A Sharp PageRank Algorithm with Applications to Edge Ranking and Graph Sparsification

Fan Chung and Wenbo Zhao

University of California, San Diego
La Jolla, CA 92093
{fan,w3zhao}@ucsd.edu

Abstract. We give an improved algorithm for computing personalized PageRank vectors with tight error bounds which can be as small as $\Omega(n^{-p})$ for any fixed positive integer p. The improved PageRank algorithm is crucial for computing a quantitative ranking of edges in a given graph. We will use the edge ranking to examine two interrelated problems – graph sparsification and graph partitioning. We can combine the graph sparsification and the partitioning algorithms using PageRank vectors to derive an improved partitioning algorithm.

1 Introduction

PageRank, which was first introduced by Brin and Page [9], is at the heart of Google's web searching algorithms. Originally, PageRank was defined for the Web graph (which has all webpages as vertices and hyperlinks as edges). For any given graph, PageRank is well-defined and can be used for capturing quantitative correlations between pairs of vertices as well as pairs of subsets of vertices. In addition, PageRank vectors can be efficiently computed and approximated (see [3, 4, 8, 13, 14]). The running time of the approximation algorithm in [3] for computing a PageRank vector within an error bound of ϵ is basically $O(1/\epsilon)$. For the problems that we will examine in this paper, it is quite crucial to have a sharper error bound. In Section 2, we will give an improved approximation algorithm with running time $O(m \log(1/\epsilon))$ to compute PageRank vectors within an error bound of ϵ.

The PageRank is originally meant for determining the "importance" of vertices in the Web graph. It is also essential to identify the "importance" of edges in dealing with various problems. We will use PageRank vectors to define a qualitative ranking of edges that we call *Green* values for edges because of its connection with discrete Green functions. The Green values for edges can also be viewed as a generalized version of effective resistances in electric network theory. The detailed definition for Green values of edges will be given in Section 3. We then use the sharp approximate PageRank algorithm to compute Green values within sharp error bounds.

To illustrate the usage of Green values, we examine a basic problem on sparsifying graphs. Graph sparsification was first introduced by Benczúr and Karger

[7, 15–17] for approximately solving various network design problems. The heart of the graph sparsification algorithms are the sampling techniques for randomly selecting edges. The goal is to approximate a given graph G with m edges on n vertices by a sparse graph $\tilde{G}$, called *sparsifier*, with $O(n \log n)$ edges (or fewer) on the same set of vertices in such a way that every cut in sparsifier $\tilde{G}$ is within a factor of $(1 \pm \epsilon)$ of the corresponding cut in G for some constant $\epsilon > 0$. It was shown that, for all $x \in \{0,1\}^n$, $|x^T Lx - x^T \tilde{L}x| \leq \epsilon x^T Lx$, where L and $\tilde{L}$ are the Laplacians of the graph G and its sparsifier $\tilde{G}$, respectively.

Spielman and Teng [24] devised a sampling scheme to construct a *spectral sparsifier* with $O(n \log^c n)$ edges for some (large) constant c in $O(m \text{ polylog}(n))$ time. A spectral sparsifier $\tilde{G}$ for graph G is a sparsifier satisfying $|x^T Lx - x^T \tilde{L}x| \leq \epsilon x^T Lx$ for all $x \in \mathbb{R}^n$. In [25] Spielman and Srivastava gave a different sampling scheme using the effective resistances of electrical networks to construct an improved spectral sparsifier with only $O(n \log n)$ edges. In the process for constructing this spectral sparsifier, they need to use the Spielman-Teng solver [24] as subroutines for solving $O(\log n)$ linear systems. The running time of their sparsification algorithm is mainly dominated by the running time of Spielman-Teng solver which is $O(m \log^c n)$ for a very large constant c [24]. Recently, Batson, Spielman and Srivastava [6] gave an elegant construction for a spectral sparsifier with a linear number of edges although the running time is $O(n^3 m)$. Here, we will use Green values to sample edges of G in order to form the sparsifier $\tilde{G}$ with $O(n \log n)$ edges. There are two advantages of sampling using PageRank and Green values. The running time of our sparsification algorithm is significantly faster and simpler than those in [6, 24] since we avoid using Spielman-Teng solver for solving linear system. In addition, the graph sparsification problem is closely related to graph partitioning algorithms.

For graph partitioning algorithms, previously widely used approach is the recursive spectral method which finds a balanced cut in a graph on n vertices with running time $O((n^2/\lambda)\text{polylog}(n))$ (see [23]), together with an approximation guarantee within a quadratic root of the optimal conductance (where λ denotes the spectral gap of the normalized Laplacian). The running time can be further improved to $O(n^2 \text{polylog}(n))$ by using Spielman-Teng solver for linear systems [24]. Another approach for the balanced cut problem is by using commodity flows [2, 20]. In [2] the approximation guarantee is within a factor of $\log n$ of the optimal, which was further reduced to $O(\sqrt{\log n})$ in [5] but the running time is still $O(n^2 \text{polylog}(n))$. In another direction, Spielman and Teng [24, 25] introduced local graph partitioning which yields a cut near the specified seeds with running time only depending on the volume of the output. Their local partitioning algorithm has an approximation guarantee similar to the spectral method by using a mixing result on random walks [19]. Andersen, Chung and Lang [3] used PageRank vectors to give a local partitioning algorithm with improved approximation guarantee and running time. Recently, Andersen and Peres use involving sets instead of PageRank to further improved the running time [4].

Our balanced-cut algorithm consisting of two parts. First we use PageRank vectors to sparsify the graph. Then we use the known PageRank partitioning algorithm to find a balanced cut. Both parts have the same complexity as computing the PageRank vectors. Consequently, the complexity for our PageRank balanced-cut algorithm is $O(m \log^2 n/\phi + n \text{ polylog}(n))$ for any input graph on n vertices and m edges. The balanced-cut algorithm here can be viewed as an application of graph sparsification.

In this paper, we omit the proofs for most of lemmas and theorems (except Theorems 4, 5, and 7) which will be included in the full paper version.

2 A sharp PageRank approximation algorithm

We consider an undirected, weighted graph $G = (V, E, w)$ with n vertices and m edges where the edge weights $w(u, v) = w(v, u) \geq 0$ and the edge set E consists of all pairs (u, v) with $w(u, v) > 0$. The weighted degree $d(u)$ of vertex u is the sum of $w(u, v)$ over all v, i.e., $d(u) = \sum_v w(u, v)$.

A typical random walk is defined by its transition probability matrix P satisfying $P(u, v) = w(u, v)/d(u)$. We may write $P = D^{-1}A$, where A is the weighted adjacency matrix satisfying $A(u, v) = w(u, v)$ for all pairs of $(u, v) \in E$ and D is the diagonal matrix of weighted degree. We here consider the *lazy walk* Z on G, defined by $Z = (I + P)/2$. In [3], PageRank vector pr is defined by a recurrence relation involving a seed vector s (as a probability distribution) and a positive jumping constant $\alpha < 1$ (or transportation constant), i.e. $\text{pr} = \alpha s + (1 - \alpha)\text{pr}Z$ where pr and s are taken to be row vectors. In this paper, we consider the PageRank pr as a discrete Green's function $\alpha s(I - (1 - \alpha)Z)^{-1} = \beta s(\beta I + \mathbf{L})^{-1}$ where $\beta = 2\alpha/(1-\alpha)$ and $\mathbf{L} = I - P$. Note that the usual Green's function is associated with the pseudo inverse of L. Another way to express the recurrence of PageRank in terms of β and s is that: for a positive value $\beta > 0$, the (personalized) PageRank vector $\text{pr}_{\beta,s}$ with a seed vector s is the unique solution of equation $\text{pr}_{\beta,s} = \frac{\beta}{2+\beta}\, s + \frac{2}{2+\beta}\text{pr}_{\beta,s}Z$. If a seed vector is the characteristic function χ_u of a single vertex u, we may write $\text{pr}_{\beta,\chi_u} = \text{pr}_{\beta,u}$ if there is no confusion. It is easy to check that $\sum_{v\in V} \text{pr}_{\beta,s}(v) = 1$ since $\sum_{v\in V} s(v) = 1$.

The PageRank approximation algorithm in [3] contains the following routines, called **Push** and **ApproximatePR**, which serve as subroutines later in the sharp approximate PageRank algorithm. Given a vertex u, an approximate PageRank vector p and a residual vector r, the **Push** operation is as follows:

```
Push(u):
  Let $p' = p$ and $r' = r$, except for these changes:
    1. let $p'(u) = p(u) + \frac{\beta}{2+\beta} r(u)$ and $r'(u) = \frac{r(u)}{2+\beta}$;
    2. for each vertex $v$ such that $(u,v) \in E$: $r'(v) = r(v) + \frac{r(u)}{(2+\beta)d(u)}$.
```

Lemma 1 ([3]). *Let p' and r' denote the resulting vectors after performing operation* **Push**(u) *with vectors p and r. Then $p = \text{pr}_{\beta,s-r}$ implies $p' = \text{pr}_{\beta,s-r'}$.*

Theorem 1 ([3]). *For any vector s with $\|s\|_1 \leq 1$, and any constant $\epsilon \in (0, 1]$, the algorithm* **ApproximatePR**(s, β, ϵ) *computes an approximate PageRank* *vector $p = \mathrm{pr}_{\beta,s-r}$ such that the residual vector r satisfies $|r(v)/d(v)| \le \epsilon$ for all $v \in V$. The running time of algorithm is $O(\frac{2+\beta}{\epsilon\beta})$.*

```
ApproximatePR($s, \beta, \epsilon$):
  1. Let $p = \mathbf{0}$ and $r = s$.
  2. While $r(u) \ge \epsilon d(u)$ for some vertex $u$:
       pick any vertex $u$ where $r(u) \ge \epsilon d(u)$ and apply operation Push(u).
  3. Return $p$ and $r$.
```

We will improve the estimate error bound for the above algorithm by the following iterative process.

```
SharpApproximatePR($s, \beta, \epsilon$):
  1. Let $\epsilon' = 1$, $r = s$ and $p = \mathbf{0}$.
  2. While $\epsilon' > \epsilon$ :
     (a) set $\epsilon' = \epsilon'/2$;
     (b) let $p'$ and $r'$ be the output of ApproximatePR($r$, $\beta$, $\epsilon'$);
     (c) let $p = p + p'$ and $r = r'$.
  3. Return $p$ and $r$.
```

Theorem 2. *Given constants $\epsilon \in (0, 1]$, $\beta > 0$ and seed vector s, to approximate PageRank vector $\mathrm{pr}_{\beta,s}$, the algorithm* **SharpApproximatePR**(s, β, ϵ) *computes approximate PageRank vector $p = pr_{\beta,s-r}$ such that the residual vector r satisfies $|r(v)/d(v)| \le \epsilon$ for all $v \in V$ and the running time is $O(\frac{(2+\beta)m \log(1/\epsilon)}{\beta})$. In particular, if ϵ is an inverse of a polynomial on n with degree p, i.e. $\Omega(n^{-p})$, the running time can be bounded by $O(\frac{m \log n}{\beta})$.*

3 The Green values for edges in a graph

Recall that the combinatorial Laplacian of G is defined by $L = D - A$. If we orient the edges of G in an arbitrary but fixed way, we can write its Laplacian as $L = B^T W B$, where $W_{m\times m}$ is a diagonal matrix with $W(e, e) = w(e)$ and $B_{m\times n}$ is the signed edge-vertex incidence matrix such that $B(e, v) = 1$ for v is e's head; $B(e, v) = -1$ for v is e's tail; and $B(e, v) = 0$ otherwise.

The normalized Laplacian of G is defined to be $\mathcal{L} = D^{-1/2} L D^{-1/2}$ and we can write $\mathcal{L} = S^T W S$ where $S_{m\times n} = BD^{-1/2}$. Since $\mathcal{L}$ is symmetric and we have $\mathcal{L} = \sum_{i=0}^{n-1} \lambda_i \phi_i^T \phi_i$, where $\lambda_0 = 0 < \lambda_1 \le \lambda_2 \le \ldots \le \lambda_{n-1} \le 2$ are eigenvalues of $\mathcal{L}$ and $\phi_0, \ldots, \phi_{n-1}$ are a corresponding orthonormal basis of eigenvectors. Various properties concerning eigenvalues of the normalized Laplacian can be found in [10]. Denote the β-normalized Laplacian $\mathcal{L}_\beta$ by $\beta I + \mathcal{L}$, and we may write $\mathcal{L}_\beta = S'^T W_\beta S'$ where we define S' and W_β as follows:

$$S' = \begin{bmatrix} I \\ S \end{bmatrix}_{(n+m)\times n} \text{ and } W_\beta = \begin{bmatrix} \beta I & 0 \\ 0 & W \end{bmatrix}_{(n+m)\times(n+m)}.$$

Here the index set for the columns of S' and columns (rows) of W_β is $V \cup E$ where the first n columns (rows) are indexed by V and the last m columns (rows) are indexed by E.

Green's functions were firstly introduced in a celebrated essay by George Green [12] in 1828. The discrete analog of Green's functions which are associated with the normalized Laplacian of graphs were considered in [11] in connection with the study of Dirichlet eigenvalues with boundary conditions. In particular, the following modified Green's function $\mathcal{G}_\beta$ was used in [11]. For $\beta \in \mathbb{R}^+$, let Green's function $\mathcal{G}_\beta$ denote the symmetric matrix satisfying $\mathcal{L}_\beta \mathcal{G}_\beta = I$. Clearly, we have $\mathcal{G}_\beta = \sum_{i=0}^{n-1} \frac{1}{\lambda_i+\beta} \phi_i^T \phi_i$. We remark that the discrete Green's function is basically a symmetric form of the PageRank. Namely, it is straightforward to check that $\mathrm{pr}_{\beta,s} = \beta s D^{-1/2} \mathcal{G}_\beta D^{1/2}$. For each edge $e = \{u, v\} \in E$, we define the *Green value* $g_\beta(u,v)$ of e to be a combination of four terms in PageRank vectors

$$g_\beta(u,v) = \frac{\mathrm{pr}_{\beta,u}(u)}{d(u)} - \frac{\mathrm{pr}_{\beta,u}(v)}{d(v)} + \frac{\mathrm{pr}_{\beta,v}(v)}{d(v)} - \frac{\mathrm{pr}_{\beta,v}(u)}{d(u)}. \tag{1}$$

Actually, one can also verify that $g_\beta(u,v) = \beta(\chi_u - \chi_v) D^{-1/2} \mathcal{G}_\beta D^{-1/2} (\chi_u - \chi_v)^T$. By using the above properties of Green's function, we can prove the following facts which will be useful later.

Lemma 2. *The Green value $g_\beta(u,v)$ can be expressed as $\sum_{i=0}^{n-1} \frac{\beta}{\lambda_i+\beta} (\frac{\phi_i(u)}{\sqrt{d(u)}} - \frac{\phi_i(v)}{\sqrt{d(v)}})^2$. Particularly, for two distinct vertices $u, v \in V$, $\frac{\beta}{2+\beta}\left(\frac{1}{d(u)} + \frac{1}{d(v)}\right) \le g_\beta(u,v) \le \frac{1}{d(u)} + \frac{1}{d(v)}$.*

Since the Green values are relatively small (e.g., of order $\Omega(1/n^p)$ as Lemma 2 for positive integer p), we need very sharply approximate PageRank to be within a factor of $1 + O(1/n^p)$ of the exact values in the analysis of the performance bound for the graph sparsification algorithms that we will examine in Section 4. For all the pairs (u, v), we define the approximate Green value $\tilde{g}_\beta(u,v)$ by

$$\tilde{g}_\beta(u,v) = \frac{\mathrm{pr}_{\beta,\chi_u - r_{\chi_u}}(u)}{d(u)} - \frac{\mathrm{pr}_{\beta,\chi_u - r_{\chi_u}}(v)}{d(v)} + \frac{\mathrm{pr}_{\beta,\chi_v - r_{\chi_v}}(v)}{d(v)} - \frac{\mathrm{pr}_{\beta,\chi_v - r_{\chi_v}}(u)}{d(u)}.$$

Here, $\mathrm{pr}_{\beta,\chi_u - r_{\chi_u}}$ and $\mathrm{pr}_{\beta,\chi_u - r_{\chi_u}}$ are the approximate PageRank vectors as outputs of **ApproximatePR** for $\mathrm{pr}_{\beta,u}$ and $\mathrm{pr}_{\beta,v}$ respectively; r_{χ_u} and r_{χ_v} are the corresponding residual vectors satisfying $\|r_{\chi_u} D^{-1}\|_1 \le \epsilon/4$ and $\|r_{\chi_v} D^{-1}\|_1 \le \epsilon/4$. With this definition, we can prove that

Lemma 3. *For two distinct vertices $u, v \in V$, we have $|g_\beta(u,v) - \tilde{g}_\beta(u,v)| \le \epsilon$.*

Here we will give a rough estimate for computing Green's values by directly using Lemma 3 and Theorem 1. Note that by invoking Theorem 1 without using further techniques the running time does not seem to be improved.

Theorem 3. *Given any constant $\epsilon > 0$ and any pair $(u, v) \in V \times V$, the approximate Green value $\tilde{g}_\beta(u,v)$ can be computed in $O(\frac{2+\beta}{\beta\epsilon})$ time such that*

$|g_\beta(u,v) - \tilde{g}_\beta(u,v)| \leq \epsilon$. In particular, after $O(\frac{(2+\beta)n}{\beta\epsilon})$ preprocessing time, for each $(u,v) \in V \times V$, we can compute such $\tilde{g}_\beta(u,v)$ by using a constant number of queries.

Recall that in Lemma 2, we established a lower bound for $g_\beta(u,v)$. We denote as Δ the maximum degree of graph G. Then, a direct consequence of the above theorem is the following.

Corollary 1. *Given any constant $\epsilon > 0$ and any pair $(u,v) \in V \times V$, we can compute quantity $\tilde{g}_\beta(u,v)$ in $O(\frac{\Delta}{\beta^2\epsilon})$ time such that $|g_\beta(u,v) - \tilde{g}_\beta(u,v)| \leq \epsilon g_\beta(u,v)$. In particular, after $O(\frac{\Delta n}{\beta^2\epsilon})$ preprocessing time, for each $(u,v) \in V \times V$, we can compute such $\tilde{g}_\beta(u,v)$ by using a constant number of queries.*

We will improve both Theorem 3 and Corollary 1 in the Section 5 by using sharp approximate PageRank algorithms and dimension reduction techniques.

4 Graph sparsification using Green values

To construct our sparsifier, we use a method quite similar to the scheme used by Spielman and Srivastava except that PageRank is used here instead of effective resistance. We will give three graph sparsification algorithms, two of which involve approximate Green values (which will be examined in section 5). In this section, we use exact Green values for edges.

Recall that the graph $G = (V, E, w)$ we consider here is an undirected weighted graph. For a subset S of vertices in G, the *edge boundary* $\partial(S)$ of S consists of all edges with exactly one endpoint in S. The weight of $\partial(S)$, denoted by $w(S, \bar{S})$, is the sum of all edge weights of edges in $\partial(S)$. The *volume* of S, denoted by $\mathrm{vol}(S)$, is defined to be the sum of *degrees* $d(v)$ over all v in S. When $S = V$, we write $\mathrm{vol}(S) = \mathrm{vol}(G)$. The *Cheeger ratio* (or *conductance*) $h_G(S)$ of S in G is defined by $h_G(S) = w(S, \bar{S})/\min\{\mathrm{vol}(S), \mathrm{vol}(\bar{S})\}$. The *conductance* h_G of G is defined to be the minimum Cheeger ratio among all subsets S with $\mathrm{vol}(S) \leq \mathrm{vol}(G)$.

The goal of sparsification is to approximate a given graph G by a sparse graph $\tilde{G}$ on the same set of vertices while the sparse graph $\tilde{G}$ preserves the Cheeger ratios of every subset of vertices to within a factor of $1 \pm \epsilon$. The main step in any sparsification algorithm [7, 15–17, 24, 25] is to choose an appropriate probability distribution for random sampling the edges in a way that Cheeger ratios of subsets change little. Our sparsification algorithm is a sampling process using probabilities proportional to the Green values g_β's as follows:

```
$\tilde{G}$ = SparsifyExactGreen($G, q, \beta$):
  For each $e = (u,v) \in E$, set probability $p_e \propto w(e)g_\beta(e)$ and repeat the
  following steps for $q$ times:
    1. Choose an edge $e \in G$ randomly with probability $p_e$
    2. Add $e$ to $\tilde{G}$ with weight $w(e)/qp_e$.
    3. Sum the weights if an edge is chosen more than once.
```

The analysis of the above algorithm will be examined in the following subsections. Our main theorem is the following

Theorem 4. *Given an unweighed graph G on n vertices with m edges, for any $\epsilon \in (0,1)$, let $\tilde{G}$ denote the output of the algorithm* **SparsifyExactGreen**(G, q, β)*, where $q = 256C^2 n \log n/\epsilon^2$, $\beta = 1/2$, $C \geq 1$ is a absolute constant. Then with probability at least 1/2, we have $|h_{\tilde{G}}(S) - h_G(S)| \leq \epsilon$ for all $S \subset V$.*

4.1 Analyzing the sparsifier

Our analysis follows the general scheme as that of [25]. In our analysis of sparsifier, we consider the matrix $\Lambda_\beta = W_\beta^{1/2} S' \mathcal{G}_\beta S'^T W_\beta^{1/2}$. Note that Λ_β is a $(n+m) \times (n+m)$ matrix and we index its the first n columns (rows) by V and its last m columns (rows) by E. From the definition and properties of Green values in Section 3, one can verify that $\Lambda_\beta(e,e) = \frac{1}{\beta}\sqrt{W_\beta(e,e)} g_\beta(e) \sqrt{W_\beta(e,e)} = \frac{1}{\beta} w(e) g_\beta(e)$. Here are several useful properties for Λ_β.

Lemma 4. *(i) $\Lambda_\beta^2 = \Lambda_\beta$. (ii) The dimension (or rank) of Λ_β, denoted by dim(Λ_β) is n. (iii) The eigenvalues of Λ_β are 1 with multiplicity n and 0 with multiplicity m. (iv) $\Lambda_\beta(e,e) = \|\Lambda_\beta(\cdot,e)\|_2^2$.*

Next, we introduce some notations and several lemmas that the theorems in later sections rely on. Let $\tilde{w}(e)$ be the edge weight of edge e in $\tilde{G}$. Recall that q is a number and p_e is the sampling probability for $e \in E$. Denote I_β as a nonnegative diagonal matrix

$$I_\beta = \begin{bmatrix} I_{n\times n} & 0 \\ 0 & R \end{bmatrix}_{(n+m)\times(n+m)} \text{ where } R(e,e) = \frac{\tilde{w}(e)}{w(e)} = \frac{\text{\# of times } e \text{ is sampled}}{q p_e}.$$

Lemma 5. *Suppose I_β is a nonnegative diagonal matrix such that $\|\Lambda_\beta I_\beta \Lambda_\beta - \Lambda_\beta \Lambda_\beta\|_2 \leq \epsilon$. Then $\forall x \in \mathbb{R}^n$, $|x\tilde{\mathcal{L}}_\beta x^T - x\mathcal{L}_\beta x^T| \leq \epsilon x \mathcal{L}_\beta x^T$ where $\mathcal{L}_\beta = S'^T W_\beta S'$ and $\tilde{\mathcal{L}}_\beta = S'^T W_\beta^{1/2} I_\beta W_\beta^{1/2} S'$.*

Lemma 6 ([22]). *Let $\mathbf{p}$ be a probability distribution over $\Omega \subseteq \mathbb{R}^d$ such that $\sup_{y\in\Omega}\|y\|_2 \leq M$ and $\mathbb{E}_{\mathbf{p}}\|y^T y\|_2 \leq 1$. Let $y_1 \ldots y_q$ be independently samples drawn from $\mathbf{p}$. Then for every $1 > \epsilon > 0$, $\mathbb{P}\left\{\left\|\frac{1}{q}\sum_{i=1}^q y_i^T y_i - \mathbb{E} y^T y\right\|_2 > \epsilon\right\} \leq 2\exp(-\epsilon^2/a^2)$, where $a = \min\left(CM\sqrt{\frac{\log q}{q}}, 1\right)$ and C is an absolute constant.*

4.2 Proof of Theorem 4

We first prove the following theorem which leads to the proof of Theorem 4.

Theorem 5. *Let $\mathcal{L}$ be the normalized Laplacian of G and $\tilde{G}$ be the output of the algorithm* **SparsifyExactGreen**(G, q, β)*, where $q = 4C^2 n \log n/\epsilon^2$, $\epsilon \in (0,1]$ and C is a absolute constant. Then with probability at least 1/2, we have $\forall x \in \mathbb{R}^n$, $|x\tilde{\mathcal{L}}_\beta x^T - x\mathcal{L}_\beta x^T| \leq \epsilon x\mathcal{L}_\beta x^T$, where $\mathcal{L}_\beta = \beta I + \mathcal{L} = S'^T W_\beta S'$ and $\tilde{\mathcal{L}}_\beta = S'^T W_\beta^{1/2} I_\beta W_\beta^{1/2} S'$.*

Brief proof of Theorem 5: Before applying Lemmas 5 and Lemma 6 we observe that $\Lambda_\beta I_\beta \Lambda_\beta = \sum_{e\in E} R(e,e)\Lambda_\beta(e,\cdot)^T\Lambda_\beta(e,\cdot) + \sum_{v\in V}\Lambda_\beta(v,\cdot)^T\Lambda_\beta(v,\cdot)$ and $\Lambda_\beta\Lambda_\beta = \sum_{e\in E}\Lambda_\beta(e,\cdot)^T\Lambda_\beta(e,\cdot) + \sum_{v\in V}\Lambda_\beta(v,\cdot)^T\Lambda_\beta(v,\cdot)$. Thus we have $\Lambda_\beta I_\beta\Lambda_\beta - \Lambda_\beta\Lambda_\beta = \sum_{e\in E}(R(e,e)-1)\Lambda_\beta(e,\cdot)^T\Lambda_\beta(e,\cdot)$. Now, let us consider $\sum_{e\in E} R(e,e)\Lambda_\beta(e,\cdot)^T\Lambda_\beta(e,\cdot)$ which can be expressed as

$$\sum_{e\in E}\frac{\#\text{ of times } e \text{ is sampled}}{qp_e}\Lambda_\beta(e,\cdot)^T\Lambda_\beta(e,\cdot) = \frac{1}{q}\sum_{i=1}^{q} y_i^T y_i$$

where $y_1,\dots,y_q$ are random vectors drawn independently with replacement from the distribution $\mathbf{p}$ defined by setting $y = \frac{1}{\sqrt{p_e}}\Lambda_\beta(e,\cdot)$ with probability p_e.

We also need to bound the norm of the expectation of y^Ty and the norm of y. By using the properties of Λ_β in Lemma 4, we can show that $\|\mathbb{E}_{\mathbf{p}} y^T y\|_2 \le 1$ and $\frac{1}{\sqrt{p_e}}\|\Lambda_\beta(e,\cdot)\|_2 \le \sqrt{n}$ (details are omitted). Notice that if we let $q = 4C^2 n\log n/\epsilon^2$ then we have $\min\left(CM\sqrt{\frac{\log q}{q}},1\right) \le C\sqrt{\epsilon^2\frac{n\log(4C^2 n\log n/\epsilon^2)}{4C^2 n\log n}} \le \epsilon/2$. By applying the Rudelson and Vershynin's lemma in [22] (Lemma 6), we completes the proof of the theorem.

□

Before applying Theorem 5 to prove Theorem 4, we still need the following two lemmas. We here consider G as an unweighted graph first, i.e $w(e) = 1$ for all edges, although this can be easily extended to the general weighted graphs.

Lemma 7. *For any constant $\epsilon \in (0,1]$, let $\tilde{G}$ be the output of algorithm* **SparsifyExactGreen** *(G,q,β), where $q = 4C^2 n(\beta+2)\log n/\epsilon^2$. Then, with probability $1-1/n$, for all subsets $S\subset V$, we have $|\mathrm{vol}_{\tilde{G}}(S) - \mathrm{vol}_G(S)| \le \epsilon \mathrm{vol}_G(S)$.*

Lemma 8. *If sparse graph $\tilde{G}$ corresponding to graph G satisfies two conditions: (a) for all $x\in\mathbb{R}^n$, $|x\tilde{\mathcal{L}}_\beta x - x\mathcal{L}_\beta x^T| \le \epsilon x\mathcal{L}_\beta x^T$; (b) for all subsets $S\subset V$, $|\mathrm{vol}_{\tilde{G}}(S) - \mathrm{vol}_G(S)| \le \epsilon\mathrm{vol}_G(S)$. Then $|h_{\tilde{G}}(S) - h_G(S)| \le 2\epsilon h_G(S) + \epsilon\beta$.*

Proof of Theorem 4: To prove Theorem 4, we need to combine Lemma 7, Lemma 8 and Theorem 5. For any $1 > \epsilon > 0$, let $\tilde{G}$ be the output of the algorithm **SparsifyExactGreen**(G,q,β), where $q = 256C^2 n\log n/\epsilon^2$, $\beta = 1/2$, and $C\ge 1$ is a constant. By Theorem 5 and Lemma 7, the conditions of Lemma 8 are satisfied with probability at least 1/2. Note that we have chosen β to be 1/2 and $h_G(S)\le 1$, thus algorithm **SparsifyExactGreen** can be applied by using $O(\frac{n\log n}{\epsilon^2})$ sampling. Furthermore, for all $S\in V$, we have $|h_{\tilde{G}}(S) - h_G(S)| \le \epsilon$.
□

By choosing a different β, namely, $\beta = \phi/2$, we have the following:

Theorem 6. *Given constants $\epsilon,\phi\in(0,1]$, let $\tilde{G}$ be the output of the algorithm* **SparsifyExactGreen***(G,q,β), where $q = 256C^2 n\log n/\epsilon^2$, $\beta = \phi/2$, and $C\ge 1$. Then with probability at least 1/2, we have $|h_{\tilde{G}}(S) - h_G(S)| \le \epsilon h_G(S)$ for all $S\in V$ with $h_G(S)\ge\phi$.*

5 Sparsification using approximate PageRank vectors

In Corollary 1, we can compute approximate Green values $\tilde{g}_\beta(u,v)$ satisfying $(1-\kappa)g_\beta(u,v) \le \tilde{g}_\beta(u,v) \le (1+\kappa)g_\beta(u,v)$ for all edges $(u,v) \in E$ in $O(\Delta n)$ time, where κ is any absolute constant such that $\kappa < 1$ (e.g., $\kappa = 0.01$). Instead of using exact Green values, we can use approximate Green values as we run the algorithm **SparsifyExactGreen**. The approximate Green values $\tilde{g}_\beta$'s are combinations of approximate PageRank vectors $\mathrm{pr}'_{\beta,v}$'s. Here we choose the parameters for algorithm **ApproximatePR**: $\mathrm{pr}'_{\beta,v} = \textbf{ApproximatePR}(\chi_v, \beta, \frac{\beta}{(2+\beta)\Delta}\kappa)$. It is not difficult to verify that all results in Section 4 will change at most by a constant factor if we run the algorithm **SparsifyExactGreen** by using approximate Green values as above. The performance guarantee and the number of sampled edges in the Theorems 4 differ by at most a constant factor, although the computational complexity will increase to $O(\Delta n)$. In order to further improve the running time, we use several methods in the following subsections.

5.1 Graph sparsification by using sharply approximate Green values

In order to have better error estimate of approximate Green values, we need to improve the error estimate for approximate PageRank vectors in Theorem 1. We will use the strengthened approximate PageRank algorithm **SharpApproximatePR** and the dimension reduction technique in [25] to approximate the Green values by using these sharply approximate PageRank vectors produced by **SharpApproximatePR**.

First, we recall that $g_\beta(u,v) = \beta(\chi_u-\chi_v)D^{-1/2}\mathcal{G}_\beta D^{-1/2}(\chi_u-\chi_v)^T$ and thus

$$\begin{aligned} & g_\beta(u,v) \\ &= \beta(\chi_u-\chi_v)D^{-1/2}\mathcal{G}_\beta\mathcal{L}_\beta\mathcal{G}_\beta D^{-1/2}(\chi_u-\chi_v)^T \\ &= \beta\|W_\beta S'\mathcal{G}_\beta D^{-1/2}(\chi_u-\chi_v)^T\|_2^2 = \frac{1}{\beta}\|W_\beta S' D^{1/2}[\beta D^{-1/2}\mathcal{G}_\beta D^{-1/2}](\chi_u-\chi_v)^T\|_2^2. \end{aligned}$$

Therefore, $g_\beta(u,v)$'s are just pairwise distances between vectors $\{\mathcal{Z}\chi_v^T\}_{v\in V}$ where $\mathcal{Z} = W_\beta S' D^{1/2}[\beta D^{-1/2}\mathcal{G}_\beta D^{-1/2}]$. However, the dimension of the vectors in $\{\mathcal{Z}\chi_v^T\}_{v\in V}$ is $m+n$. In order to reduce the computational complexity for computing these vectors, we project these vectors into a lower dimensional space while preserving their pairwise distances by the following lemma.

Lemma 9 ([1]). *Given vectors $x_1,\ldots,x_n \in \mathbb{R}^d$ and constants $\epsilon,\gamma > 0$, let $k_0 = c_\gamma \log n/\epsilon^2$ where c_γ is a constant depending on γ. For integer $k \ge k_0$, let $R_{k\times d}$ be a random matrix where $\{R_{ij}\}$ are independent random variables with values $\pm 1/\sqrt{k}$. Then with probability $1-\frac{1}{n^\gamma}$, we have*

$$(1-\epsilon)||x_i-x_j||_2^2 \le ||Rx_i - Rx_j||_2^2 \le (1+\epsilon)||x_i-x_j||_2^2.$$

Now, we are ready to state our algorithm to approximate the Green values. Later, in order to analysis our algorithm **ApproxiamteGreen**, we will give a bound for y_i's by Lemma 10.

```
**ApproxiamteGreen**($\beta$, $\epsilon$, $k$):
  1. Let $R_{k\times(n+m)} = [R_1, R_2]$ be a random matrix whose entries are
     independent random variables with values $\pm 1/\sqrt{k}$, where $R_1$ is an $k \times n$
     matrix and $R_2$ is an $k \times m$ matrix.
  2. Let $Y = RW_\beta^{1/2}S'D^{1/2}$ and $\tilde{\mathcal{Z}} = R\mathcal{Z}$.
  3. For $i = 1, \dots, k$, do the following
     (a) Let $y_i$ be the $i$th row of $Y$ and $\tilde{z}_i$ be the $i$th row of $\tilde{\mathcal{Z}}$.
     (b) Approximate $\tilde{z}_i$ by $\tilde{z}_i{}' = $ **SharpApproximatePR**($y_i$, $\beta$, $\epsilon/n^r$).
  4. Let $\tilde{\mathcal{Z}}'$ be the approximated matrix for $\tilde{\mathcal{Z}}$ whose rows are $\tilde{z}_1{}', \dots, \tilde{z}_k{}'$.
     For all $(u, v) \in E$, return $\tilde{g}_\beta(u, v) = \|\tilde{\mathcal{Z}}'(\chi_u - \chi_v)^T\|_2^2$.
```

Lemma 10. *Given an integer k and a random matrix whose entries are independent random variables with values $\pm 1/\sqrt{k}$, with probability $1 - 1/n^2$, we have $||y_i||_1 \le c(\sum_{v\in V}\sqrt{d(v)})\sqrt{\log n}$ for $1 \le i \le k$, where c is an absolute constant.*

By combining the above lemmas and Theorem 2, we have the follow theorem.

Theorem 7. *Given any constant $\epsilon > 0$, let $k = c\log n/\epsilon^2$. If $\beta = \Omega(\text{poly}(1/n))$, algorithm* **ApproxiamteGreen**(β, ϵ, k) *will output $\tilde{g}_\beta(u, v)$ satisfying $|g_\beta(u, v) - \tilde{g}_\beta(u, v)| \le \epsilon g_{\beta(u,v)}$ in $O(\frac{m\log^2 n}{\beta\epsilon^2})$ time.*

Proof. To bound the running time, note that step 1 can be completed in $O(m\log n/\epsilon)$ time since R is a $k \times (n+m)$ random matrix. In step 2, we set $Y = RW_\beta^{1/2}S'D^{1/2}$ and it only takes $O(m\log n/\epsilon^2)$ time since S' has $O(m)$ entries and $W_\beta^{1/2}$ is a diagonal matrix.

In step 3, Let y_i be the ith row of Y and $\tilde{z}_i$ be the ith row of $\tilde{\mathcal{Z}}$, i.e., $(Y[\beta D^{-1/2}\mathcal{G}_\beta D^{-1/2}])_{k\times n}$. Therefore, we have $\tilde{z}_i = y_i[\beta D^{-1/2}\mathcal{G}_\beta D^{-1/2}]$ and we can view $\tilde{z}_i$ as a scaled PageRank vector with seed vector y_i.

In Lemma 10, we have proved that with probability at least $1 - 1/n$, $\|y_i\|_1 = O(\sum_{v\in V}\sqrt{d(v)\log n})$ for $1 \le i \le k$. Without loss of generality, we may assume $O(\sum_{v\in V}\sqrt{d(v)\log n}) = O(m)$ otherwise the graph is sufficient sparse. Thus, $\tilde{z}_i$ can be approximated by using algorithm **SharpApproximatePR** with arbitrary small absolute error, say, ϵ'. Thus, each call of **SharpApproximatePR** just takes $O(\frac{m\log(1/\epsilon')}{\beta})$ time. By Lemma 2, $g_\beta(u, v) = \Omega(\beta/n)$ which implies that we only need to set $\epsilon' = \epsilon/n^r$ for some large enough but fixed constant r. Thus, each call of **SharpApproximatePR** will actually take $O(m\log n/\beta)$ time. Since there are $k = O(\log n/\epsilon^2)$ calls, the total running time of step 3 is $O(m\log^2 n/\beta\epsilon^2)$.

In step 4, since each column of $\tilde{\mathcal{Z}}'$ has $k = O(\log n/\epsilon^2)$ entries and there are m edges, the running time of step 4 is $O(m\log n/\epsilon^2)$. The lemma then follows. □

```
$\tilde{G} = $ **SparsifyApprGreen**($G, q, \beta, \epsilon$):
  1. For all $(u, v) \in E$, compute approximate Green values $\tilde{g}_\beta(e)$ by calling
     **ApproxiamteGreen** ($\beta$, $\kappa$, $k$) where $\kappa = 1/2$ and $k = c\log n/\epsilon^2$.
  2. Apply **SparsifyExactGreen**($G, q, \beta$) with approximate Green values.
```

Theorem 8. *Given constants $\epsilon > 0$, $\phi > 0$ and a graph G on n vertices with m edges, set $q = 256C^2 n \log n/\epsilon^2$, $\beta = \phi/2$, and let $\tilde{G}$ be the output of the algorithm* **SparsifyApprGreen***(G, q, β, ϵ). Then with probability at least 1/2, we have (i) $|h_{\tilde{G}}(S) - h_G(S)| \leq \epsilon h_G(S)$ for all $S \in V$ satisfying $h_G(S) \geq \phi$; (ii) Algorithm* **SparsifyApprGreen** *can be performed by using $O(\frac{m \log^2 n}{\phi})$ preprocessing time and $O(\frac{n \log n}{\epsilon^2})$ sampling.*

The proof is quite similar to the analysis in Section 4 and will be omitted.

6 Partitioning using approximate PageRank vectors

In this section, we combine the graph sparsification and the partitioning algorithms using PageRank vectors to derive an improved partitioning algorithm.

An application of our sparsification algorithm by PageRank is the *balanced cut* problem. For a given graph G, we first use our sparsification algorithm to preprocess the graph. Then we apply the local partitioning algorithm using PageRank vectors [3, 4] on the sparsifier. Since the local partitioning algorithm is a subroutine for the balance cut problem, we obtain a balanced cut algorithm with an improved running time. Spielman and Teng [24] gave a local partitioning algorithm which, for a fixed value of ϕ, gives a cut with approximation ratio $O(\phi^{1/2} \log^{3/2} n)$ and of volume v_ϕ in $O(m \log^c(n)/\phi^{-5/3})$ time where v_ϕ is the largest volume of the set with cheeger ratio ϕ. Note that he constant c above is quite large [24]. In [3], PageRank vectors were used to derive a local partitioning algorithm with an improved running time $(m + n\phi^{-1})O(\text{polylog}(n))$. In [4], the running time was further reduced to $O(m + n\phi^{-1/2})O(\text{polylog}(n))$ by preprocessing using sparsification algorithms in [7].

Given an undirected, weighted graph $G = (V, E, w)$ with n vertices and m edges, we can apply algorithm **SparsifyApprGreen** as a preprocess procedure on graph G to get a sparsifier $\tilde{G}$ with only $O(n \log n/\epsilon^2)$ edges in time $O(m \log^2 n/\phi)$ such that $|h_{\tilde{G}}(S) - h_G(S)| \leq \epsilon h_G(S)$ for all S with $h_G(S) \geq \phi$. Then, we use the algorithm **PageRank-Partition** [3] on graph $\tilde{G}$ instead of G for balanced cut problem. The algorithm **PageRank-Partition** has two inputs including a parameter ϕ and a graph with m edges. As stated in [3], the **PageRank-Partition** algorithm has expected running time $O(m \log^4 m/\phi^2)$. Furthermore, with high probability the **PageRank-Partition** algorithms was shown to be able to find a set S, if exists, such that $\text{vol}(S) \geq vol(C)/2$ and $h_G(S) \leq \phi$. This can be summarized as follows:

Theorem 9. *Given an undirected, weighted graph $G = (V, E, w)$ with n vertices and m edges, constant $\phi > 0$, and $\epsilon > 0$. With probability 1/2, we can preprocess graph G in $O(\frac{m \log^2 n}{\phi})$ time to obtain a sparse graph $\tilde{G}$ with $O(\frac{n \log n}{\epsilon^2})$ edges such that for all $S \in V$ satisfying $h_G(S) \geq \phi$, $|h_{\tilde{G}}(S) - h_G(S)| \leq \epsilon h_G(S)$. Algorithm* **PageRank-Partition** *takes as inputs a parameter ϕ and a graph $\tilde{G}$ and has expected running time $O(n \log^6 m/(\phi^2 \epsilon^2))$. If there exists a set C with $h_G(C) = O(\phi^2/log^2 n)$, then with high probability the* **PageRank-Partition** *algorithm finds a set S such that* $\text{vol}(S) \geq \text{vol}(C)/2$ *and $h_G(S) \leq \phi$.*

References

1. D. Achlioptas, Database-friendly random projections, *PODS 01*, 274–281.
2. S. Arora and S. Kale, A combinatorial, primal-dual approach to semidefinite programs, *STOC 07*, 227–236.
3. R. Andersen, F. Chung, and K. Lang, Local graph partitioning using pagerank vectors, *FOCS 06*, 475–486.
4. R. Andersen and Y. Peres, Finding sparse cuts locally using evolving sets, *STOC 09*, 235–244.
5. S. Arora, E. Hazan, and S. Kale, $\Theta(\sqrt{\log n})$ approximation to sparsest cut in $\tilde{O}(n^2)$ time, *FOCS 04*, 238–247.
6. J. Batson, D. A. Spielman, and N. Srivastava, Twice-Ramanujan sparsifiers, *STOC 09*, 255–262.
7. A. A. Benczúr and D. R. Karger, Approximating s-t minimum cuts in $\tilde{O}(n^2)$ time, *STOC 96*, 47–55.
8. P. Berkhin, Bookmark-coloring approach to personalized pagerank computing, *Internet Mathematics*, **3(1)**, (2007), 41–62
9. S. Brin and L. Page, The anatomy of a large-scale hypertextual Web search engine, *Computer Networks and ISDN Systems*, **30(1-7)**, (1998), 107–117.
10. F. Chung, *Spectal Graph Theory*, AMS Publication, 1997.
11. F. Chung and S.-T. Yau, Discrete Green's Functions, *Journal of Combinatorial Theory, Series A*, **91(1-2)**(2000), 191–214.
12. G. Green, An Essay on the Application of Mathematical Analysis to the Theories of Electricity and Magnetism, Nottingham, 1828.
13. H. Haveliwala, Topic-sensitive pagerank: A context-sensitive ranking algorithm for web search, *IEEE Trans. Knowl. Data Eng.*, **15(4)**, (2003), 784–796.
14. G. Jeh and J. Widom, Scaling personalized web search, *WWW 03*, 271–279.
15. D. R. Karger, Random sampling in cut, flow, and network design problems, *STOC 94*, 648–657.
16. David R. Karger, Using randomized sparsification to approximate minimum cuts, *SODA 94*, 424–432.
17. David R. Karger, Minimum cuts in near-linear time, *JACM*, **47(1)**, (2000), 46–76.
18. L. Loväsz, Random walks on graphs: A survey, *Combinatorics, Paul Erdös is Eighty* **2** (1993), 1–46.
19. L. Lovász and M. Simonovits, The mixing rate of Markov chains, an isoperimetric inequality, and computing the volume, *FOCS 90*, 346–354.
20. L. Orecchia, L. J. Schulman, U. V. Vazirani, and N. K. Vishnoi. On partitioning graphs via single commodity flows, *STOC 08*, 461–470.
21. L. Page, S. Brin, R. Motwani, and T. Winograd, The pagerank citation ranking: Bringing order to the web, Technical report, Stanford Digital Library Technologies Project, 1998.
22. M. Rudelson and R. Vershynin, Sampling from large matrices: An approach through geometric functional analysis, *Journal of the ACM*, **54(4)** (2007).
23. D. A. Spielman and S.-H. Teng. Spectral partitioning works: Planar graphs and finite element meshes, *FOCS 96*, 96–105.
24. D. A. Spielman and S.-H. Teng. Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems. *STOC 04*, 81–90.
25. D. A. Spielman and N. Srivastava, Graph sparsification by effective resistances. *STOC 2008*, 563–568.