_{u,w} \geq (P^l_{u,v})^{\lceil (i+1)/2 \rceil} (P^l_{v,u})^{\lfloor (i+1)/2 \rfloor}$$

and

$$P^{il}_{v,w} \geq (P^l_{v,u})^{\lceil i/2 \rceil} (P^l_{u,v})^{\lfloor i/2 \rfloor}.$$

Let u_0, v_0 be the queried pair of vertices. If i is even, then $u = u_0, v = v_0$, and we get

$$\begin{aligned} P^{(i+1)l}_{u,w} \cdot P^{il}_{w,v} &\geq \gamma^{-1} (P^l_{u_0,v_0})^{\lceil (i+1)/2 \rceil + \lfloor i/2 \rfloor} (P^l_{v_0,u_0})^{\lfloor (i+1)/2 \rfloor + \lceil i/2 \rceil} \\ &\geq \gamma^{-(i+1)} (P^l_{u_0,v_0})^{2i+1} \end{aligned}$$

If i is odd, then $u = v_0, v = u_0$, and we get

$$\begin{aligned} P^{(i+1)l}_{u,w} \cdot P^{il}_{w,v} &\geq \gamma^{-1} (P^l_{v_0,u_0})^{\lceil (i+1)/2 \rceil + \lfloor i/2 \rfloor} (P^l_{u_0,v_0})^{\lfloor (i+1)/2 \rfloor + \lceil i/2 \rceil} \\ &\geq \gamma^{-(i+2)} (P^l_{u_0,v_0})^{2i+1} \end{aligned}$$

Hence, we have

$$\left(P^{(i+1)l}_{u,w} \cdot P^{il}_{w,v} \right)^{1/(2i+1)} \geq \gamma^{-1} P^l_{u_0,v_0}.$$

■

Lemma 21 *Let $G = (V, E)$ be a γ-regular graph. Let $u, v \in V$. Denote by $\hat{p}$ the output of the algorithm on (u, v). Then*

$$\hat{p} \leq 4 \max_{1 \leq j \leq k} \left(P_{u,v}^{(2j-1)l} / n^{(j-1)/k} \right)^{1/(2j-1)}.$$

Proof: Suppose that the algorithm outputs $\hat{p} = r^{1/(2j-1)}$, where $r \in Q_j$.

First note that since

$$\sum_{c \in V} P_{u,c}^{(j-1)l} P_{c,v}^{jl} = P_{u,v}^{(2j-1)l},$$

we have

$$\mathbf{E}\left[\sum_{c \in A_j} P_{u,c}^{(i-1)l} P_{c,v}^{il} \right] = P_{u,v}^{(2j-1)l} / n^{(j-1)/k},$$

and in particular,

$$\mathbf{E}\left[\max_{c \in A_j} P_{u,c}^{(i-1)l} P_{c,v}^{il} \right] \leq P_{u,v}^{(2j-1)l} / n^{(j-1)/k}.$$

Let $c_j^s := \text{argmax}_{c \in A_j^s} P_{u,c}^{(i-1)l} P_{c,v}^{il}$. Let

$$X_{js} = \begin{cases} 1 & \text{if } P_{u,c_j^s}^{(j-1)l} P_{c_j^s,v}^{jl} > 4j^2 P_{u,v}^{(2j-1)l} / n^{(j-1)/k} \\ 0 & \text{o.w.} \end{cases}$$

Note that X_{js} are independent for fixed j and $\mathbf{E}[X_{js}] \leq 1/(4j^2)$ by Markov's inequality, and for each j we have by the Chernoff bound that

$$\mathbf{Pr}\left[\sum_{s=1}^{6k \log n} X_{js} \geq 6k^2 \log n / (2j^2) \right] \leq n^{-3}$$

Hence, with probability at least $1 - n^{-3}$ one has that $\sum_{j=1}^{k} \sum_{s=1}^{6k \log n} X_{js} < 3k^2 \log n \sum_{j \geq 1} 1/j^2 < 6k^2 \log n$. Thus, with probability at least $1 - 1/n$ for every pair of vertices there exists i_0 such that $Q_{i_0}^* \neq \emptyset$. Then for all $r \in Q_{i_0}^*$ one has

$$r \leq 4i^2 \mathbf{E}\left[\sum_{c \in A_j} P_{u,c}^{(i-1)l} P_{c,v}^{il} \right] = 4i^2 P_{u,v}^{(2i-1)l} / n^{(i-1)/k}.$$

Hence,

$$\hat{p} = r^{1/(2i-1)} \leq \left(4i^2 P_{u,v}^{(2i-1)l} / n^{(i-1)/k} \right)^{1/(2i-1)} \leq 4 \left(P_{u,v}^{(2i-1)l} / n^{(i-1)/k} \right)^{1/(2i-1)}.$$

■

The proof of Theorem 8 follows by putting together Lemma 20 and Lemma 21. The bound of $O(k^3 \log n)$ query time follows since the algorithm performs $O(k)$ hash table lookups for each $s = 1, \ldots, 6k^2 \log n$. The preprocessing time is bounded by $\tilde{O}(kln^{\omega})$, where $\omega \geq 2$ is the matrix multiplication constant.

Finally, we show that Algorithm 4 can be used to approximate a symmetrized version of $P^l(u,v)$. Define $S^t(u,v) = \sqrt{P^t(u,v)P^t(v,u)} = \sqrt{d(v)/d(u)}P^t(u,v)$. We now check that the two properties of $P^l(u,v)$ that were crucially used in Algorithm 4 also hold for $S^l(u,v)$. The first property, used in Lemma 20 is

$$\begin{aligned} S^{(i+1)l}_{u,v} &= \sqrt{d(v)/d(u)}P^l(u,v) \geq \sqrt{d(v)/d(u)}P^l(u,c)P^{il}_{c,v} \\ &= \sqrt{d(c)/d(u)}P^l_{u,c}\sqrt{d(v)/d(c)}P^{il}_{c,v} = S^l_{u,c}S^{il}_{c,v}. \end{aligned}$$

The second property, used in Lemma 21, is

$$S^{(2i-1)l}_{u,v} = \sum_{c \in V} S^{il}_{u,c}S^{(i-1)l}_{c,v}.$$

Thus, we get the same guarantees for Algorithm 4 as we have for $P^t(u,v)$ after setting $\gamma := 1$.