Sublinear Estimation of a Single Element in Sparse Linear Systems

Nitin Shyamkumar, Siddhartha Banerjee, Peter Lofgren

***Abstract*— We present a fast *bidirectional* algorithm for estimating a *single element* of the product of a matrix power and vector. This is an important primitive in many applications; in particular, we describe how it can be used to estimate a single element in the solution of a linear system $Ax = b$, with sublinear average-case running time guarantees for sparse systems. Our work combines the von Neumann-Ulam MCMC scheme for matrix multiplication with recent developments in bidirectional algorithms for estimating random-walk metrics. In particular, given a target additive-error threshold, we show how to combine a reverse local-variational technique with forward MCMC sampling, such that the resulting algorithm is order-wise faster than each individual approach.**

I. Introduction

We present a *bidirectional algorithm* for estimating a *single element* of the product of a matrix power and a vector. Formally, given any matrix $A \in \mathbb{R}^{n\times n}$, vector $\mathbf{z}$ and *a target index* $t \in \{1, 2, \ldots, n\}$, we consider the problem of computing $\mathbf{p}^\ell_\mathbf{z}[t] := \left(A^\ell \mathbf{z}\right)[t]$ for a given exponent $\ell \in \mathbb{N}$. This is a basic primitive in many matrix algorithms, and there is a large body of work on efficient ways of doing exact computations. However, such techniques are often infeasible in large-scale settings (i.e., with very large values of n), and one needs to resort to approximate computation of $A^\ell\mathbf{z}$. In particular, we are interested in developing techniques which can estimate a single element of $A^\ell\mathbf{z}$ in time which is provably faster than computing the entire vector.

The problem of estimating $\mathbf{p}^\ell_\mathbf{z}[t]$ has gained attention recently in the context of estimating random-walk probabilities, in particular, for Personalized PageRank [1], and other network centrality metrics. A recent line of work [2], [3], [4] shows how to develop fast bidirectional algorithms for Pagerank, and more generally, for settings where A^T is a *stochastic* matrix and $\mathbf{z}$ is a probability distribution. This paper extends these techniques to more general $A, \mathbf{z}$; in particular, our main result can be paraphrased as follows:

Proposition 1: Given matrix A (with $||A||_1 \leq 1$), vector z (with $||z||_1 \leq 1$) and any index t in $[n]$, Algorithm `BIDIR-MATRIX-POWER` *(cf. Section IV) returns an estimate $\widehat{\mathbf{p}}^\ell_\mathbf{z}[t]$ such that $|\mathbf{p}^\ell_\mathbf{z}[t] - \widehat{\mathbf{p}}^\ell_\mathbf{z}[t]| < \max\{\epsilon \mathbf{p}^\ell_\mathbf{z}[t], 1/n\}$ with high probability; moreover, it has an average running time of $\widetilde{O}\left((nnz(A)/\epsilon)^{2/3}\right)$ for a uniform random choice of t.*

N. Shyamkumar is with the Department of Computer Science, and S. Banerjee is with the School of Operations Research and Information Engineering, Cornell University, USA, Email: `nhs56@cornell.edu`, `sbanerjee@cornell.edu`. P. Lofgren is with the Department of Computer Science, Stanford University, USA, Email: `plofgren@cs.stanford.edu`.

This generalizes the existing results for stochastic matrices, albeit with a loss in the running time scaling [1]. Its significance arises from the fact that computing $\left(A^\ell\mathbf{z}\right)[t]$ is often a subroutine in more complex algorithms for various matrix computations. In Section II, we highlight one such application – solving linear equations via the von Neumann-Ulam scheme [5]. This technique has been gaining interest as a promising candidate for solving large-scale equations, owing to its ease of parallelization and asynchronous and local nature [6], [7], [8]. More generally, our algorithm for estimating $\left(A^\ell\mathbf{z}\right)[t]$ may prove useful in other estimation routines which use matrix polynomials as approximators.

A. Related Work

Bidirectional algorithms for estimating transition probabilities in Markov chains were first developed for *reversible Markov chains* using random-walk collision statistics. Kale et al. [11] proposed such a technique for estimating length-2ℓ random walk transition probabilities in a *regular undirected graph*. The main idea is that to test if a random walk goes from s to t in 2ℓ steps with probability $\geq \delta$, we can generate two independent random walks of length ℓ, starting from s and t respectively, and detect if they collide. The critical observation is that using $\sqrt{n}$ walks from s and t gives n potential collisions, which is sufficient to estimate probabilities on the order of $1/n$. A bidirectional estimator in general graphs was first developed by Lofgren et al. [2] for PageRank estimation; this was subsequently simplified in [3], and extended to general Markov chains in [4]. Our work here generalizes this line of work to computing single elements of powers of arbitrary matrices.

Our algorithm, as well as those in [2], [3], [4], are all essentially based on combining local-variational techniques with MCMC sampling techniques. The specific local variational iteration we use was originally proposed for computing Personalized Page Rank (PPR) in [10]; we demonstrate how their procedure extends to general matrices in Section III. An identical iteration for general matrices was considered in [8] with different termination conditions; they provide worst-case running time guarantees, which, however, can be quite weak even in sparse matrices.

Monte Carlo sampling methods form an alternate approach for large-scale approximate matrix computations. These are widely used for estimating transition probabilities and PageRank [**?**]. Their use in matrix multiplication and inversion can be traced back to the von Neumann-Ulam

[1]For ℓ-step transition probabilities, the running time of an equivalent algorithm in [4] is $\widetilde{O}\left((nnz(A)/\epsilon)^{1/2}\right)$ for uniform random t.

scheme [5], which is based on expanding $(I-A)^{-1}$ as an infinite power series, and then using random walks to estimate this series. With these methods, one can solve the system $\mathbf{x} = G\mathbf{x} + \mathbf{z}$ provided the spectral norm $\rho(G) < 1$. Their convergence was studied in [6], where the authors present matrices G with $\rho(G) < 1$ but $||G||_\infty > 1$ resulting in the von Neumann-Ulam algorithm's divergence. Our algorithm instead uses knowledge of $||G||_\infty$ to adapt its procedure, thereby guaranteeing accurate estimates.

II. Solving Linear Equations via Series Approximation

We now describe how a fast estimator for $\left(A^\ell \mathbf{z}\right)[t]$ can be used to solve for a single element of a linear system. Unless specified otherwise, we use boldface letters (e.g. $\mathbf{x}, \mathbf{y}$) to denote vectors in $\mathbb{R}^{n\times 1}$, and capital letters (e.g. A, Q) to denote matrices in $\mathbb{R}^{n\times n}$. We also use $\mathbf{e}_v$ denote the indicator for index v (i.e., $\mathbf{e}_v[i] = \mathbb{1}_{\{i=v\}}$).

Given a linear system $\mathbf{y} = A\mathbf{x}$, where A is a positive definite matrix, our aim is to estimate $\mathbf{x}[t]$ for some given index $t \in [n]$. Since A is positive definite, we can find a γ such that $G = I - \gamma A$ satisfies $\rho(G) < 1$ (Refer [7], [8] for discussions of how this can be achieved). This gives us a transformed system $\mathbf{x} = G\mathbf{x} + \mathbf{z}$, where $\mathbf{z} = \gamma \mathbf{y}$ and $G = I - \gamma A$. Since we ensure $\rho(G) < 1$, we can write $\mathbf{x}$ as a Neumann series: $\mathbf{x} = \sum_{k=0}^{\infty} G^k \mathbf{z}$. Finally, we can compute $\mathbf{x}[t]$ via $\langle \mathbf{x}, \mathbf{e}_t\rangle = \sum_{k=0}^{\infty} \langle G^k \mathbf{z}, \mathbf{e}_t\rangle$.

Computing $\mathbf{x}[t]$ thus amounts to computing $\mathbf{p}^\ell_{\mathbf{z}}[t] := \langle G^\ell \mathbf{z}, \mathbf{e}_t\rangle = \langle \mathbf{z}, (G^T)^\ell \mathbf{e}_t\rangle$ for any ℓ, and taking their sum over $\ell \in \{0, \dots, \ell_{\max}\}$, where $\ell_{\max}$ is a finite term truncation of the power series. Note that the error from truncating the series to $\ell_{\max}$ can be determined a priori given bounds for $\mathbf{z}$ and $\rho(G)$ – in particular, we can set $\ell_{\max} \geq \frac{1}{\ln \rho(G)} \ln\left(\frac{\Delta(1-\rho(G))}{||\mathbf{z}||}\right)$ to bound the series truncation error by Δ. To see this, note that the error from truncating upto $\ell_{\max}$ terms is given by $ERR_{\ell_{\max}} = \left|\langle \mathbf{z}, Q^{\ell_{\max}+1} \sum_{\ell=0}^{\infty} Q^\ell \mathbf{e}_t\rangle\right|$. Now, for any inner product of the form $\langle \mathbf{z}, Q^\ell \mathbf{e}_t\rangle$, by the Cauchy - Schwarz inequality we have $|\langle \mathbf{z}, Q^\ell \mathbf{e}_t\rangle| \leq ||\mathbf{z}|| \cdot ||Q^\ell \mathbf{e}_t|| \leq ||\mathbf{z}||\rho(Q)^\ell$. Hence we have $ERR_{\ell_{\max}} \leq ||\mathbf{z}||\rho(Q)^{\ell_{\max}+1}\left(\frac{1}{1-\rho(Q)}\right)$ Using $\ell_{\max} \geq \frac{1}{\ln \rho(Q)} \ln\left(\frac{\Delta(1-\rho(Q))}{||\mathbf{z}||}\right)$, we get $ERR_{\ell_{\max}} \leq \Delta$.

III. Existing Algorithms for Estimating Matrix Powers

Based on the above discussion, we henceforth focus on developing bidirectional algorithms for estimating $\mathbf{p}^\ell_{\mathbf{z}}[t] = \langle \mathbf{z}, Q^\ell \mathbf{e}_t\rangle$. To do so, we first describe two existing algorithms that we use as primitives for our procedure: a forward MCMC technique based on the von Neumann-Ulam scheme [13], [6], and a local variational method proposed by Andersen et al. [10] for computing PageRank, and used by Lee et al. [8] in this setting. We present these along with their accuracy and running time guarantees – some of these results follow directly from previous work (as we note in the appropriate sections), and we include their proofs here mainly for the sake of completeness.

We first introduce some notation which will help us better visualize the algorithm. Drawing parallels to the case where Q is a stochastic matrix and $\mathbf{z}$ an element in the n-dimensional simplex (as in [4]), we define a (weighted) directed graph $\mathcal{G}_Q(\mathcal{V}, \mathcal{E})$ with states $\mathcal{V} = [n]$, and edges $(i,j) \in \mathcal{E}$ if $Q_{ij} \neq 0$. Each edge $(i,j) \in \mathcal{E}$ has an associated weight $w_{ij} \in \mathbb{R}$, which we describe later. We refer to the label for a node $v \in V$ (i.e., a dimension $v \in [n]$) as a *dimension-index*, and the exponent of Q as the *step-index*. Finally, we define Q^+ to denote the matrix with $Q^+_{ij} = |Q_{ij}|$.

A. Computing Matrix Powers via Iterative Local-Update

One approach for estimating $\mathbf{p}^\ell_{\mathbf{z}}[t]$ is via a standard power iteration for computing $Q^\ell \mathbf{e}_t$. In settings where n is large such that direct power iteration is infeasible, one can use a 'local' power iteration, which corresponds to a natural dynamic programming update. Informally, the algorithm estimates $\mathbf{p}^\ell_{\mathbf{z}}[t]$ by starting off with a mass of 1 on dimension-index t, and then 'pushing' this mass in reverse along the edges of graph G_Q.

To describe this `REVERSE-LOCAL-UPDATE` algorithm, we first define a `REVERSE-PUSH` local iteration. Essentially, this is a local power-iteration for computing $Q^\ell \mathbf{e}_t$, which adaptively exploits any sparsity in the computation. This operation was defined by Andersen et al. [10] and subsequently used as a primitive in [4], [8]. The critical associated invariant in Lemma 1 was first shown for Personalized PageRank[10]; however, it holds more generally for any matrix Q (as we show below; also see [8]).

For each step-index $k \in \{0, 1, \dots, \ell\}$, we store two vectors: the *estimate vector* $\mathbf{q}^k_t$ and the *residual vector* $\mathbf{r}^k_t$. We initialize all $\mathbf{r}^k_t, \mathbf{q}^k_t, k \in [\ell]$ to 0 except for $\mathbf{r}^0_t$, which we set to $\mathbf{e}_t$. Now, given any dimension-index $v \in [n]$ and step-index $k \in [\ell]$, the `REVERSE-PUSH` operation iteratively updates these vectors as follows:

Algorithm 1 `REVERSE-PUSH`(t, v, k)

Inputs: Matrix Q, estimates $\mathbf{q}^k_t$, residuals $\mathbf{r}^k_t, \mathbf{r}^{k+1}_t$

1: **return** New estimates $\widetilde{\mathbf{q}}^k_t$, residuals $\widetilde{\mathbf{r}}^k_t$ computed as:

$$\widetilde{\mathbf{q}}^k_t \leftarrow \mathbf{q}^k_t + \langle \mathbf{r}^k_t, \mathbf{e}_v\rangle \mathbf{e}_v$$
$$\widetilde{\mathbf{r}}^k_t \leftarrow \mathbf{r}^k_t - \langle \mathbf{r}^k_t, \mathbf{e}_v\rangle \mathbf{e}_v$$
$$\widetilde{\mathbf{r}}^{k+1}_t \leftarrow \mathbf{r}^{k+1}_t + \langle \mathbf{r}^k_t, \mathbf{e}_v\rangle (Q\mathbf{e}_v)$$

Lemma 1: Given the initialization described above, after any sequence of `REVERSE-PUSH` operations, and for any $\mathbf{z} \in \mathbb{R}^n$ and $\ell \geq 0$, the estimates $\{\mathbf{q}^k_t\}$ and residuals $\{\mathbf{r}^k_t\}$ satisfy the following invariant:

$$\mathbf{p}^\ell_{\mathbf{z}}[t] = \langle \mathbf{z}, \mathbf{q}^\ell_t\rangle + \sum_{k=0}^{\ell} \langle \mathbf{z}, Q^k \mathbf{r}^{\ell-k}_t\rangle = \left\langle \mathbf{z}, \mathbf{q}^\ell_t + \sum_{k=0}^{\ell} Q^k \mathbf{r}^{\ell-k}_t \right\rangle$$

The above invariant was first stated in [10] for the case of PageRank vectors. For the sake of completeness, we present a proof (adapted from [2]) for general matrices; an identical invariant was also presented in [8].

Proof: For our chosen initialization (i.e., $\mathbf{r}_t^0 = \mathbf{e}_t$, and all other estimate and residual vectors set to 0), the invariant simplifies to $\mathbf{p}_{\mathbf{z}}^\ell[t] = \langle \mathbf{z}, Q^\ell \mathbf{e}_t \rangle$ which is true by definition. Now, assuming the invariant holds at any stage with vectors $\{\mathbf{q}_t^k\}, \{\mathbf{r}_t^k\}_{k\in[\ell]}$, let $\{\tilde{\mathbf{q}}_t^k\}, \{\tilde{\mathbf{r}}_t^k\}_{k\in[\ell]}$ be the new vectors after executing a REVERSE-PUSH(t, v, k) operation for any given $k \in [\ell]$ and $\mathbf{z} \in \mathbb{R}^n$. We define:

$$\Delta_v^k = \left(\tilde{\mathbf{q}}_t^\ell + \sum_{i=0}^{\ell} (Q^i) \tilde{\mathbf{r}}_t^{\ell-i} \right) - \left(\mathbf{q}_t^\ell + \sum_{i=0}^{\ell} (Q^i) \mathbf{r}_t^{\ell-i} \right)$$

Now to show that the invariant holds following REVERSE-PUSH(t, v, k), it suffices to show that Δ_v^k is zero for any $v \in V$ and $k \in [\ell]$.

We now have three cases: (i) if $\ell < k$, then the REVERSE-PUSH(t, v, k) operation does not affect the residual or estimate vectors $\{\mathbf{q}_t^i, \mathbf{r}_t^i\}_{i<k}$, and hence $\Delta_v^k = 0$; (ii) If $\ell = k$, we have:

$$\begin{aligned} \Delta_v^k &= \left(\tilde{\mathbf{q}}_t^k + \tilde{\mathbf{r}}_t^k\right) - \left(\mathbf{q}_t^k + \mathbf{r}_t^k\right) \\ &= \mathbf{q}_t^k + \langle \mathbf{r}_t^k, \mathbf{e}_v \rangle \mathbf{e}_v + \mathbf{r}_t^k - \langle \mathbf{r}_t^k, \mathbf{e}_v \rangle \mathbf{e}_v - \mathbf{q}_t^k - \mathbf{r}_t^k = 0 \end{aligned}$$

(iii) Finally, when $\ell > k$, we have:

$$\begin{aligned} \Delta_v^k &= Q^{\ell-k} \left(\tilde{\mathbf{r}}_t^k - \mathbf{r}_t^k\right) + Q^{\ell-k-1} \left(\tilde{\mathbf{r}}_t^{k+1} - \mathbf{r}_t^{k+1}\right) \\ &= - \langle \mathbf{r}_t^k, \mathbf{e}_v \rangle Q^{\ell-k} \mathbf{e}_v + \langle \mathbf{r}_t^k, \mathbf{e}_v \rangle Q^{\ell-k-1} (Q\mathbf{e}_v) = 0 \end{aligned}$$

Hence we have shown that the invariant is preserved for any sequence of reverse push operations. ■

The above invariant gives a natural iterative algorithm for computing $\mathbf{p}_{\mathbf{z}}^\ell[t]$: perform repeated REVERSE-PUSH operations controlling the residual vectors $\mathbf{r}_t^k$, and use $\widehat{\mathbf{p}}_{\mathbf{z}}^\ell[t] = \langle \mathbf{z}, \mathbf{q}_t^\ell \rangle$ as the estimate. Depending on the norm we choose to control, we can get a bound for the error via Hölder's inequality. In particular, placing an upper bound on the infinity norm (i.e., the maximum absolute value of the residual vectors) of some chosen $\delta_r > 0$ gives us the error bounds: $|\mathbf{p}_{\mathbf{z}}^\ell[t] - \langle \mathbf{z}, \mathbf{q}_t^\ell \rangle| \leq ||x||_1 \delta_r ||Q||_\infty^\ell$ [2].

Algorithm 2 REVERSE-LOCAL-UPDATE(t, Q, ℓ, δ_r)

Inputs: Matrix Q, maximum step-index ℓ, target residual threshold δ_r

1: Initialize all residual $\mathbf{r}_t^k$ and estimate vectors $\mathbf{q}_t^k, k \in [\ell]$ to 0; set $\mathbf{r}_t^0 = \mathbf{e}_t$
2: **for** $k \in \{0, 1, 2, ...\ell\}$ **do**
3:     **while** $\exists v$ such that $\left|\mathbf{r}_t^k[v]\right| > \delta_r$ **do**
4:         REVERSE-PUSH(t, v, k)
5:     **end while**
6: **end for**
7: **return** $\{\mathbf{q}_t^k\}, \{\mathbf{r}_t^k\}_{k\in[\ell]}$

Finally, we want to bound the running time of REVERSE-LOCAL-UPDATE(t, Q, ℓ, δ_r). It is easy to see that in the worst case, the running time can be as much as the ℓ-hop in-neighborhood of t in Q. However, for a *uniform random* choice of t, we can obtain a more informative bound.

[2]This approach was used in [10], [2], [4]; an alternative is to control $||r_t^k||_2$ giving error bounds in terms of $||x||_2$, which was suggested in [8].

Lemma 2: For any $Q \in \mathbb{R}^{n\times n}$ and uniform random dimension-index $t \in [n]$, the expected running time of REVERSE-LOCAL-UPDATE(t, Q, ℓ, δ_r) is

$$O\left(\frac{nnz(Q)}{n\delta_r}(\ell+1)||Q||_\infty^\ell\right)$$

In particular, note that if $||Q||_\infty \leq 1$ and $\ell = O(1)$, then the average running time is $O\left(\frac{nnz(Q)}{n\delta_r}\right)$.

Proof: Let $T(t)$ be the running time of REVERSE-LOCAL-UPDATE(t, Q, ℓ, δ_r). Recall we define Q^+ as the matrix with $Q_{ij}^+ = |Q_{ij}|$; let $\hat{T}(t)$ be the running time of REVERSE-LOCAL-UPDATE(t, Q^+, ℓ, δ_r). Then we have that for every matrix Q and every t, we have $\hat{T}(t) > T(t)$ – this follows from the fact that any cancellation between positive and negative residuals in REVERSE-LOCAL-UPDATE(t, Q^+, ℓ, δ_r) can only decrease the number of iterations. Also, note that under Q^+ all residuals are positive, so we have for any $k \leq \ell, v \in [n]$, the residuals satisfy $\mathbf{r}_t^k[v] \leq \left(\mathbf{e}_v^T Q^\ell\right)[t]$.

Now let $d_i := \sum_j \mathbb{1}_{\{Q_{ij}\neq 0\}}$, i.e., the support of i^{th} row in Q, and $\mathbf{r}_t^k$ denote the residuals under REVERSE-LOCAL-UPDATE(t, Q^+, ℓ, δ_r). From Algorithm 2, we have $\hat{T}(t) = \sum_{k=0}^{\ell} \sum_{v\in[n]} \mathbb{1}_{\mathbf{r}_t^k[v]>\delta_r}$. Thus, the expected running time over a uniform random choice of $t \in [n]$ is given by

$$\begin{aligned} \frac{1}{n}\sum_t \hat{T}(t) &= \frac{1}{n}\sum_{t\in[n]}\sum_{k=0}^{\ell}\sum_{v\in[n]} \mathbb{1}_{\{\mathbf{r}_t^k>\delta_r\}} d_v \\ &= \frac{1}{n}\sum_{k=0}^{\ell}\sum_{v\in[n]}\sum_{t\in[n]} \mathbb{1}_{\{\mathbf{r}_t^k>\delta_r\}} d_v \\ &\leq \frac{1}{n}\sum_{k=0}^{\ell}\sum_{v\in[n]} \mathbb{1}_{\{(\mathbf{e}_w^T(Q^+)^k)[t]>\delta_r\}} d_v \\ &= \frac{1}{n}\sum_{k=0}^{\ell}\sum_{v\in[n]} \frac{||\mathbf{e}_w^T(Q^+)^k||_1}{\delta_r} d_v \\ &\leq \frac{1}{n}\sum_{k=0}^{\ell}\sum_{v\in[n]} \frac{||Q^+||_\infty^k}{\delta_r} d_v \\ &\leq (\ell+1)\,||Q||_\infty^\ell \frac{nnz(Q)}{n\delta_r} \end{aligned}$$

■

B. Computing Matrix Powers via MCMC Sampling

In the previous section, we computed $\mathbf{p}_{\mathbf{z}}^\ell[t]$ by working backwards from t. Note that our final algorithm is independent of $\mathbf{z}$. We now present an alternate technique which is based on a forward MCMC sampling technique called the von Neumann-Ulam scheme. In this case, the algorithm starts from $\mathbf{z}$, and computes $\mathbf{p}_{\mathbf{z}}^\ell[t]$ for all $t \in [n]$.

More generally, given any vectors $\mathbf{a}$ and $\mathbf{b}$ and matrix Q, the von Neumann-Ulam scheme can be used for computing $\langle \mathbf{a}, Q^\ell \mathbf{b} \rangle$. To understand the algorithm, note that we can expand $\langle \mathbf{a}, Q^\ell \mathbf{b} \rangle$ as the sum

$\sum_{(v_0,\dots,v_\ell)\in V^\ell}\left(\prod_{j\in[\ell]} Q_{v_{j-1}v_j}\right)\mathbf{a}[v_0]\mathbf{b}[v_\ell]$. Now we can interpret this sum as an expectation over an ℓ-step random walk $W=(V_0,V_1,\dots,V_\ell)$ on $\mathcal{G}$, specified as follows:

- V_0, the starting node for the random walk, is sampled from $\sigma_\mathbf{a}=\{|\mathbf{a}[i]|/||\mathbf{a}||_1\}_{i\in[n]}$, and has an associated weight $w_{V_0}=sgn(\mathbf{a}[V_0])||\mathbf{a}||_1$.
- The transition probability matrix for the walk is given by $P_{ij}=\{|Q_{ij}|/||Q_i||_1\}$ (where $||Q_i||_1$ is the 1-norm of the i^{th} row of Q).
- Each edge $(i,j)\in E$ has associated weight $w_{ij}=sgn(Q_{ij})||Q_i||_1$.
- The 'score' for a walk W is the product of weights of traversed edges, i.e.,

$$S_t^\ell(W)=w_{V_0}\prod_{i=0}^{\ell-1}w_{V_iV_{i+1}}\mathbf{b}[V_\ell]$$

This procedure is summarized in Algorithm 3.

Algorithm 3 MCMC-SAMPLER$(Q,\ell,\mathbf{a},\mathbf{b})$

Inputs: Matrix Q, exponent ℓ, vectors $\mathbf{a}$ and $\mathbf{b}$

1: Construct transition matrix $P_{ij}=\{|Q_{ij}|/||Q_i||_1\}$ and starting measure $\sigma_a=\{|\mathbf{a}[i]|/||\mathbf{a}||_1\}_{i\in V}$
2: Define node weights $w_{V_0}=sgn(\mathbf{a}[V_0])||\mathbf{a}||_1$ and edge weights $w_{V^iV^j}=sgn(Q_{V^iV^j})||Q_i||_1$
3: Construct source distribution $\sigma_\mathbf{a}$ with $\sigma_\mathbf{a}[i]=\frac{\mathbf{z}[i]}{||\mathbf{z}||_1}$
4: Sample $V^0\sim\sigma_a$, and generate a random walk $W=\{V^0,V^1,\dots,V^\ell\}$ of length ℓ on $\mathcal{G}$ using transition probability P.
5: **return** Walk-score $S_t^\ell(W)=w_{V_0}\prod_{i=0}^{\ell-1}w_{V_iV_{i+1}}\mathbf{b}[V_\ell]$

Now we have the following Lemma.

Lemma 3: *MCMC-SAMPLER$(Q,\ell,\mathbf{a},\mathbf{b})$ returns a walk-score $S_t^\ell(W)$ which satisfies:*

1) $\mathbb{E}_{W\sim\sigma_\mathbf{a}(P)^\ell}\left[S_t^\ell(W)\right]=\left\langle\mathbf{a},Q^\ell\mathbf{b}\right\rangle$
2) $S_t^\ell(W)\in[\,-||Q||_\infty^\ell||\mathbf{a}||_1||\mathbf{b}||_\infty,\ ||Q||_\infty^\ell||\mathbf{a}||_1||\mathbf{b}||_\infty\,]$

Proof: We can expand $\mathbb{E}_{W\sim\sigma_\mathbf{a}(P)^\ell}\left[S_t^\ell(W)\right]$ to obtain:

$$\begin{aligned}&\mathbb{E}_{W\sim\sigma_\mathbf{a}(P)^\ell}\left[S_t^\ell(W)\right]\\&=\sum_{(V^0,\dots V^\ell)\in[n]^{\ell+1}}\left(\frac{sgn(\mathbf{a}[V_0])||\mathbf{a}||_1|\mathbf{a}[V^0]|}{||\mathbf{a}||_1}\right.\\&\times\left.\left(\prod_{j\in[0,\ell-1]}\frac{sgn(Q_{V^jV^{j+1}}||Q_{V^j}||_1)|Q_{V^jV^{j+1}}|}{||Q_{V^j}||_1}\right)\mathbf{b}[V^\ell]\right)\\&=\sum_{(V^0,\dots V^k)\in[n]^{\ell+1}}\left(\mathbf{a}[V_0]\left(\prod_{j\in[0,\ell-1]}Q_{V^jV^{j+1}}\right)\mathbf{b}[V^\ell]\right)\end{aligned}$$

The final expression is exactly the definition of $\left\langle\mathbf{a},Q^\ell\mathbf{b}\right\rangle$; hence $\mathbb{E}_{W\sim\sigma_\mathbf{a}(P)^\ell}\left[S_t^\ell(W)\right]=\left\langle\mathbf{a},Q^\ell\mathbf{b}\right\rangle$.

Next, in order to bound the score $S_t^\ell(W)$, recall that

$$S_t^\ell(W)=w_{V_0}\prod_{i=0}^{\ell-1}w_{V_iV_{i+1}}\mathbf{b}[V_\ell]$$

We defined $w_{V_0}=sgn(\mathbf{a}[V_0])||\mathbf{a}||_1$, so trivially $|w_{V_0}|\le||\mathbf{a}||_1$. Additionally, $w_{V_iV_{i+1}}=sgn(Q_{V^iV^j})||Q_i||_1$ and by definition $||Q_i||_1\le||Q||_\infty$ and $|\mathbf{b}[V^\ell]|\le||\mathbf{b}||_\infty$. Hence, since we are multiplying ℓ edge scores, we finally obtain $|S_t^\ell(W)|\le||Q||_\infty^\ell||\mathbf{a}||_1||\mathbf{b}||_\infty$ as stated in the lemma. ■

IV. A Bidirectional Algorithm for Computing Matrix Powers

Finally, we present our main contribution: a bidirectional estimator for $\mathbf{p}_\mathbf{z}^\ell[t]=\langle\mathbf{z},Q^\ell\mathbf{e}_t\rangle$. Our algorithm follows the general structure proposed by Lofgren et al. [2], [4] for PageRank and Markov Chain transition probability estimation. It comprises of two distinct components: first we use REVERSE-LOCAL-UPDATE to estimate approximate values of $\left(Q^\ell\mathbf{e}_t\right)[i]$ for all steps $\ell\in[\ell_{\max}]$ and $i\in[n]$. We then use MCMC-SAMPLER to reduce the error in these estimates to get our desired accuracy.

Intuitively, the advantage we gain from combining the two previous algorithms is that running a small amount of local-updates reduces the variance in the walk-scores significantly, which then allows us to perform sampling more effectively. More specifically, given a desired error threshold δ, we first run REVERSE-LOCAL-UPDATE(t,Q,ℓ,δ_r) for an appropriately chosen $\delta_r\gg\delta$ (cf. Theorem 1). At this point, from Lemma 1, we know that we have $\mathbf{p}_\mathbf{z}^\ell[t]=\left\langle\mathbf{z},\mathbf{q}_t^\ell\right\rangle+\sum_{k=0}^\ell\left\langle\mathbf{z},Q^k\mathbf{r}_t^{\ell-k}\right\rangle$, with $\mathbf{r}_t^k[v]\le\delta_r$ for all $v\in[n],k\le\ell$. Now, instead of ignoring the residual terms (as done in [10], [8]; cf. Section III-A), we can further reduce our error by using MCMC-SAMPLER$(Q,k,\mathbf{z},\mathbf{r}_t^k)$ to estimate the residual $\langle\mathbf{z},\mathbf{r}_t^k\rangle$ for all $k\le\ell$. The resulting error is better than ignoring the residual, and also better than directly executing MCMC-SAMPLER$(Q,\ell,\mathbf{z},\mathbf{e}_t)$, as the residuals have magnitude much smaller than 1, and hence the walk-scores have lower variance.

Algorithm 4 BIDIR-MATRIX-POWER$(Q,\mathbf{z},t,\ell)$

Inputs: Matrix Q, exponent ℓ, vector $\mathbf{z}$, target index t.

1: Compute estimate and residual vectors $\{\mathbf{r}_t^k,\mathbf{q}_t^k\}_{\{k\le\ell\}}$ = REVERSE-LOCAL-UPDATE$(t,Q,\ell_{\max},\delta_r)$
2: **for** $i\in[n_f]$ **do**
3: $\quad k\sim Unif[0,\ell]$
4: $\quad S_{i,t}^\ell$ = MCMC-SAMPLER $(Q,k,\mathbf{z},\mathbf{r}_t^{\ell-k})$
5: **end for**
6: **return** $\widehat{\mathbf{p}}_\mathbf{z}^\ell[t]=\left\langle\mathbf{z},\mathbf{q}_t^\ell\right\rangle+\frac{\ell+1}{n_f}\sum_{i=1}^{n_f}S_{i,t}^\ell$

First, we observe that BIDIR-MATRIX-POWER always returns an unbiased estimate for $\mathbf{p}_\mathbf{z}^\ell[t]$ for any choice of n_f.

Lemma 4: *BIDIR-MATRIX-POWER returns an unbiased estimator of $\mathbf{p}_\mathbf{z}^\ell[t]$, i.e. $\mathbb{E}[\widehat{\mathbf{p}}_\mathbf{z}^\ell[t]]=\mathbf{p}_\mathbf{z}^\ell[t]$*

Proof: By definition of $\widehat{\mathbf{p}}_\mathbf{z}^\ell[t]$ and linearity of expectation

$$\mathbb{E}\left[\widehat{\mathbf{p}}_\mathbf{z}^\ell[t]\right]=\left\langle\mathbf{z},\mathbf{q}_t^\ell\right\rangle+(\ell+1)\mathbb{E}_{K\sim Unif[0,\ell]}\left[S_t^K\right]$$

From Lemma 3, we know MCMC-SAMPLER $(Q,k,\mathbf{z},\mathbf{r}_t^k)$ returns an estimate satisfying $\mathbb{E}\left[S_t^k\right]=\left\langle\mathbf{z},Q^k\mathbf{r}_t^k\right\rangle$ for all

$k \le \ell$. Thus, with $K \sim Unif[0,\ell]$, we have

$$\begin{aligned}\mathbb{E}\left[\widehat{\mathbf{p}}^{\ell}_{\mathbf{z}}[t]\right] &= \left\langle \mathbf{z}, \mathbf{q}^{\ell}_t \right\rangle + (\ell+1)\mathbb{E}_K\left[\left\langle \mathbf{z}, Q^K \mathbf{r}^{\ell-K}_t \right\rangle\right] \\ &= \left\langle \mathbf{z}, \mathbf{q}^{\ell}_t \right\rangle + \sum_{k=0}^{\ell} \left\langle \mathbf{z}, Q^k \mathbf{r}^{\ell-k}_t \right\rangle\end{aligned}$$

However, Lemma 1 states that after any sequence of reverse push operations, we have the invariant $\mathbf{p}^{\ell}_{\mathbf{z}}[t] = \left\langle \mathbf{z}, \mathbf{q}^{\ell}_t \right\rangle + \sum_{k=0}^{\ell} \left\langle \mathbf{z}, Q^k \mathbf{r}^{\ell-k}_t \right\rangle$. Thus, $\mathbb{E}\left[\widehat{\mathbf{p}}^{\ell}_{\mathbf{z}}[t]\right] = \mathbf{p}^{\ell}_{\mathbf{z}}[t]$. ■

To choose n_f to get the desired accuracy, we need the following concentration bound, which is a restatement of Hoeffding's inequality (cf. Chapter 1 in [14]).

Lemma 5 (Hoeffding's Inequality): Given $\{X_i\}$ independent random variables s.t. for all i, $X_i \in [-c, c]$ a.s., and $|\mathbb{E}[X_i]| \ge a$. Then $X = \sum_{i=1}^{n} X_i$ satisfies $\mathbb{P}[|X - \mathbb{E}[X]| \ge \epsilon \mathbb{E}[X]] \le p_{fail}$ provided that

$$n \ge \frac{2c^2}{\epsilon^2 a^2} \ln\left(\frac{2}{p_{fail}}\right)$$

Now we can state our main result, where we show how to choose δ_r and n_f such that `BIDIR-MATRIX-POWER` returns an estimate of desired accuracy. The choices of (n_f, δ_r) also affect the running time guarantee, which, as in Lemma 2, is averaged over all targets $t \in [n]$. To this end, we define $T(t)$ to be the running time for a target index t.

Finally, we have our main theorem:

Theorem 1: Given matrix Q and vector $\mathbf{z}$, let $\widehat{\mathbf{p}}^{\ell}_{\mathbf{z}}[t]$ be the estimate of $\mathbf{p}^{\ell}_{\mathbf{z}}[t]$ returned by the `BIDIR-MATRIX-POWER` algorithm with

$$\begin{aligned}\delta_r &= \sqrt[3]{\frac{nnz(Q)\epsilon^2\delta^2}{\ell^2 n ||\mathbf{z}||_1^2 ||Q||_\infty^{\ell} \ln(\ell/p_{fail})}} \\ n_f &= \frac{2\ell_{\max}^2 |\mathbf{z}|_1^2 \delta_r^2 ||Q||_\infty^{2\ell_{\max}}}{\epsilon^2\delta^2} \ln\left(\frac{2}{p_{fail}}\right)\end{aligned}$$

Then we have the following

- *For all $t \in [n]$, we have $|\widehat{\mathbf{p}}^{\ell}_{\mathbf{z}}[t] - \mathbf{p}^{\ell}_{\mathbf{z}}[t]| \le \max\{\delta, \epsilon \mathbf{p}^{\ell}_{\mathbf{z}}[t]\}$ with probability at least $1 - p_{fail}$*
- *For a uniform random choice of $t \in [n]$, the expected running time of the algorithm is*

$$\mathbb{E}[T(t)] = O\left(\left(\frac{||\mathbf{z}||_1 nnz(Q)}{\epsilon\delta n}\right)^{2/3} \ell^{5/3} \left(\ln\frac{2}{p_{fail}}\right)^{1/3}\right)$$

In particular, suppose we choose $\delta = 1/n$, and assuming $||\mathbf{z}||_1 \le 1$ and $||Q||_\infty \le 1$, and $\ell, 1/p_{fail} = O(1)$, then we get $\mathbb{E}[T(t)] = O\left((nnz(Q)/\epsilon)^{2/3}\right)$, as stated in Section I.

Proof: Consider the random walk-score $\mathcal{S}^{\ell}_t = ((\ell+1)/n_f)\sum_{i=1}^{n_f} S^{\ell}_{i,t}$ as in Algorithm 4. From Lemma 4 we have that $\mathbb{E}\left[\mathcal{S}^{\ell}_t\right] = \mathbf{p}^{\ell}_{\mathbf{z}}[t] - \left\langle \sigma, \mathbf{q}^{\ell}_t \right\rangle$; moreover, from Lemma 3, we have that $|\mathcal{S}^{\ell}_t| \le (\ell+1)||\mathbf{z}||_1 \delta_r ||Q||_\infty^{\ell}$. Thus, to achieve relative error ϵ with probability at least $1 - p_{fail}$, we have from Lemma 5 that we need

$$n_f \ge \frac{2(\ell+1)^2 |\mathbf{z}|_1^2 \delta_r^2 ||Q||_\infty^{2\ell_{\max}}}{\epsilon^2\delta^2} \ln\left(\frac{2}{p_{fail}}\right)$$

Given this choice, we know that the running time for the forward MCMC sampling is $O(n_f \ell)$; moreover, from Lemma 2, we get that the reverse work running time for a uniform random choice of $t \in [n]$ is $\frac{nnz(Q)}{n\delta_r}(\ell+1)||Q||_\infty^{\ell}$. Now substituting the value of δ_r given in the statement (which is chosen to balance the two running times), we get the promised running time guarantee. ■

References

[1] L. Page, S. Brin, R. Motwani, and T. Winograd, "The pagerank citation ranking: bringing order to the web.," 1999.

[2] P. Lofgren, S. Banerjee, A. Goel, and C. Seshadhri, "FAST-PPR: Scaling personalized PageRank estimation for large graphs," in *ACM SIGKDD'14*, 2014.

[3] P. Lofgren, S. Banerjee, and A. Goel, "Personalized pagerank estimation and search: A bidirectional approach," in *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining*, pp. 163–172, ACM, 2016.

[4] S. Banerjee and P. Lofgren, "Fast bidirectional probability estimation in markov models," in *Advances in Neural Information Processing Systems*, pp. 1423–1431, 2015.

[5] G. E. Forsythe and R. A. Leibler, "Matrix inversion by a monte carlo method," *Mathematics of Computation*, vol. 4, no. 31, pp. 127–129, 1950.

[6] H. Ji, M. Mascagni, and Y. Li, "Convergence analysis of markov chain monte carlo linear solvers using ulam–von neumann algorithm," *SIAM Journal on Numerical Analysis*, vol. 51, no. 4, pp. 2107–2122, 2013.

[7] I. Dimov, S. Maire, and J. M. Sellier, "A new walk on equations monte carlo method for solving systems of linear algebraic equations," *Applied Mathematical Modelling*, vol. 39, no. 15, pp. 4494–4510, 2015.

[8] C. E. Lee, A. Ozdaglar, and D. Shah, "Asynchronous approximation of a single component of the solution to a linear system," *arXiv preprint arXiv:1411.2647*, 2014.

[9] T. Wu and D. F. Gleich, "Multi-way monte carlo method for linear systems," *arXiv preprint arXiv:1608.04361*, 2016.

[10] R. Andersen, C. Borgs, J. Chayes, J. Hopcraft, V. S. Mirrokni, and S.-H. Teng, "Local computation of pagerank contributions," in *International Workshop on Algorithms and Models for the Web-Graph*, pp. 150–165, Springer, 2007.

[11] S. Kale, Y. Peres, and C. Seshadhri, "Noise tolerance of expanders and sublinear expander reconstruction," in *IEEE FOCS'08*, 2008.

[12] R. Motwani, R. Panigrahy, and Y. Xu, "Estimating sum by weighted sampling," in *Automata, Languages and Programming*, pp. 53–64, Springer, 2007.

[13] W. Wasow, "A note on the inversion of matrices by random walks," *Mathematical Tables and Other Aids to Computation*, pp. 78–81, 1952.

[14] D. P. Dubhashi and A. Panconesi, *Concentration of measure for the analysis of randomized algorithms*. Cambridge University Press, 2009.