Heat Kernel Based Community Detection

Joint with David F. Gleich, (Purdue), supported by NSF CAREER 1149756-CCF

Kyle Kloster
Purdue University
Local Community Detection

Given seed(s) $S$ in $G$, find a community that contains $S$. 

“Community”? 

seed
Local Community Detection

Given seed(s) $S$ in $G$, find a community that contains $S$.

“Community”?

high internal, low external connectivity
Low-conductance sets are communities

\[
\text{conductance}(T) = \frac{\text{# edges leaving } T}{\text{# edge endpoints in } T}
\]

= “chance a random step exits \( T \) ”
Low-conductance sets are communities

\[ \text{conductance}(T) = \frac{\text{# edges leaving } T}{\text{# edge endpoints in } T} \]

= “chance a random step exits \( T \)”

\[ \text{conductance}(\text{comm}) = \frac{39}{381} = 0.102 \]

How to find these?
Graph diffusions find low-conductance sets

A diffusion propagates “rank” from a seed across a graph.
Graph diffusions find low-conductance sets

A diffusion propagates “rank” from a seed across a graph.

Okay… how does this work?
Graph Diffusion

A diffusion models how a mass (green dye, money, popularity) spreads from a seed across a network.
A diffusion models how a mass (green dye, money, popularity) spreads from a seed across a network.
Graph Diffusion

A diffusion models how a mass (green dye, money, popularity) spreads from a seed across a network.

Once mass reaches a node, it propagates to the neighbors, with some decay.

“decay”: dye dilutes, money is taxed, popularity fades
Graph Diffusion

A diffusion models how a mass (green dye, money, popularity) spreads from a seed across a network.

Once mass reaches a node, it propagates to the neighbors, with some decay.

“decay”: dye dilutes, money is taxed, popularity fades
Graph Diffusion

A diffusion models how a mass (green dye, money, popularity) spreads from a seed across a network.

Once mass reaches a node, it propagates to the neighbors, with some decay.

“decay”: dye dilutes, money is taxed, popularity fades
Diffusion score

“diffusion score” of a node = weighted sum of the mass at that node during different stages.

\[ c_0 p_0 + c_1 p_1 + c_2 p_2 + c_3 p_3 + \ldots \]
**Diffusion score**

“diffusion score” of a node = weighted sum of the mass at that node during different stages.

\[ f = \sum_{k=0}^{\infty} c_k P^k s \]

- **Diffusion score vector** = \( f \)
- **\( P \)** = random-walk transition matrix
- **\( s \)** = normalized seed vector
- **\( c_k \)** = weight on stage \( k \)
**Heat Kernel vs. PageRank Diffusions**

**Heat Kernel** uses $t^k/k!$

Our work is new analysis for this diffusion.

**PageRank** uses $\alpha^k$ at stage k.

Standard, widely-used diffusion we use for comparison.
Heat Kernel vs. PageRank Behavior

HK emphasizes earlier stages of diffusion.

⇒ involve shorter walks from seed,
⇒ so HK looks at smaller sets than PR
### Heat Kernel vs. PageRank Theory

<table>
<thead>
<tr>
<th>good conductance</th>
<th>fast algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>PR: Local Cheeger Inequality: “PR finds set of near-optimal conductance”</td>
<td>“PPR-push” is $O(1/(\varepsilon(1-\alpha)))$ in theory, fast in practice [Andersen Chung Lang 06]</td>
</tr>
</tbody>
</table>
## Heat Kernel vs. PageRank Theory

<table>
<thead>
<tr>
<th></th>
<th>good conductance</th>
<th>fast algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>PR</strong></td>
<td>Local Cheeger Inequality: “PR finds set of near-optimal conductance”</td>
<td>“PPR-push” is $O(1/(\varepsilon(1-\alpha)))$ in theory, fast in practice</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[Andersen Chung Lang 06]</td>
</tr>
<tr>
<td><strong>HK</strong></td>
<td>Local Cheeger Inequality [Chung 07]</td>
<td></td>
</tr>
</tbody>
</table>
## Heat Kernel vs. PageRank Theory

<table>
<thead>
<tr>
<th></th>
<th>good conductance</th>
<th>fast algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>PR</strong></td>
<td>Local Cheeger Inequality: “PR finds set of near-optimal conductance”</td>
<td>“PPR-push” is $O(1/(\varepsilon(1-\alpha)))$ in theory, fast in practice [Andersen Chung Lang 06]</td>
</tr>
<tr>
<td><strong>HK</strong></td>
<td>Local Cheeger Inequality [Chung 07]</td>
<td><strong>Our work</strong></td>
</tr>
</tbody>
</table>
Our work on **Heat Kernel: theory**

**THEOREM** Our algorithm for a relative $\varepsilon$-accuracy in a degree-weighted norm has runtime $\leq O\left( e^t(\log(1/\varepsilon) + \log(t)) / \varepsilon \right)$

(which is constant, regardless of graph size)
Our work on **Heat Kernel**: theory

**THEOREM** Our algorithm for a relative \( \varepsilon \)-accuracy in a degree-weighted norm has

\[
\text{runtime} \leq O( e^t \log(1/\varepsilon) + \log(t) ) / \varepsilon
\]

(which is constant, regardless of graph size)

**COROLLARY** **HK** is local!

\((O(1) \text{ runtime } \rightarrow \text{ diffusion vector has } O(1) \text{ entries})\)
Our work on **Heat Kernel**: results

First efficient, deterministic HK algorithm. Deterministic is important to be able to compare the behaviors of HK and PR experimentally:

Our key findings

- **HK** more accurately describes ground-truth communities in real-world networks
- identifies smaller sets $\rightarrow$ better precision
- speed & conductance comparable with PR
Python demo

Twitter graph
41.6 M nodes
2.4 B edges

un-optimized Python code on a laptop

Available for download:

https://gist.github.com/dgleich/cf170a226aa848240cf4
Algorithm Outline

Computing HK

1. Pre-compute “push” thresholds
2. Do “push” on all entries above threshold
Algorithm Intuition

Computing HK given parameters $t, \varepsilon, \text{seed } s$

Starting from here…

How to end up here?
Algorithm Intuition

Begin with mass at seed(s) in a “residual” staging area, $r_0^\text{seed}$

The residuals $r_k$ hold mass that is unprocessed – it’s like error

\[
\frac{t^0}{0!} p_0 + \frac{t^1}{1!} p_1 + \frac{t^2}{2!} p_2 + \frac{t^3}{3!} p_3 + \ldots
\]
**Push Operation**

**push** – (1) remove entry in $r_k$,  
(2) put in $p$,

\[
\begin{align*}
  &\frac{t^0}{0!}p_0 + \frac{t^1}{1!}p_1 + \frac{t^2}{2!}p_2 + \frac{t^3}{3!}p_3 + \ldots \\
  &r_0 \quad r_1 \quad r_2 \quad r_3 \quad \ldots
\end{align*}
\]
**Push Operation**

**push** – (1) remove entry in \( r_k \),
(2) put in \( p \),
(3) then scale and spread to neighbors in next \( r \)

\[
\frac{t^0}{0!} p_0 + \frac{t^1}{1!} p_1 + \frac{t^2}{2!} p_2 + \frac{t^3}{3!} p_3 + \ldots
\]
**Push Operation**

**push** – (1) remove entry in \( r_k \),  
(2) put in \( p \),  
(3) then scale and spread to neighbors in next \( r \)  
(repeat)
Push Operation

push – (1) remove entry in \( r_k \),
(2) put in \( p \),
(3) then scale and spread to neighbors in next \( r \)
(repeat)
Push Operation

push – (1) remove entry in $r_k$,  
(2) put in $p$,  
(3) then scale and spread to neighbors in next $r$  
(repeat)
Thresholds

ERROR equals weighted sum of entries left in $r_k$

→ Set threshold so “leftovers” sum to $\leq \varepsilon$

$$\frac{t^0}{0!} p_0 + \frac{t^1}{1!} p_1 + \frac{t^2}{2!} p_2 + \frac{t^3}{3!} p_3 + \ldots$$
Thresholds

ERROR equals weighted sum of entries left in $r_k$

→ Set threshold so “leftovers” sum to $< \varepsilon$

Threshold for stage $r_k$ is

\[
\frac{t^0}{0!} p_0 + \frac{t^1}{1!} p_1 + \frac{t^2}{2!} p_2 + \frac{t^3}{3!} p_3 + \ldots
\]
Algorithm Outline

Computing HK

1. Pre-compute “push” thresholds
2. Do “push” on all entries above threshold
Communities in Real-world Networks

Given a seed in an unidentified real-world community, how well can HK and PR describe that community? Measure quality using $F_1$-measure.

| Graph   | $|V|$   | $|E|$   | $F_1$-measure |
|---------|--------|--------|---------------|
| amazon  | 330 K  | 930 K  | precision     |
| dblp    | 320 K  | 1 M    |               |
| youtube | 1.1 M  | 3 M    | recall        |
| lj      | 4 M    | 35 M   |               |
| orkut   | 3.1 M  | 120 M  |               |
| friendster | 66 M | 1.8 B  |               |

Datasets from SNAP collection [Leskovec]
<table>
<thead>
<tr>
<th>data</th>
<th>$F_1$ (HK)</th>
<th>$F_1$ (PR)</th>
<th>precision (HK)</th>
<th>precision (PR)</th>
<th>set size (HK)</th>
<th>set size (PR)</th>
<th>comm size</th>
</tr>
</thead>
<tbody>
<tr>
<td>amazon</td>
<td>0.325</td>
<td>0.140</td>
<td>0.244</td>
<td>0.107</td>
<td>193</td>
<td>15293</td>
<td>495</td>
</tr>
<tr>
<td>dblp</td>
<td>0.257</td>
<td>0.115</td>
<td>0.208</td>
<td>0.081</td>
<td>44</td>
<td>16026</td>
<td>1429</td>
</tr>
<tr>
<td>youtube</td>
<td>0.177</td>
<td>0.136</td>
<td>0.135</td>
<td>0.098</td>
<td>1010</td>
<td>6079</td>
<td>1615</td>
</tr>
<tr>
<td>lj</td>
<td>0.131</td>
<td>0.107</td>
<td>0.102</td>
<td>0.086</td>
<td>283</td>
<td>738</td>
<td>662</td>
</tr>
<tr>
<td>orkut</td>
<td>0.055</td>
<td>0.044</td>
<td>0.036</td>
<td>0.031</td>
<td>537</td>
<td>1989</td>
<td>4526</td>
</tr>
<tr>
<td>friendster</td>
<td>0.078</td>
<td>0.090</td>
<td>0.066</td>
<td>0.075</td>
<td>229</td>
<td>333</td>
<td>724</td>
</tr>
<tr>
<td>data</td>
<td>$F_1$ HK</td>
<td>$F_1$ PR</td>
<td>precision HK</td>
<td>precision PR</td>
<td>set size HK</td>
<td>set size PR</td>
<td>comm size</td>
</tr>
<tr>
<td>----------</td>
<td>---------</td>
<td>---------</td>
<td>--------------</td>
<td>--------------</td>
<td>-------------</td>
<td>-------------</td>
<td>-----------</td>
</tr>
<tr>
<td>amazon</td>
<td>0.325</td>
<td>0.140</td>
<td>0.244</td>
<td>0.107</td>
<td>193</td>
<td>15293</td>
<td>495</td>
</tr>
<tr>
<td>dblp</td>
<td>0.257</td>
<td>0.115</td>
<td>0.208</td>
<td>0.081</td>
<td>44</td>
<td>16026</td>
<td>1429</td>
</tr>
<tr>
<td>youtube</td>
<td>0.177</td>
<td>0.136</td>
<td>0.135</td>
<td>0.098</td>
<td>1010</td>
<td>6079</td>
<td>1615</td>
</tr>
<tr>
<td>lj</td>
<td>0.131</td>
<td>0.107</td>
<td>0.102</td>
<td>0.086</td>
<td>283</td>
<td>738</td>
<td>662</td>
</tr>
<tr>
<td>orkut</td>
<td>0.055</td>
<td>0.044</td>
<td>0.036</td>
<td>0.031</td>
<td>537</td>
<td>1989</td>
<td>4526</td>
</tr>
<tr>
<td>friendster</td>
<td>0.078</td>
<td>0.090</td>
<td>0.066</td>
<td>0.075</td>
<td>229</td>
<td>333</td>
<td>724</td>
</tr>
</tbody>
</table>

PR achieves high recall by “guessing” a huge set

HK identifies a tighter cluster, so attains better precision
**Runtime & Conductance**

HK is comparable in runtime and conductance.

As graphs scale, the diffusions' performance becomes even more similar.
Code, references, future work

Code available at

http://www.cs.purdue.edu/homes/dgleich/codes/hkgrow

Ongoing work

- generalizing to other diffusions
- simultaneously compute multiple diffusions

Questions or suggestions? Email Kyle Kloster at kkloste-at-purdue-dot-edu