Heat Kernel Based Community Detection

Kyle Kloster
Purdue University
West Lafayette, IN
kkloste@purdue.edu

David F. Gleich
Purdue University
West Lafayette, IN
dgleich@purdue.edu

ABSTRACT

The heat kernel is a type of graph diffusion that, like the much-used personalized PageRank diffusion, is useful in identifying a community nearby a starting seed node. We present the first deterministic, local algorithm to compute this diffusion and use that algorithm to study the communities that it produces. Our algorithm is formally a relaxation method for solving a linear system to estimate the matrix exponential in a degree-weighted norm. We prove that this algorithm stays localized in a large graph and has a worst-case constant runtime that depends only on the parameters of the diffusion, not the size of the graph. On large graphs, our experiments indicate that the communities produced by this method have better conductance than those produced by PageRank, although they take slightly longer to compute. On a real-world community identification task, the heat kernel communities perform better than those from the PageRank diffusion.

Categories and Subject Descriptors

G.2.2 [**Discrete mathematics**]: Graph theory—*Graph algorithms*; I.5.3 [**Pattern recognition**]: Clustering—*Algorithms*

General Terms

Algorithms,Theory

Keywords

heat kernel; local clustering

1. INTRODUCTION

The community detection problem is to identify a set of nodes in a graph that are internally cohesive but also separated from the remainder of the network. One popular way to capture this idea is through the conductance measure of a set. We treat this idea formally in the next section, but informally, the conductance of a set is a ratio of the number of edges leaving the set to the number of edges touched by the set of vertices. If this value is small, then it indicates a set with many internal edges and few edges leaving.

In many surveys and empirical studies [47, 33, 17], the conductance measure surfaces as one of the most reliable measures of a community. Although this measure has been critized for producing *cavemen*-type communities [24], empirical properties of real-world communities correlate highly with sets produced by algorithms that optimize conductance [1]. Furthermore, state-of-the-art methods for identifying overlapping sets of communities use conductance to find real-world communities better than any alternative [52].

Virtually all of the rigorous algorithms that identify sets of small conductance are based on min-cuts [6, 46], eigenvector computations [21, 4], or local graph diffusions [5, 14]. (One notable exception is the graclus method [17] that uses a relationship between kernel k-means and a variant of conductance.) In this paper, we study a new algorithm that uses a heat kernel diffusion [14] to identify small-conductance communities in a network. (The heat kernel is discussed formally in Section 3.) Although the properties of this diffusion had been analyzed in theory in Chung's work [14], that work did not provide an efficient algorithm to compute the diffusion. Recently, Chung and Simpson stated a randomized Monte Carlo method to estimate the diffusion [16].

This paper introduces an efficient and deterministic method to estimate this diffusion. We use it to study the properties of the small conductance sets identified by the method as communities. For this use, a deterministic approach is critical as we need to differentiate between subtle properties of the diffusion. Our primary point of comparison is the well-known personalized PageRank diffusion [5], which has been used to establish new insights into the properties of communities in large scale networks [33]. Thus, we wish to understand how the communities produced by the heat kernel compare to those produced by personalized PageRank.

The basic operation of our algorithm is a coordinate relaxation step. This has been used to efficiently compute personalized PageRank [23, 36, 5] where it is known as the "push" operation on a graph; the term "coordinate relaxation" is the classical name for this operation, which dates back to the Gauss-Seidel method. What distinguishes our approach from prior work is the use of coordinate relaxation on an implicitly constructed linear system that will estimate the heat kernel diffusion, which is formally the exponential of the random walk transition matrix.

We began looking into this problem recently in a workshop paper [27], where we showed that this style of algorithm

KDD'14, August 24–27, 2014, New York, NY, USA.

ACM 978-1-4503-2956-9/14/08 ...$15.00.
http://dx.doi.org/10.1145/2623330.2623706.

successfully estimates a related quantity. This paper tackles a fundamentally new direction and, although similar in techniques, required an entirely new analysis. In particular, we are able to estimate this diffusion in constant time in a degree-weighted norm that depends only on the parameters of the diffusion, and not the size of the graph. A Python implementation of our algorithm to accomplish this task is presented in Figure 2.

In the remainder of the paper, we first review these topics formally (Sections 2 and 3); present our algorithm (Section 4) and discuss related work (Section 5). Then we show how our new approach differs from and improves upon the personalized PageRank diffusion in synthetic and real-world problems.

Summary of contributions.

- We propose the first local, deterministic method to accurately compute a heat kernel diffusion in a graph. The code is simple and scalable to any graph where out-link access is *inexpensive*.
- Our method is always localized, even in massive graphs, because it has a provably constant runtime in a degree-weighted norm.
- We compare the new heat kernel diffusion method to the venerable PageRank diffusion in synthetic, small, and large networks (up to 2 billion edges) to demonstrate the differences. On large networks such as Twitter, our method tends to produce smaller and tighter communities. It is also more accurate at detecting ground-truth communities.

We make our experimental codes available in the spirit of reproducible research:
https://www.cs.purdue.edu/homes/dgleich/codes/hkgrow

2. PRELIMINARIES

We begin by fixing our notation. Let $G = (V, E)$ be a simple, undirected graph with $n = |V|$ vertices. Fix an ordering of the vertices from 1 to n such that we can refer to a vertex by it's numeric ID. For a vertex $v \in V$, we denote by d_v the degree of node v. Let $\mathbf{A}$ be the associated adjacency matrix. Because G is undirected, $\mathbf{A}$ is symmetric. Furthermore, let $\mathbf{D}$ be the diagonal matrix of degrees ($\mathbf{D}_{ii} = d_i$) and $\mathbf{P} = (\mathbf{D}^{-1}\mathbf{A})^T = \mathbf{A}\mathbf{D}^{-1}$ be the random walk transition matrix. Finally, we denote by $\mathbf{e}_i$ the vector (of an appropriate length) of zeros having a single 1 in entry i, and by $\mathbf{e}$ the vector of all 1s.

Conductance. Given a subset $S \subseteq V$, we denote by $\mathrm{vol}(S)$ the sum of the degrees of all vertices in S, and by $\partial(S)$, the *boundary* of S, the number of edges with one endpoint inside of S and one endpoint outside of S. With this notation, the conductance of a set S is given by

$$\phi(S) := \frac{\partial(S)}{\min\{\mathrm{vol}(S), \mathrm{vol}(V - S)\}}$$

Conceptually, $\phi(S)$ is the probability that a random walk of length one will land outside of S, given that we start from a node chosen uniformly at random inside S.

Matrix exponential. The heat kernel of a graph involves the matrix exponential, and we wish to briefly mention some facts about this operation. See Higham [22] for a more in-depth treatment. Consider a general matrix $\mathbf{G}$. The exponential function of a matrix is not just the exponential applied element-wise, but rather is given by substituting the matrix into the Taylor series expansion of the exponential function:

$$\exp\{\mathbf{G}\} = \sum_{k=0}^{\infty} \frac{1}{k!}\mathbf{G}^k.$$

That said, the exponential of a diagonal matrix *is* the exponential function applied to the diagonal entries. This phenomenon occurs because powers of a diagonal matrix are simply the diagonal elements raised to the given power. For any matrix $\mathbf{G}$, if $\mathbf{G}^T\mathbf{G} = \mathbf{G}\mathbf{G}^T$ (which is a generalization of the notion of a symmetric matrix, called a *normal* matrix), then $\mathbf{G}$ has an eigenvalue decomposition $\mathbf{G} = \mathbf{X}\mathbf{\Lambda}\mathbf{X}^{-1}$ and there is a simple, albeit inefficient, means of computing the exponential: $\exp\{\mathbf{G}\} = \mathbf{X}\exp\{\mathbf{\Lambda}\}\mathbf{X}^{-1}$. Computing the matrix exponential, or even the *product* of the exponential with a vector, $\exp\{\mathbf{G}\}\mathbf{z}$, is still an active area of research [2].

3. FINDING SMALL CONDUCTANCE COMMUNITIES WITH DIFFUSIONS

A graph diffusion is any sum of the following form:

$$\mathbf{f} = \sum_{k=0}^{\infty} \alpha_k \mathbf{P}^k \mathbf{s}$$

where $\sum_k \alpha_k = 1$ and $\mathbf{s}$ is a stochastic vector (that is, it is nonnegative and sums to one.) Intuitively, a diffusion captures how a quantity of s_i material on node i *flows* through the graph. The terms α_k provide a decaying weight that ensures that the diffusion eventually dissipates. In the context of this paper, we are interested in the diffusions of single nodes or neighborhood sets of a single vertex; and so in these cases $\mathbf{s} = \mathbf{e}_i$ or $\mathbf{s} = \sum_{i \in S} \mathbf{e}_i/|S|$ for a small set S. We call the origins of the diffusion the *seeds*.

Given an estimate of a diffusion $\mathbf{f}$ from a seed, we can produce a small conductance community from it using a sweep procedure. This involves computing $\mathbf{D}^{-1}\mathbf{f}$, sorting the nodes in descending order by their magnitude in this vector, and computing the conductance of each prefix of the sorted list. Due to the properties of conductance, there is an efficient means of computing all of these conductances. We then return the set of smallest conductance as the community around the seeds.

The personalized PageRank diffusion. One of the most well-known instances of this framework is the personalized PageRank diffusion. Fix $\alpha \in (0, 1)$. Then $\mathbf{p}$ is defined:

$$\mathbf{p} = (1 - \alpha)\sum_{k=0}^{\infty} \alpha^k \mathbf{P}^k \mathbf{s}. \tag{1}$$

The properties of this diffusion have been studied extensively. In particular, Andersen et al. [5] establish a local Cheeger inequality using a particular algorithm called "push" that locally distributes mass. The local Cheeger inequality informally states that, *if* the seed is nearby a set with small conductance, *then* the result of the sweep procedure is a set with a related conductance. Moreover, they show that their "push" algorithm estimates $\mathbf{f}$ with an error ε in a degree-weighted norm by looking at $\frac{1}{(1-\alpha)\varepsilon}$ edges.

The heat kernel diffusion. Another instance of the same framework is the heat kernel diffusion [14, 15]. It

simply replaces the weights α_k with $t^k/k!$:

$$\mathbf{h} = e^{-t}\left(\sum_{k=0}^{\infty}\frac{t^k}{k!}(\mathbf{P})^k\right)\mathbf{s} = \exp\left\{-t(\mathbf{I}-\mathbf{P}^{-1})\right\}\mathbf{s}. \quad (2)$$

While it was known that estimating $\mathbf{h}$ gave rise to a similar type of local Cheeger inequality [15]; until Chung and Simpson's Monte Carlo approach [16], no methods were known to estimate this quantity efficiently. Our new algorithm is a deterministic approach that is more suitable for comparing the properties of the diffusions. It terminates after exploring $\frac{2Ne^t}{\varepsilon}$ edges (Theorem 1), where N is a parameter that grows slowly with ε.

Heat kernels compared to PageRank. These different sets of coefficients simply assign different levels of importance to walks of varying lengths: the heat kernel coefficients $\frac{t^k}{k!}$ decay much more quickly than α^k, and so the heat kernel more heavily weights shorter walks in the overall sum (depicted in Figure 1). This property, in turn, will have important consequences when we study these methods in large graphs in Section 6.

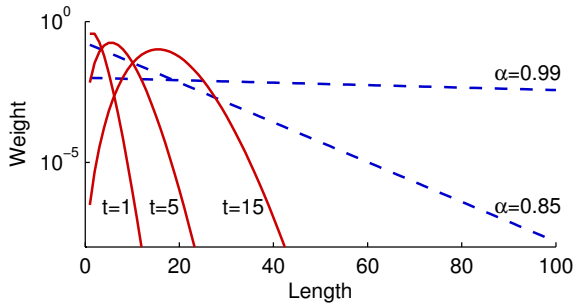


Figure 1: Each curve represents the coefficients of $(\mathbf{AD}^{-1})^k$ in a sum of walks. The dotted blue lines give α^k, and the red give $t^k/k!$, for the indicated values of α and t.

4. ALGORITHM

The overall idea of the local clustering algorithm is to approximate a heat kernel vector of the form

$$\mathbf{h} = \exp\left\{-t(\mathbf{I}-\mathbf{P})\right\}\mathbf{s}$$

so that we can perform a sweep over $\mathbf{h}$. Here we describe a coordinate-relaxation method, which we call `hk-relax`, for approximating $\mathbf{h}$. This algorithm is rooted in our recent work on computing an accurate column of $\exp\{\mathbf{P}\}$ [27]; but is heavily tuned to the objective below. Thus, while the overall strategy is classical – just as the PageRank push method is a classic relaxation method – the simplifications and efficient implementation are entirely novel. In particular, the new objective in this paper enables us to get a constant runtime bound independent of any property of the graph, which differs markedly from both of our previous methods [27, 26].

Our objective. Recall that the final step of finding a small conductance community involves dividing by the degree of each node. Thus, our goal is to compute $\mathbf{x} \approx \mathbf{h}$ satisfying the degree weighted bound:

$$\|\mathbf{D}^{-1}\exp\left\{-t(\mathbf{I}-\mathbf{P})\right\}\mathbf{s} - \mathbf{D}^{-1}\mathbf{x}\|_\infty < \varepsilon.$$

By using standard properties of the matrix exponential, we can factor $\exp\{-t(\mathbf{I}-\mathbf{P})\} = e^{-t}\exp\{t\mathbf{P}\}$ and scale by e^t so that the above problem is equivalent to computing $\mathbf{y}$ satisfying $\|\mathbf{D}^{-1}(\exp\{t\mathbf{P}\}\mathbf{s} - \mathbf{y})\|_\infty < e^t\varepsilon$. The element-wise characterization is that $\mathbf{y}$ must satisfy:

$$|e^t\mathbf{h}_i - \mathbf{y}_i| < e^t\varepsilon d_i \quad (3)$$

for all i. A similar weighted objective was used in the push algorithm for PageRank [5].

Outline of algorithm. To accomplish this, we first approximate $\exp\{t\mathbf{P}\}$ with its degree N Taylor polynomial, $T_N(t\mathbf{P})$, and then we compute $T_N(t\mathbf{P})\mathbf{s}$. But we use a large, implicitly formed linear system to avoid explicitly evaluating the Taylor polynomial. Once we have the linear system, we state a relaxation method in the spirit of Gauss-Seidel and the PageRank push algorithm in order to compute an accurate approximation of $\mathbf{h}$.

4.1 Taylor Polynomial for exp{X}

Determining the exponential of a matrix is a sensitive computation with a rich history [39, 40]. For a general matrix $\mathbf{G}$, an approximation via the Taylor polynomial,

$$\exp\{\mathbf{G}\} = \sum_{k=0}^{\infty}\tfrac{1}{k!}\mathbf{G}^k \approx \sum_{k=0}^{N}\tfrac{1}{k!}\mathbf{G}^k,$$

can be inaccurate when $\|\mathbf{G}\|$ is large and $\mathbf{G}$ has mixed signs, as large powers $\mathbf{G}^k$ can contain large, oppositely signed numbers that cancel properly only in exact arithmetic. However, we intend to compute $\exp\{t\mathbf{P}\}\mathbf{s}$, where $\mathbf{P}$, t, and $\mathbf{s}$ are nonnegative, so the calculation does not rely on any delicate cancellations. Furthermore, our approximation need not be highly precise. We therefore use the polynomial $\exp\{t\mathbf{P}\}\mathbf{s} \approx T_N(t\mathbf{P}) = \sum_{k=0}^{N}\frac{t^k}{k!}\mathbf{P}^k$ for our approximation. For details on choosing N, see Section 4.5. For now, assume that we have chosen N such that

$$\|\mathbf{D}^{-1}\exp\{t\mathbf{P}\}\mathbf{s} - \mathbf{D}^{-1}T_N(t\mathbf{P})\mathbf{s}\|_\infty < \varepsilon/2. \quad (4)$$

This way, if we compute $\mathbf{y} \approx T_N(t\mathbf{P})\mathbf{s}$ satisfying

$$\|\mathbf{D}^{-1}T_N(t\mathbf{P})\mathbf{s} - \mathbf{D}^{-1}\mathbf{y}\|_\infty < \varepsilon/2,$$

then by the triangle inequality we will have

$$\|\mathbf{D}^{-1}\exp\{t\mathbf{P}\}\mathbf{s} - \mathbf{D}^{-1}\mathbf{y}\|_\infty < \varepsilon, \quad (5)$$

our objective.

4.2 Error weights

Using a degree N Taylor polynomial, `hk-relax` ultimately approximates $\mathbf{h}$ by approximating each term in the sum of the polynomial times the vector $\mathbf{s}$:

$$\mathbf{s} + \tfrac{t}{1}\mathbf{P}\mathbf{s} + \cdots + \tfrac{t^N}{N!}\mathbf{P}^N\mathbf{s}.$$

The total error of our computed solution is then a weighted sum of the errors at each individual term, $\frac{t^k}{k!}\mathbf{P}^k\mathbf{s}$. We show in Lemma 1 that these weights are given by the polynomials $\psi_k(t)$, which we define now. For a fixed degree N Taylor polynomial of the exponential, $T_N = \sum_{k=0}^{N}\frac{t^k}{k!}$, we define

$$\psi_k := \sum_{m=0}^{N-k}\frac{k!}{(m+k)!}t^m \text{ for } k = 0, \cdots, N. \quad (6)$$

These polynomials $\psi_k(t)$ are closely related to the ϕ functions central to exponential integrators in ODEs [37]. Note that $\psi_0 = T_N$.

To guarantee the total error satisfies the criterion (3) then, it is enough to show that the error at each Taylor term satisfies an ∞-norm inequality analogous to (3). This is discussed in more detail in Section A.

4.3 Deriving a linear system

To define the basic step of the `hk-relax` algorithm and to show how the ψ_k influence the total error, we rearrange the Taylor polynomial computation into a linear system.

Denote by $\mathbf{v}_k$ the k^{th} term of the vector sum $T_N(t\mathbf{P})\mathbf{s}$:

$$T_N(t\mathbf{P})\mathbf{s} = \mathbf{s} + \tfrac{t}{1}\mathbf{P}\mathbf{s} + \cdots + \tfrac{t^N}{N!}\mathbf{P}^N\mathbf{s} \quad (7)$$

$$= \mathbf{v}_0 + \mathbf{v}_1 + \cdots + \mathbf{v}_N. \quad (8)$$

Note that $\mathbf{v}_{k+1} = \frac{t^{k+1}}{(k+1)!}\mathbf{P}^{k+1} = \frac{t}{(k+1)}\mathbf{P}\mathbf{v}_k$. This identity implies that the terms $\mathbf{v}_k$ exactly satisfy the linear system

$$\begin{bmatrix} \mathbf{I} & & & & \\ \frac{-t}{1}\mathbf{P} & \mathbf{I} & & & \\ & \frac{-t}{2}\mathbf{P} & \ddots & & \\ & & \ddots & \mathbf{I} & \\ & & & \frac{-t}{N}\mathbf{P} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{v}_0 \\ \mathbf{v}_1 \\ \vdots \\ \vdots \\ \mathbf{v}_N \end{bmatrix} = \begin{bmatrix} \mathbf{s} \\ 0 \\ \vdots \\ \vdots \\ 0 \end{bmatrix}. \quad (9)$$

Let $\mathbf{v} = [\mathbf{v}_0; \mathbf{v}_1; \cdots; \mathbf{v}_N]$. An approximate solution $\hat{\mathbf{v}}$ to (9) would have block components $\hat{\mathbf{v}}_k$ such that $\sum_{k=0}^{N} \hat{\mathbf{v}}_k \approx T_N(t\mathbf{P})\mathbf{s}$, our desired approximation to $e^t\mathbf{h}$. In practice, we update only a single length n solution vector, adding all updates to that vector, instead of maintaining the $N+1$ different block vectors $\hat{\mathbf{v}}_k$ as part of $\hat{\mathbf{v}}$; furthermore, the block matrix and right-hand side are never formed explicitly.

With this block system in place, we can describe the algorithm's steps.

4.4 The hk-relax algorithm

Given a random walk transition matrix $\mathbf{P}$, scalar $t > 0$, and seed vector $\mathbf{s}$ as inputs, we solve the linear system from (9) as follows. Denote the initial solution vector by $\mathbf{y}$ and the initial $nN \times 1$ residual by $\mathbf{r}^{(0)} = \mathbf{e}_1 \otimes \mathbf{s}$. Denote by $r(i,j)$ the entry of $\mathbf{r}$ corresponding to node i in residual block j. The idea is to iteratively remove all entries from $\mathbf{r}$ that satisfy

$$r(i,j) \geq \frac{e^t \varepsilon d_i}{2N\psi_j(t)}. \quad (10)$$

To organize this process, we begin by placing the nonzero entries of $\mathbf{r}^{(0)}$ in a queue, $Q(\mathbf{r})$, and place updated entries of $\mathbf{r}$ into $Q(\mathbf{r})$ only if they satisfy (10).

Then `hk-relax` proceeds as follows.

1. At each step, pop the top entry of $Q(\mathbf{r})$, call it $r(i,j)$, and subtract that entry in $\mathbf{r}$, making $\mathbf{r}(i,j) = 0$.
2. Add $r(i,j)$ to $\mathbf{y}_i$.
3. Add $r(i,j)\frac{t}{j+1}\mathbf{P}\mathbf{e}_j$ to residual block $\mathbf{r}_{j+1}$.
4. For each entry of $\mathbf{r}_{j+1}$ that was updated, add that entry to the back of $Q(\mathbf{r})$ if it satisfies (10).

Once all entries of $\mathbf{r}$ that satisfy (10) have been removed, the resulting solution vector $\mathbf{y}$ will satisfy (3), which we prove in Section A, along with a bound on the work required to achieve this. We present working code for our method in Figure 2 that shows how to optimize this computation using sparse data structures. These make it highly efficient in practice.

```
# G is graph as dictionary-of-sets,
# seed is an array of seeds,
# t, eps, N, psis are precomputed
x = {} # Store x, r as dictionaries
r = {} # initialize residual
Q = collections.deque() # initialize queue
for s in seed:
  r[(s,0)] = 1./len(seed)
  Q.append((s,0))
while len(Q) > 0:
  (v,j) = Q.popleft() # v has r[(v,j)] ...
  rvj = r[(v,j)]
  # perform the hk-relax step
  if v not in x: x[v] = 0.
  x[v] += rvj
  r[(v,j)] = 0.
  mass = (t*rvj/(float(j)+1.))/len(G[v])
  for u in G[v]:    # for neighbors of v
    next = (u,j+1) # in the next block
    if j+1 == N: # last step, add to soln
      x[u] += rvj/len(G(v))
      continue
    if next not in r: r[next] = 0.
    thresh = math.exp(t)*eps*len(G[u])
    thresh = thresh/(N*psis[j+1])/2.
    if r[next] < thresh and \
       r[next] + mass >= thresh:
       Q.append(next) # add u to queue
    r[next] = r[next] + mass
```

Figure 2: Pseudo-code for our algorithm as working python code. The graph is stored as a dictionary of sets so that the `G[v]` statement returns the set of neighbors associated with vertex v. The solution is the vector x indexed by vertices and the residual vector is indexed by tuples (v,j) that are pairs of vertices and steps j. A fully working demo may be downloaded from github `https://gist.github.com/dgleich/7d904a10dfd9ddfaf49a`.

4.5 Choosing N

The last detail of the algorithm we need to discuss is how to pick N. In (4) we want to guarantee the accuracy of $\mathbf{D}^{-1}\exp\{t\mathbf{P}\}\mathbf{s} - \mathbf{D}^{-1}T_N(t\mathbf{P})\mathbf{s}$. By using $\mathbf{D}^{-1}\exp\{t\mathbf{P}\} = \exp\left\{t\mathbf{P}^T\right\}\mathbf{D}^{-1}$ and $\mathbf{D}^{-1}T_N(t\mathbf{P}) = T_N(t\mathbf{P}^T)\mathbf{D}^{-1}$, we can get a new upperbound on $\|\mathbf{D}^{-1}\exp\{t\mathbf{P}\}\mathbf{s} - \mathbf{D}^{-1}T_N(t\mathbf{P})\mathbf{s}\|_\infty$ by noting

$$\|\exp\left\{t\mathbf{P}^T\right\}\mathbf{D}^{-1}\mathbf{s} - T_N(t\mathbf{P}^T)\mathbf{D}^{-1}\mathbf{s}\|_\infty$$
$$\leq \|\exp\left\{t\mathbf{P}^T\right\} - T_N(t\mathbf{P}^T)\|_\infty \|\mathbf{D}^{-1}\mathbf{s}\|_\infty.$$

Since $\mathbf{s}$ is stochastic, we have $\|\mathbf{D}^{-1}\mathbf{s}\|_\infty \leq \|\mathbf{D}^{-1}\mathbf{s}\|_1 \leq 1$. From [34] we know that the norm $\|\exp\left\{t\mathbf{P}^T\right\} - T_N(t\mathbf{P}^T)\|_\infty$ is bounded by

$$\frac{\|t\mathbf{P}^T\|_\infty^{N+1}}{(N+1)!}\frac{(N+2)}{(N+2-t)} \leq \frac{t^{N+1}}{(N+1)!}\frac{(N+2)}{(N+2-t)}. \quad (11)$$

So to guarantee (4), it is enough to choose N that implies $\frac{t^{N+1}}{(N+1)!}\frac{(N+2)}{(N+2-t)} < \varepsilon/2$. Such an N can be determined efficiently simply by iteratively computing terms of the Taylor polynomial for e^t until the error is less than the desired error for `hk-relax`. In practice, this required a choice of N no greater than $2t\log(\frac{1}{\varepsilon})$, which we think can be made rigorous.

4.6 Outline of convergence result

The proof proceeds as follows. First, we relate the error vector of the Taylor approximation $E_1 = T_N(t\mathbf{P})\mathbf{s} - \mathbf{x}$, to the error vector from solving the linear system described in Section 4.4, $E_2 = T_N(t\mathbf{P})\mathbf{s} - \mathbf{y}$. Second, we express the error vector E_2 in terms of the residual blocks of the linear system (9); this will involve writing E_2 as a sum of residual blocks $\mathbf{r}_k$ with weights $\psi_k(t\mathbf{P})$. Third, we use the previous results to upper-bound $\|\mathbf{D}^{-1}T_N(t\mathbf{P})\mathbf{s} - \mathbf{D}^{-1}\mathbf{x}\|_\infty$ with $\sum_{k=0}^{N}\psi_k(t)\|\mathbf{D}^{-1}\mathbf{r}_k\|_\infty$, and use this to show that $\|\mathbf{D}^{-1}T_N(t\mathbf{P})\mathbf{s} - \mathbf{D}^{-1}\mathbf{x}\|_\infty < \varepsilon/2$ is guaranteed by the stopping criterion of `hk-relax`, (3). Finally, we prove that performing steps of `hk-relax` until the stopping criterion is attained requires work bounded by $\frac{2N\psi_1(t)}{\varepsilon} \leq 2Ne^t/\varepsilon$.

5. RELATED WORK

We wish to highlight a few ideas that have recently emerged in the literature to clarify how our method differs. We discuss these in terms of community detection, the matrix exponential, fast diffusion methods, and relaxation methods.

Community detection and conductance. Conductance often appears in community detection and is known to be one of the most important measures of a community [47]. The personalized PageRank method is one of the most scalable methods to find sets of small conductance, although recent work opened up a new possibility with localized max-flow algorithms [46]. For the PageRank algorithm we use as a point of comparison, Zhu et al. [54] recently provided an improved bound on the performance of this algorithm when finding sets with high internal conductance. The internal conductance of a set is the minimum conductance of the subgraph induced by the set and we would expect that real-world communities have large internal conductance. Due to the similarity between our algorithm and the personalized PageRank diffusion, we believe that a similar result likely holds here as well.

The matrix exponential in network analysis. Recently, the matrix exponential has frequently appeared as a tool in the network analysis literature. It has been used to estimate node centrality [18, 20, 19], for link-prediction [29], in graph kernels [28], and – as already mentioned – clustering and community detection [14]. Many of these studies involve fast ways to approximate the *entire* matrix exponential, instead of a single column as we study here. For instance, Sui et al. [49] describe a low-parameter decomposition of a network that is useful both for estimating Katz scores [25] and the matrix exponential. Orecchia and Mahoney [44] show that the heat kernel diffusion implicitly approximates a diffusion operator using a particular type of generalized entropy, which provides a principled rationale for its use.

Fast methods for diffusions. Perhaps the most related work is a recent Monte Carlo algorithm by Chung and Simpson [16] to estimate the heat kernel diffusion via a random walk sampling scheme. This approach involves directly simulating a random walk with transition probabilities that mirror the Taylor series expansion of the exponential. In comparison, our approach is entirely deterministic and thus more useful to compare between diffusions, as it eliminates the algorithmic variance endemic to Monte Carlo simulations. A similar idea is used by Borgs et al. [13] to achieve a randomized sublinear time algorithm to estimate the largest PageRank entries, and in fact, Monto Carlo methods frequently feature in PageRank computations due to the relationship between the diffusion and a random walk [7, 13, 8, 9]. Most other deterministic approaches for the matrix exponential involve at least one matrix-vector product [45, 2].

Relaxation methods. The algorithm we use is a coordinate relaxation method similar to Gauss-Seidel and Gauss-Southwell. If we applied it to a symmetric positive definite matrix, then it would be a coordinate descent method [35]. It has been proposed for PageRank in a few difference cases [23, 36, 5]. The same type of relaxation method has also been used to estimate the Katz diffusion [12]. We recently used it to estimate a column of the matrix exponential $\exp\{\mathbf{P}\}\,\mathbf{e}_i$ in a strict, 1-norm error and were able to prove a sublinear convergence bound by assuming a very slowly growing maximum degree [27] or a power-law degree distribution [26]. This paper, in comparision, treats the scaled exponential $\exp\{-t(\mathbf{I}-\mathbf{P})\}\,\mathbf{e}_i$ in a degree-weighted norm; it also shows a constant runtime independent of any network property.

6. EXPERIMENTAL RESULTS

Here we compare `hk-relax` with a PageRank-based local clustering algorithm, pprpush [5]. Both algorithms accept as inputs a symmetric graph $\mathbf{A}$ and seed set $\mathbf{s}$. The parameters required are t and ε, for `hk-relax`, and α and ε for pprpush. Both algorithms compute their respective diffusion ranks starting from the seed set, then perform a sweep-cut on the resulting ranks. The difference in the algorithms lies solely in the diffusion used and the particular parameters. We conducted the timing experiments on a Dual CPU system with the Intel Xeon E5-2670 processor (2.6 GHz, 8 cores) with 16 cores total and 256 GB of RAM. None of the experiments needed anywhere near all the memory, nor any of the parallelism. Our implementation uses Matlab's sparse matrix data structure through a C++ mex interface. It uses C++ unordered maps to store sparse vectors and is equivalent to the code in Figure 2.

6.1 Synthetic results

In this section, we study the behavior of the PageRank and heat kernel diffusions on the symbolic image graph of a chaotic function f [48]. The graphs that result are loosely reminiscent of a social network because they have pockets of structure, like communities, and also chaotic behaviour that results in a small-world like property.

The symbolic image of a function f is a graph where each node represents a region of space, and edges represent the action of the function in that region of space. We consider two-dimensional functions $f(x, y)$ in the unit square $[0, 1]^2$ so that we can associate each node with a pixel of an image and illustrate the vectors as images. In Figure 3 (left), we illustrate how the graph construction works. In the remaining examples, we let f be a map that results from a chaotic, nonlinear dynamical system [48] (we use the T10 construction with $k = 0.22, \eta = 0.99$ and sample 1000 points from each region of space, then symmetrize the result and discard weights). We discretize space in a 512×512 grid, which results in a $262,144$ node graph with $2M$ edges. In Figure 3 (right), we also show the PageRank vector with uniform teleportation as an image to "illustrate the structure" in the function f.

Next, in Figure 4, we compare the vectors and sets identified by both diffusions starting from a single seed node. We chose the parameters $\alpha = 0.85$ and $t = 3$ so the two

methods perform the same amount of work. These results are what would be expected from Figure 1, and what many of the remaining experiments show: PageRank diffuses to a larger region of the graph whereas the heat-kernel remains more focused in a sub-region. PageRank, then, finds a large community with about 5,000 nodes whereas the heat kernel finds a small community with around 452 nodes with slightly worse conductance. This experiment suggests that, if these results also hold in real-world networks, then because real-world communities are often small [33], the heat kernel diffusion should produce *more accurate* communities in real-world networks.

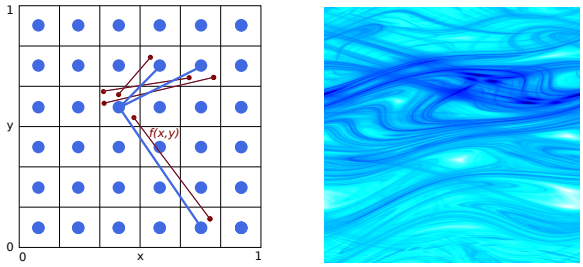
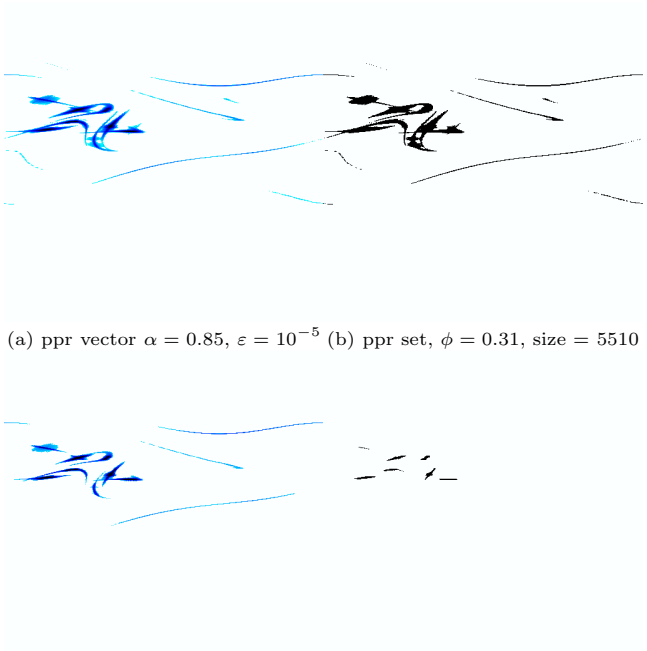


Figure 3: (Left) An illustration of the symbolic image of a function f as a graph. Each large, blue node represents a region of space. Thick, blue edges represent how the thin, red values of the function behave in that region. (Right) The global PageRank vector is then an image that illustrates features of the chaotic map f and shows that it has pockets of structure.

6.2 Runtime and conductance

We next compare the runtime and conductance of the algorithms on a suite of social networks. For pprpush, we fix $\alpha = 0.99$, then compute PageRank for multiple values of $\varepsilon = 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}$, and output the set of best conductance obtained. (This matches the way this method is commonly used in past work.) For `hk-relax`, we compute the heat kernel rank for four different parameter sets, and output the set of best conductance among them: $(t, \varepsilon) = (10, 10^{-4}); (20, 10^{-3}); (40, 5 \cdot 10^{-3}); (80, 10^{-2})$. We also include in `hk-relax` an early termination criterion, in the case that the sum of the degrees of the nodes which are relaxed, $\sum d_{i_l}$, exceeds $n^{1.5}$. However, even the smaller input graphs (on which the condition is more likely to be met because of the smaller value of $n^{1.5}$) do not appear to have reached this threshold. Furthermore, the main theorem of this paper implies that the quantity $\sum d_{i_l}$ cannot exceed $\frac{2N\psi_1(t)}{\varepsilon}$. The datasets we use are summarized in Table 1; all datasets are modified to be undirected and a single connected component. These datasets were originally presented in the following papers [3, 10, 32, 41, 42, 33, 31, 50, 30, 53, 11, 38].

To compare the runtimes of the two algorithms, we display in Figure 5 for each graph the 25%, 50%, and 75% percentiles of the runtimes from 200 trials performed. For a given graph, each trial consisted of choosing a node of that graph uniformly at random to be a seed, then calling both the PageRank and the heat kernel algorithms. On the larger datasets, which have a much broader spectrum of node degrees and therefore greater variance, we instead performed 1,000 trials. Additionally, we display the 25%, 50%, and 75% percentiles



(a) ppr vector $\alpha = 0.85$, $\varepsilon = 10^{-5}$ (b) ppr set, $\phi = 0.31$, size = 5510

(c) hk vector $t = 3$, $\varepsilon = 10^{-5}$ (d) hk set, $\phi = 0.34$, size = 455

Figure 4: When we compare the heat-kernel and PageRank diffusions on the symbolic image of the Chirikov map (see Figure 3), pprgrow finds a larger set with slightly better conductance, whereas hk-grow finds a tighter set with about the same conductance. In real-world networks, these smaller sets are more like real-world communities.

of the conductances achieved during the exact same set of trials. The trendlines of these figures omit some of the trials in order to better show the trends, but all of the median results are plotted (as open circles). The figures show that in small graphs, `hk-relax` is faster than pprpush, but gets larger (worse) conductance. The picture reverses for large graphs where `hk-relax` is slower but finds smaller (better) conductance sets.

Cluster size and conductance. We highlight the individual results of the previous experiment on the symmetrized twitter network. Here, we find that `hk-relax` finds sets of better conductance than pprpush at all sizes of communities in the network. See Figure 6.

6.3 Clusters produced vs. ground-truth

We conclude with an evaluation of identifying ground-truth communities in the `com-dblp`, `com-lj`, `com-amazon`, `com-orkut`, `com-youtube`, and `com-friendster` datasets [53, 38]. In this experiment, for each dataset we first located 100 known communities in the dataset of size greater than 10. Given one such community, using every single node as an individual seed, we looked at the sets returned by `hk-relax` with $t = 5, \varepsilon = 10^{-4}$ and pprpush using the standard procedure. We picked the set from the seed that had the highest F_1 measure. (Recall that the F_1 measure is a harmonic mean of precision and recall.) We report the mean of the F_1 measure, conductance, and set size, where the average is taken over all 100 trials in Table 2. These results show that `hk-relax` produces only slightly inferior conductance

Table 1: Datasets

Graph	$\|V\|$	$\|E\|$
pgp-cc	10,680	24,316
ca-AstroPh-cc	17,903	196,972
marvel-comics-cc	19,365	96,616
as-22july06	22,963	48,436
rand-ff-25000-0.4	25,000	56,071
cond-mat-2003-cc	27,519	116,181
email-Enron-cc	33,696	180,811
cond-mat-2005-fix-cc	36,458	171,735
soc-sign-epinions-cc	119,130	704,267
itdk0304-cc	190,914	607,610
dblp-cc	226,413	716,460
flickr-bidir-cc	513,969	3,190,452
ljournal-2008	5,363,260	50,030,085
twitter-2010	41,652,230	1,202,513,046
friendster	65,608,366	1,806,067,135
com-amazon	334,863	925,872
com-dblp	317,080	1,049,866
com-youtube	1,134,890	2,987,624
com-lj	3,997,962	34,681,189
com-orkut	3,072,441	117,185,083

Table 2: The result of evaluating the heat kernel (hk) vs. PageRank (pr) on finding real-world communities. The heat kernel finds smaller, more accurate, sets with slightly worse conductance.

data	F_1-measure		conductance		set size	
	hk	pr	hk	pr	hk	pr
amazon	0.325	0.140	0.141	0.048	193	15293
dblp	0.257	0.115	0.267	0.173	44	16026
youtube	0.177	0.136	0.337	0.321	1010	6079
lj	0.131	0.107	0.474	0.459	283	738
orkut	0.055	0.044	0.714	0.687	537	1989
friendster	0.078	0.090	0.785	0.802	229	333

scores, but using much smaller sets with substantially better F_1 measures. This suggests that `hk-relax` better captures the properties of real-world communities than the PageRank diffusion in the sense that the tighter sets produced by the heat kernel are better focused around real-world communities than are the larger sets produced by the PageRank diffusion.

7. CONCLUSIONS

These results suggest that the `hk-relax` algorithm is a viable companion to the celebrated PageRank push algorithm and may even be a worthy competitor for tasks that require accurate communities of large graphs. Furthermore, we suspect that the `hk-relax` method will be useful for the myriad other uses of PageRank-style diffusions such as link-prediction [29] or even logic programming [51].

In the future, we plan to explore this method on directed networks as well as better methods for selecting the parameters of the diffusion. It is also possible that our new ideas may translate to faster methods for non-conservative diffusions such as the Katz [25] and modularity methods [43], and we plan to explore these diffusions as well.

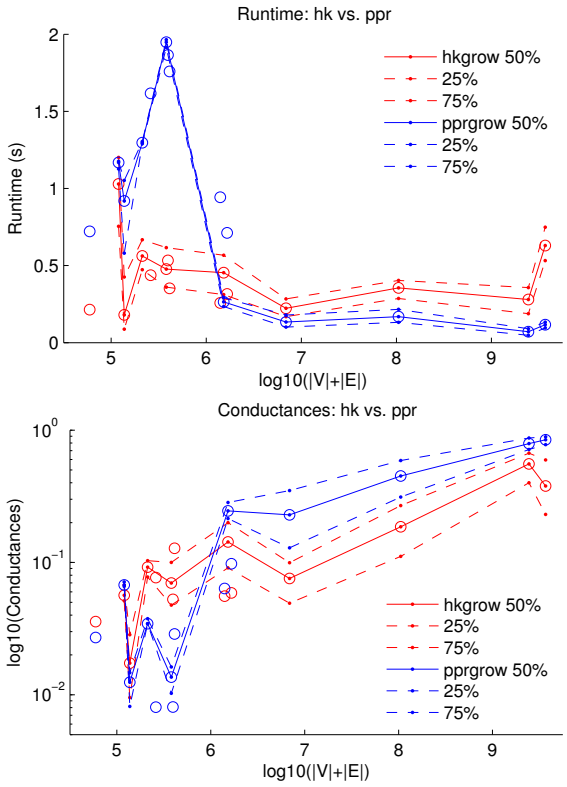


Figure 5: (Top figure) Runtimes of the hk-relax vs. ppr-push, shown with percentile trendlines from a select set of experiments. (Bottom) Conductances of hk-relax vs. ppr-push, shown in the same way.

Acknowledgements

This work was supported by NSF CAREER Award CCF-1149756.

8. REFERENCES

[1] B. Abrahao, S. Soundarajan, J. Hopcroft, and R. Kleinberg. On the separability of structural classes of communities. In *KDD*, pages 624–632, 2012.

[2] A. H. Al-Mohy and N. J. Higham. Computing the action of the matrix exponential, with an application to exponential integrators. *SIAM J. Sci. Comput.*, 33(2):488–511, March 2011.

[3] R. Alberich, J. Miro-Julia, and F. Rossello. Marvel universe looks almost like a real social network. *arXiv*, cond-mat.dis-nn:0202174, 2002.

[4] N. Alon and V. D. Milman. λ_1, isoperimetric inequalities for graphs, and superconcentrators. *J. of Comb. Theory, Series B*, 38(1):73–88, 1985.

[5] R. Andersen, F. Chung, and K. Lang. Local graph partitioning using PageRank vectors. In *FOCS*, 2006.

[6] R. Andersen and K. Lang. An algorithm for improving graph partitions. In *SODA*, pages 651–660, January 2008.

[7] 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(2):890–904, February 2007.

[8] B. Bahmani, K. Chakrabarti, and D. Xin. Fast personalized pagerank on MapReduce. In *SIGMOD*, pages 973–984, New York, NY, USA, 2011. ACM.

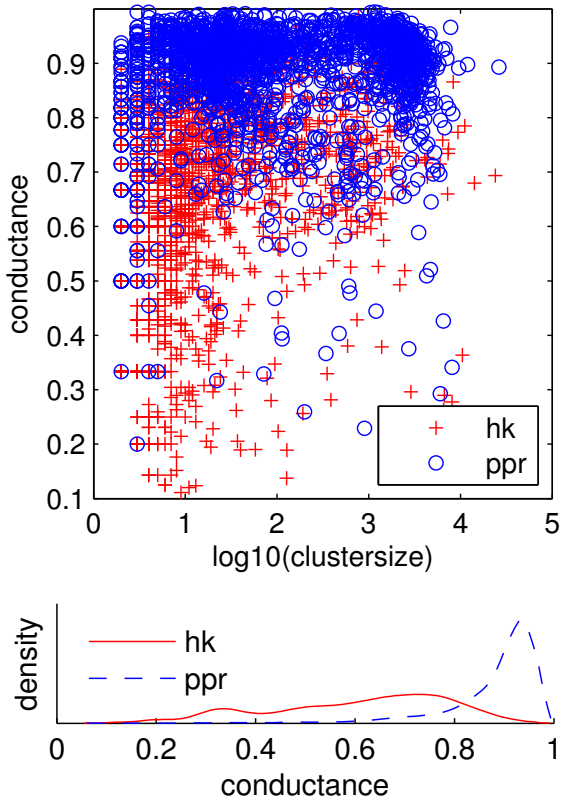


Figure 6: The top figure shows a scatter plot of conductance vs. community size in the twitter graph for the two community detection methods; the bottom figure shows a kernel density estimate of the conductances achieved by each method, which shows that hk-relax is more likely to return a set of lower conductance.

[9] B. Bahmani, A. Chowdhury, and A. Goel. Fast incremental and personalized pagerank. *Proc. VLDB Endow.*, 4(3):173–184, Dec. 2010.

[10] M. Boguñá, R. Pastor-Satorras, A. Díaz-Guilera, and A. Arenas. Models of social networks based on social distance attachment. *Phys. Rev. E*, 70(5):056122, Nov 2004.

[11] P. Boldi, M. Rosa, M. Santini, and S. Vigna. Layered label propagation: A multiresolution coordinate-free ordering for compressing social networks. In *WWW*, pages 587–596, March 2011.

[12] F. Bonchi, P. Esfandiar, D. F. Gleich, C. Greif, and L. V. Lakshmanan. Fast matrix computations for pairwise and columnwise commute times and Katz scores. *Internet Mathematics*, 8(1-2):73–112, 2012.

[13] C. Borgs, M. Brautbar, J. Chayes, and S.-H. Teng. Multi-scale matrix sampling and sublinear-time pagerank computation. *Internet Mathematics*, Online, 2013.

[14] F. Chung. The heat kernel as the PageRank of a graph. *PNAS*, 104(50):19735–19740, 2007.

[15] F. Chung. A local graph partitioning algorithm using heat kernel pagerank. *Internet Mathematics*, 6(3):315–330, 2009.

[16] F. Chung and O. Simpson. Solving linear systems with boundary conditions using heat kernel pagerank. In *Algorithms and Models for the Web Graph*, pages 203–219. Springer, 2013.

[17] I. S. Dhillon, Y. Guan, and B. Kulis. Weighted graph cuts without eigenvectors a multilevel approach. *IEEE Trans. Pattern Anal. Mach. Intell.*, 29(11):1944–1957, November 2007.

[18] E. Estrada. Characterization of 3d molecular structure. *Chemical Physics Letters*, 319(5-6):713–718, 2000.

[19] E. Estrada and D. J. Higham. Network properties revealed through matrix functions. *SIAM Review*, 52(4):696–714, 2010.

[20] A. Farahat, T. LoFaro, J. C. Miller, G. Rae, and L. A. Ward. Authority rankings from HITS, PageRank, and SALSA: Existence, uniqueness, and effect of initialization. *SIAM Journal on Scientific Computing*, 27(4):1181–1201, 2006.

[21] M. Fiedler. Algebraic connectivity of graphs. *Czechoslovak Mathematical Journal*, 23(98):298–305, 1973.

[22] N. J. Higham. *Functions of Matrices: Theory and Computation*. SIAM, 2008.

[23] G. Jeh and J. Widom. Scaling personalized web search. In *WWW*, pages 271–279. ACM, 2003.

[24] U. Kang and C. Faloutsos. Beyond 'caveman communities': Hubs and spokes for graph compression and mining. In *ICDM*, pages 300–309, Washington, DC, USA, 2011. IEEE Computer Society.

[25] L. Katz. A new status index derived from sociometric analysis. *Psychometrika*, 18(1):39–43, March 1953.

[26] K. Kloster and D. F. Gleich. A fast relaxation method for computing a column of the matrix exponential of stochastic matrices from large, sparse networks. *arXiv*, cs.SI:1310.3423, 2013.

[27] K. Kloster and D. F. Gleich. A nearly-sublinear method for approximating a column of the matrix exponential for matrices from large, sparse networks. In *Algorithms and Models for the Web Graph*, page in press, 2013.

[28] R. I. Kondor and J. D. Lafferty. Diffusion kernels on graphs and other discrete input spaces. In *ICML*, pages 315–322, San Francisco, CA, USA, 2002. Morgan Kaufmann Publishers Inc.

[29] J. Kunegis and A. Lommatzsch. Learning spectral graph transformations for link prediction. In *ICML*, pages 561–568, 2009.

[30] H. Kwak, C. Lee, H. Park, and S. Moon. What is Twitter, a social network or a news media? In *WWW*, pages 591–600, 2010.

[31] J. Leskovec, D. Huttenlocher, and J. Kleinberg. Signed networks in social media. In *CHI*, pages 1361–1370, 2010.

[32] J. Leskovec, J. Kleinberg, and C. Faloutsos. Graph evolution: Densification and shrinking diameters. *ACM Trans. Knowl. Discov. Data*, 1:1–41, March 2007.

[33] J. Leskovec, K. J. Lang, A. Dasgupta, and M. W. Mahoney. Community structure in large networks: Natural cluster sizes and the absence of large well-defined clusters. *Internet Mathematics*, 6(1):29–123, September 2009.

[34] M. Liou. A novel method of evaluating transient response. *Proceedings of the IEEE*, 54(1):20–23, 1966.

[35] Z. Q. Luo and P. Tseng. On the convergence of the coordinate descent method for convex differentiable minimization. *J. Optim. Theory App.*, 72(1):7–35, 1992. 10.1007/BF00939948.

[36] F. McSherry. A uniform approach to accelerated PageRank computation. In *WWW*, pages 575–582, 2005.

[37] B. V. Minchev and W. M. Wright. A review of exponential integrators for first order semi-linear problems. Technical Report Numerics 2/2005, Norges Teknisk-Naturvitenskapelige Universitet, 2005.

[38] A. Mislove, M. Marcon, K. P. Gummadi, P. Druschel, and B. Bhattacharjee. Measurement and analysis of online social networks. In *SIGCOMM*, pages 29–42, 2007.

[39] C. Moler and C. Van Loan. Nineteen dubious ways to compute the exponential of a matrix. *SIAM Review*, 20(4):801–836, 1978.

[40] C. Moler and C. Van Loan. Nineteen dubious ways to compute the exponential of a matrix, twenty-five years later. *SIAM Review*, 45(1):3–49, 2003.

[41] M. Newman. Network datasets. http://www-personal.umich.edu/~mejn/netdata/, 2006.
[42] M. E. J. Newman. The structure of scientific collaboration networks. *PNAS*, 98(2):404–409, 2001.
[43] M. E. J. Newman. Modularity and community structure in networks. *PNAS*, 103(23):8577–8582, 2006.
[44] L. Orecchia and M. W. Mahoney. Implementing regularization implicitly via approximate eigenvector computation. In *ICML*, pages 121–128, 2011.
[45] L. Orecchia, S. Sachdeva, and N. K. Vishnoi. Approximating the exponential, the Lanczos method and an Otilde(m)-time spectral algorithm for balanced separator. In *STOC*, pages 1141–1160, 2012.
[46] L. Orecchia and Z. A. Zhu. Flow-based algorithms for local graph clustering. In *SODA*, pages 1267–1286, 2014.
[47] S. E. Schaeffer. Graph clustering. *Computer Science Review*, 1(1):27–64, 2007.
[48] D. L. Shepelyansky and O. V. Zhirov. Google matrix, dynamical attractors, and ulam networks. *Phys. Rev. E*, 81(3):036213, March 2010.
[49] X. Sui, T.-H. Lee, J. J. Whang, B. Savas, S. Jain, K. Pingali, and I. Dhillon. Parallel clustered low-rank approximation of graphs and its application to link prediction. In *Languages and Compilers for Parallel Computing*, volume 7760 of *Lecture Notes in Computer Science*, pages 76–95. Springer Berlin, 2013.
[50] C. (The Cooperative Association for Internet Data Analyais). Network datasets. http://www.caida.org/tools/measurement/skitter/router_topology/, 2005. Accessed in 2005.
[51] W. Y. Wang, K. Mazaitis, and W. W. Cohen. Programming with personalized pagerank: A locally groundable first-order probabilistic logic. In *CIKM*, pages 2129–2138, New York, NY, USA, 2013. ACM.
[52] J. J. Whang, D. F. Gleich, and I. S. Dhillon. Overlapping community detection using seed set expansion. In *CIKM*, pages 2099–2108, New York, NY, USA, 2013. ACM.
[53] J. Yang and J. Leskovec. Defining and evaluating network communities based on ground-truth. In *ICDM*, pages 745–754, 2012.
[54] Z. A. Zhu, S. Lattanzi, and V. Mirrokni. A local algorithm for finding well-connected clusters. In *ICML*, pages 396–404, 2013.

APPENDIX

A. CONVERGENCE THEORY

Here we state our main result bounding the work required by `hk-relax` to approximate the heat kernel with accuracy as described in (3).

THEOREM 1. *Let $\boldsymbol{P}$, t, $\psi_k(t)$, and $\boldsymbol{r}$ be as in Section 4. If steps of `hk-relax` are performed until all entries of the residual satisfy $r(i,j) < \frac{e^t \varepsilon d_i}{2N\psi_j(t)}$, then `hk-relax` produces an approximation $\boldsymbol{x}$ of $\boldsymbol{h} = \exp\{-t(\boldsymbol{I}-\boldsymbol{P})\}\,\boldsymbol{s}$ satisfying*

$$\|\boldsymbol{D}^{-1}\exp\{-t(\boldsymbol{I}-\boldsymbol{P})\}\,\boldsymbol{s} - \boldsymbol{D}^{-1}\boldsymbol{x}\|_\infty < \varepsilon,$$

and the amount of work required is bounded by

$$\textit{work}(\varepsilon) \le \frac{2N\psi_1(t)}{\varepsilon} \le \frac{2N(e^t-1)}{\varepsilon t}.$$

Producing $\mathbf{x}$ satisfying

$$\|\mathbf{D}^{-1}\exp\{-t(\mathbf{I}-\mathbf{P})\}\,\mathbf{s} - \mathbf{D}^{-1}\mathbf{x}\|_\infty < \varepsilon$$

is equivalent to producing $\mathbf{y}$ satisfying

$$\|\mathbf{D}^{-1}\exp\{t\mathbf{P})\}\,\mathbf{s} - \mathbf{D}^{-1}\mathbf{y}\|_\infty < e^t\varepsilon.$$

We will show that the error vector in the `hk-relax` steps, $T_N(t\mathbf{P})\mathbf{s}-\mathbf{y}$, satisfies $\|\mathbf{D}^{-1}T_N(t\mathbf{P})\mathbf{s} - \mathbf{D}^{-1}\mathbf{y}\|_\infty < e^t\varepsilon/2$.

The following lemma expresses the error vector, $T_N(t\mathbf{P})\mathbf{s}-\mathbf{y}$, as a weighted sum of the residual blocks $\mathbf{r}_k$ in the linear system (9), and shows that the polynomials $\psi_k(t)$ are the weights.

LEMMA 1. *Let $\psi_k(t)$ be defined as in Section 4. Then in the notation of Section 4.3, we can express the error vector of `hk-relax` in terms of the residual blocks $\boldsymbol{r}_k$ as follows*

$$T_N(t\boldsymbol{P})\boldsymbol{s} - \boldsymbol{y} = \sum_{k=0}^{N}\psi_k(t\boldsymbol{P})\boldsymbol{r}_k \tag{12}$$

PROOF. Consider (9). Recall that $\mathbf{v} = [\mathbf{v}_0; \mathbf{v}_1; \cdots; \mathbf{v}_N]$ and let $\mathbf{S}$ be the $(N+1)\times(N+1)$ matrix of 0s with first subdiagonal equal to $[\frac{1}{1}, \frac{1}{2}, \cdots, \frac{1}{N}]$. Then we can rewrite this linear system more conveniently as

$$(\mathbf{I} - \mathbf{S}\otimes(t\mathbf{P}))\mathbf{v} = \mathbf{e}_1 \otimes \mathbf{s}. \tag{13}$$

Let $\mathbf{v}_k$ be the true solution vectors that the $\hat{\mathbf{v}}_k$ are approximating in (13). We showed in Section 4.3 that the error $T_N(t\mathbf{P})\mathbf{s}-\mathbf{y}$ is in fact the sum of the errors $\mathbf{v}_k - \hat{\mathbf{v}}_k$. Now we will express $T_N(t\mathbf{P})\mathbf{s}-\mathbf{y}$ in terms of the residual partitions, i.e. $\mathbf{r}_k$.

At any given step we have $\mathbf{r} = \mathbf{e}_1\otimes\mathbf{s} - (\mathbf{I}-\mathbf{S}\otimes(t\mathbf{P}))\hat{\mathbf{v}}$, so pre-multiplying by $(\mathbf{I}-\mathbf{S}\otimes(t\mathbf{P}))^{-1}$ yields $(\mathbf{I}-\mathbf{S}\otimes(t\mathbf{P}))^{-1}\mathbf{r} = \mathbf{v}-\hat{\mathbf{v}}$, because $(\mathbf{I}-\mathbf{S}\otimes(t\mathbf{P}))\mathbf{v} = \mathbf{e}_1\otimes\mathbf{s}$ exactly, by definition of $\mathbf{v}$. Note that $(\mathbf{I}-\mathbf{S}\otimes(t\mathbf{P}))^{-1}\mathbf{r} = \mathbf{v}-\hat{\mathbf{v}}$ is the error vector for the linear system (13). From this, an explicit computation of the inverse (see [27] for details) yields

$$\begin{bmatrix}\mathbf{v}_0-\hat{\mathbf{v}}_0\\ \vdots \\ \mathbf{v}_N-\hat{\mathbf{v}}_N\end{bmatrix} = \left(\sum_{k=0}^{N}\mathbf{S}^k\otimes(t\mathbf{P})^k\right)\begin{bmatrix}\mathbf{r}_0\\ \vdots\\ \mathbf{r}_N\end{bmatrix}. \tag{14}$$

For our purposes, the full block vectors $\mathbf{v}, \hat{\mathbf{v}}, \mathbf{r}$ and their individual partitions are unimportant: we want only their sum, because $T_N(t\mathbf{P})\mathbf{s}-\mathbf{y} = \sum_{k=0}^{N}(\mathbf{v}_k-\hat{\mathbf{v}}_k)$, as previously discussed.

Next we use (14) to express $\sum_{k=0}^{N}(\mathbf{v}_k-\hat{\mathbf{v}}_k)$, and hence

$$T_N(t\mathbf{P})\mathbf{s}-\mathbf{y},$$

in terms of the residual blocks $\mathbf{r}_k$. We accomplish this by examining the coefficients of an arbitrary block $\mathbf{r}_k$ in (14), which in turn requires analyzing the powers of $(\mathbf{S}\otimes t\mathbf{P})$.

Fix a residual block $\mathbf{r}_{j-1}$ and consider the product with a single term $(\mathbf{S}^k\otimes(t\mathbf{P})^k)$. Since $\mathbf{r}_{j-1}$ is in block-row j of $\mathbf{r}$, it multiplies with only the block-column j of each term $(\mathbf{S}^k\otimes(t\mathbf{P})^k)$, so we want to know what the blocks in block-column j of $(\mathbf{S}^k\otimes(t\mathbf{P})^k)$ look like.

From [27] we know that

$$\mathbf{S}^m\mathbf{e}_j = \begin{cases}\frac{(j-1)!}{(j-1+m)!}\mathbf{e}_{j+m} & \text{if } 0\le m\le N+1-j\\ 0 & \text{otherwise.}\end{cases}$$

This means that block-column j of $(\mathbf{S}^m\otimes(t\mathbf{P})^m)$ contains only a single nonzero block, $\frac{(j-1)!}{(j-1+m)!}t^m\mathbf{P}^m$, for all $0\le m\le N+1-j$. Hence, summing the $n\times n$ blocks in block-column j of each power $(\mathbf{S}^m\otimes(t\mathbf{P})^m)$ yields

$$\sum_{m=0}^{N+1-j}\frac{(j-1)!t^m}{(j-1+m)!}\mathbf{P}^m \tag{15}$$

as the matrix coefficient of $\mathbf{r}_{j-1}$ in the right-hand side expression of $\sum_{k=0}^{N}(\mathbf{v}_k-\hat{\mathbf{v}}_k) = (\mathbf{e}^T\otimes\mathbf{I})(\sum_{m=0}^{N}\mathbf{S}^m\otimes(t\mathbf{P})^m)\mathbf{r}$.

Substituting $k = j-1$ on the right-hand side of (15) yields

$$\begin{aligned}\sum_{k=0}^{N}(\mathbf{v}_k-\hat{\mathbf{v}}_k) &= (\mathbf{e}^T\otimes\mathbf{I})\left(\sum_{m=0}^{N}\mathbf{S}^m\otimes(t\mathbf{P})^m\right)\mathbf{r}\\ &= \sum_{k=0}^{N}\left(\sum_{m=0}^{N-k}\frac{k!t^m}{(k+m)!}\mathbf{P}^m\right)\mathbf{r}_k\\ &= \sum_{k=0}^{N}\psi_k(t\mathbf{P})\mathbf{r}_k,\end{aligned}$$

as desired. □

Now that we've shown $T_N(t\mathbf{P})\mathbf{s} - \mathbf{y} = \sum_{k=0}^{N} \psi_k(t\mathbf{P})\mathbf{r}_k$ we can upperbound $\|\mathbf{D}^{-1}T_N(t\mathbf{P})\mathbf{s} - \mathbf{D}^{-1}\mathbf{y}\|_\infty$. To do this, we first show that

$$\|\mathbf{D}^{-1}\psi_k(t\mathbf{P})\mathbf{r}_k\|_\infty \leq \psi_k(t)\|\mathbf{D}^{-1}\mathbf{r}_k\|_\infty. \tag{16}$$

To prove this claim, recall that $\mathbf{P} = \mathbf{A}\mathbf{D}^{-1}$. Since $\psi_k(t)$ is a polynomial, we have $\mathbf{D}^{-1}\psi_k(t\mathbf{P}) = \mathbf{D}^{-1}\psi_k(t\mathbf{A}\mathbf{D}^{-1}) = \psi_k(t\mathbf{D}^{-1}\mathbf{A})\mathbf{D}^{-1}$, which equals $\psi_k(t\mathbf{P}^T)\mathbf{D}^{-1}$.

Taking the infinity norm and applying the triangle inequality to the definition of $\psi_k(t)$ gives us

$$\|\mathbf{D}^{-1}\psi_k(t\mathbf{P})\mathbf{r}_k\|_\infty \leq \sum_{m=0}^{N-k} \frac{t^m k!}{(m+k)!}\|(\mathbf{P}^T)^m(\mathbf{D}^{-1}\mathbf{r}_k)\|_\infty.$$

Then we have $\|(\mathbf{P}^T)^m(\mathbf{D}^{-1}\mathbf{r}_k)\|_\infty \leq \|(\mathbf{P}^T)^m\|_\infty\|(\mathbf{D}^{-1}\mathbf{r}_k)\|_\infty$ which equals $\|(\mathbf{D}^{-1}\mathbf{r}_k)\|_\infty$ because $\mathbf{P}^T$ is row-stochastic. Returning to our original objective, we then have

$$\begin{aligned}\|\mathbf{D}^{-1}\psi_k(t\mathbf{P})\mathbf{r}_k\|_\infty &\leq \sum_{m=0}^{N-k} \frac{t^m k!}{(m+k)!}\|(\mathbf{D}^{-1}\mathbf{r}_k)\|_\infty \\ &= \psi_k(t)\|\mathbf{D}^{-1}\mathbf{r}_k\|_\infty,\end{aligned}$$

proving the claim in (16).

This allows us to continue:

$$T_N(t\mathbf{P})\mathbf{s} - \mathbf{y} = \sum_{k=0}^{N} \psi_k(t\mathbf{P})\mathbf{r}_k \qquad \text{so}$$

$$\begin{aligned}\|\mathbf{D}^{-1}T_N(t\mathbf{P})\mathbf{s} - \mathbf{D}^{-1}\mathbf{y}\|_\infty &\leq \sum_{k=0}^{N} \|\mathbf{D}^{-1}\psi_k(t\mathbf{P})\mathbf{r}_k\|_\infty \\ &\leq \sum_{k=0}^{N} \psi_k(t)\|\mathbf{D}^{-1}\mathbf{r}_k\|_\infty.\end{aligned}$$

The stopping criterion for `hk-relax` requires that every entry of the residual satisfies $r(i,j) < \frac{e^t\varepsilon d_i}{2N\psi_j(t)}$, which is equivalent to satisfying $\frac{r(i,j)}{d_i} < \frac{e^t\varepsilon}{2N\psi_j(t)}$. This condition guarantees that $\|\mathbf{D}^{-1}\mathbf{r}_k\|_\infty < \frac{e^t\varepsilon}{2N\psi_k(t)}$. Thus we have

$$\begin{aligned}\|\mathbf{D}^{-1}T_N(t\mathbf{P})\mathbf{s} - \mathbf{D}^{-1}\mathbf{y}\|_\infty &\leq \sum_{k=0}^{N} \psi_k(t)\|\mathbf{D}^{-1}\mathbf{r}_k\|_\infty \\ &\leq \sum_{k=0}^{N} \left(\psi_k(t)\frac{e^t\varepsilon}{2N\psi_k(t)}\right)\end{aligned}$$

which is bounded above by $e^t\varepsilon/2$. Finally, we have that

$$\|\mathbf{D}^{-1}T_N(t\mathbf{P})\mathbf{s} - \mathbf{D}^{-1}\mathbf{y}\|_\infty < e^t\varepsilon/2$$

implies $\|\mathbf{D}^{-1}(\exp\{-t(\mathbf{I}-\mathbf{P})\}\,\mathbf{s} - \mathbf{x})\|_\infty < \varepsilon/2$, completing our proof of the first part of Theorem 1.

Bounding work.

It remains to bound the work required to perform steps of `hk-relax` until the stopping criterion is achieved.

Because the stopping criterion requires each residual entry to satisfy

$$\frac{r(i,j)}{d_i} < \frac{e^t\varepsilon}{2N\psi_j(t)}$$

we know that each step of `hk-relax` must operate on an entry $r(i,j)$ larger than this threshold. That is, each step we relax an entry of $\mathbf{r}$ satisfying $\frac{r(i,j)}{d_i} \geq \frac{e^t\varepsilon}{2N\psi_j(t)}$.

Consider the solution vector $\mathbf{y} \approx T_N(t\mathbf{P})\mathbf{s}$ and note that each entry of $\mathbf{y}$ is really a sum of values that we have deleted from $\mathbf{r}$. But $\mathbf{r}$ always contains only nonnegative values: the seed vector $\mathbf{s}$ is nonnegative, and each step involves setting entry $r(i,j) = 0$ and adding a scaled column of $\mathbf{P}$, which is nonnegative, to $\mathbf{r}$. Hence, $\|\mathbf{y}\|_1$ equals the sum of the $r(i,j)$ added to $\mathbf{y}$, $\sum_{l=1}^{T} r(i_l, j_l) = \|\mathbf{y}\|_1$.

Finally, since the entries of $\mathbf{y}$ are nondecreasing (because they only change if we add positive values $r(i,j)$ to them), we know that $\|\mathbf{y}\|_1 \leq \|T_N(t\mathbf{P})\mathbf{s}\|_1 = \psi_0(t) \leq e^t$. This implies, then, that

$$\sum_{l=1}^{T} r(i_l, j_l) \leq e^t.$$

Using the fact that the values of $r(i,j)$ that we relax must satisfy $\frac{r(i,j)}{d_i} \geq \frac{e^t\varepsilon}{2N\psi_j(t)}$ we have that

$$\sum_{l=1}^{T} \frac{d_{i_l}e^t\varepsilon}{2N\psi_{j_l}(t)} \leq e^t.$$

Simplifying yields

$$\sum_{l=1}^{T} \frac{d_{i_l}}{\psi_{j_l}(t)} \leq \frac{2N}{\varepsilon}.$$

By Lemma 1 we know $\psi_k(t) \leq \psi_1(t)$ for each $k \geq 1$, so we can lowerbound $\sum_{l=1}^{T} \frac{d_{i_l}}{\psi_1(t)} \leq \sum_{l=1}^{T} \frac{d_{i_l}}{\psi_{j_l}(t)} \leq \frac{2N}{\varepsilon}$, giving us

$$\sum_{l=1}^{T} d_{i_l} \leq \frac{2N\psi_1(t)}{\varepsilon}.$$

Finally, note that the dominating suboperation in each step of `hk-relax` consists of relaxing $r(i,j)$ and spreading this "heat kernel rank" to the neighbors of node i in block $\mathbf{r}_{j+1}$. Since node i has d_i neighbors, this step consists of d_i adds, and so the work performed by `hk-relax` is $\sum_{l=1}^{T} d_{i_l}$, which is exactly the quantity we bounded above.