Using the Heat Kernel of a Graph for Local Algorithms

Complex Networks Minisymposium
SIAM Applied Linear Algebra
October 28, 2015

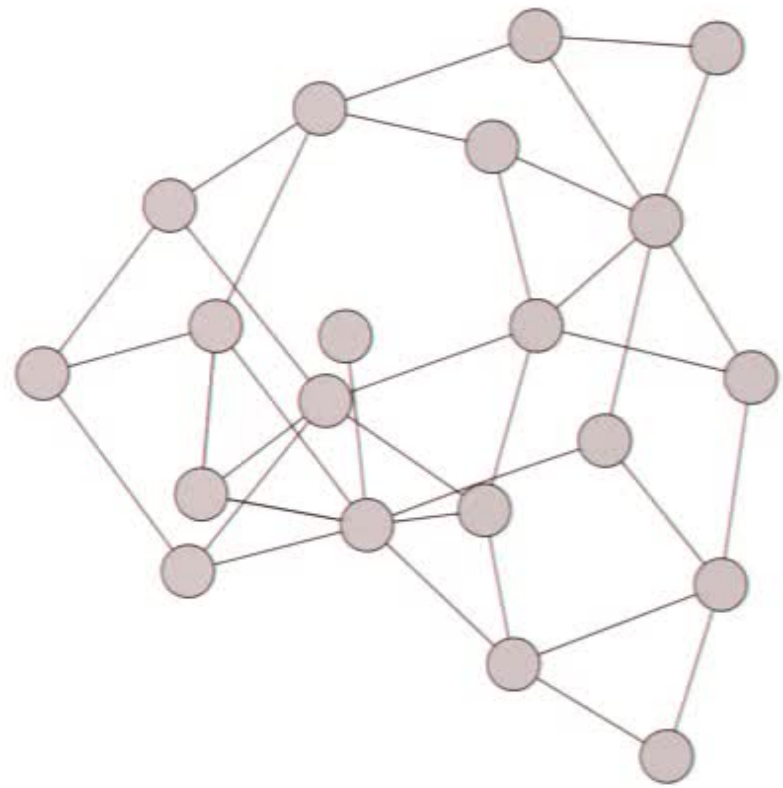
Olivia Simpson
UC San Diego
osimpson@ucsd.edu

Joint work with Fan Chung

Understanding Data

- Model relationship between data points with graphs
  - Social networks
  - Communication networks
  - Biological networks
  - Network topologies

- We will consider undirected, unweighted graphs with no multi-edges or self-loops.



Understanding Data

- These graphs tend to be:
  - **Large**, often too big to store in main memory
  - **Dynamic**: need scalable ways to update computations
  - **Distributed**: need fast ways for servers to communicate

- Computation becomes intractable

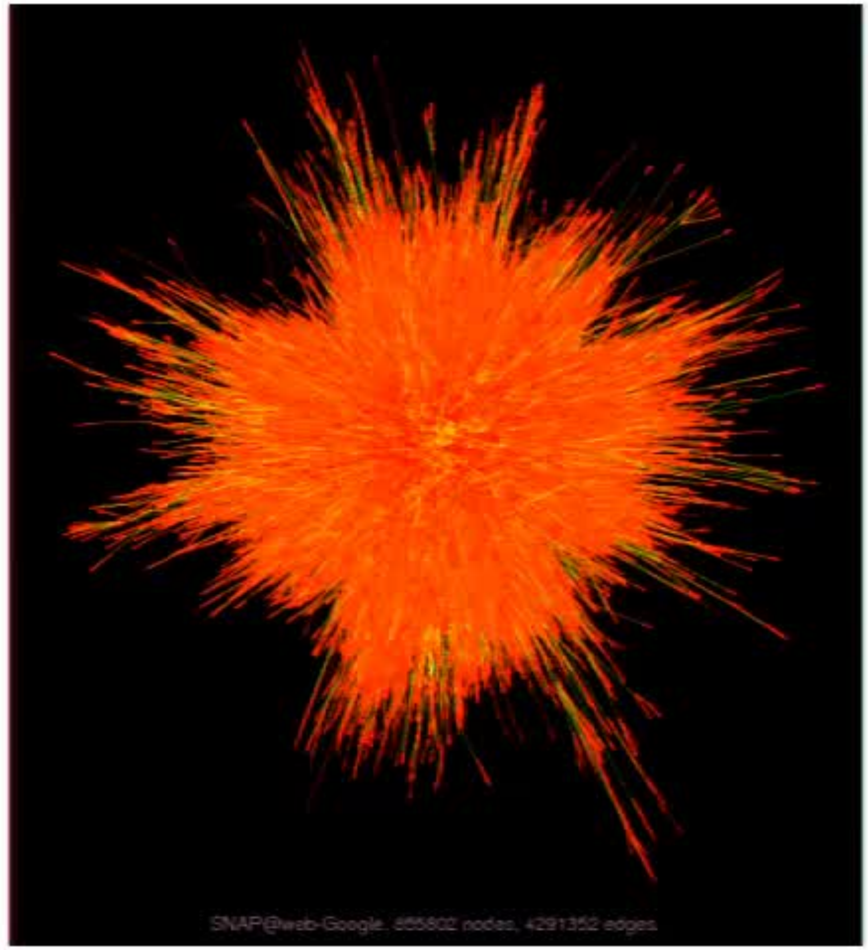


Image due to Tim Davis:
http://www.cise.ufl.edu/research/sparse/matrices/SNAP/web-Google.html

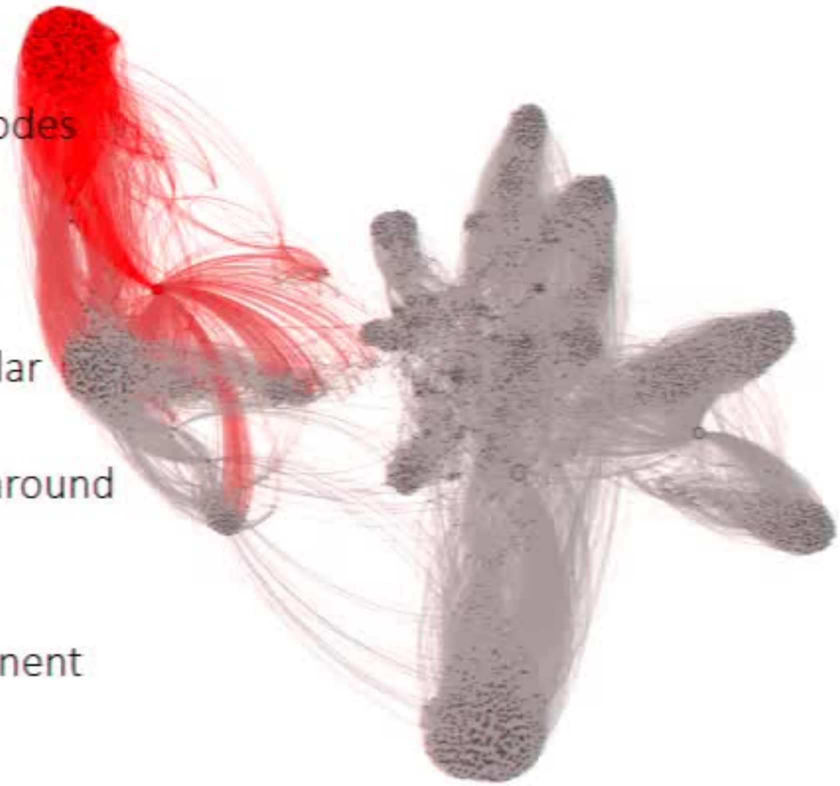
Local algorithms

- Do not rely on the state of the full graph
- Designing local algorithms:
  - Want space and time complexity in terms of size of output
  - Guarantees with good initialization (as opposed to global algorithms which have guarantees regardless of initialization but with incurred cost)

- Use random walks, analysis using spectral graph theory
- Local, lightweight queries: RandomNode(), RandomNeighbor(), etc.

Local cluster detection

- (Global) clustering
  - Increase granularity by identifying similar nodes
  - Make operations on large networks more tractable
- Local clustering
  - Only interested in a particular group of similar nodes
    - Social networks: identify a community around a particular member
    - Protein networks: isolate a group of interacting proteins to analyze a component of a biological system



Local cluster detection

- Goal: identify a good cluster near a specified node
  - We will use the *Cheeger ratio* (sometimes called conductance) as our metric:

    Let S be a subset of nodes in the network, then the Cheeger ratio is

$$\Phi(S) = \frac{number\ of\ edges\ leaving\ S}{\text{volume(S)}}.$$

$$volume(S) = \sum_{v \in S} d_v$$

- Typical formulation of a local clustering algorithm:

"If S is a subset of Cheeger ratio $\Phi(S) \leq \phi$, then with high probability there are many nodes in S that can be used as 'seeds' for finding a set T with Cheeger ratio $\Phi(T) = O(f(\phi))$"

  - Would like f to be small
  - Would like for running time to be proportional to the size of T

Finding good cuts with stochastic processes

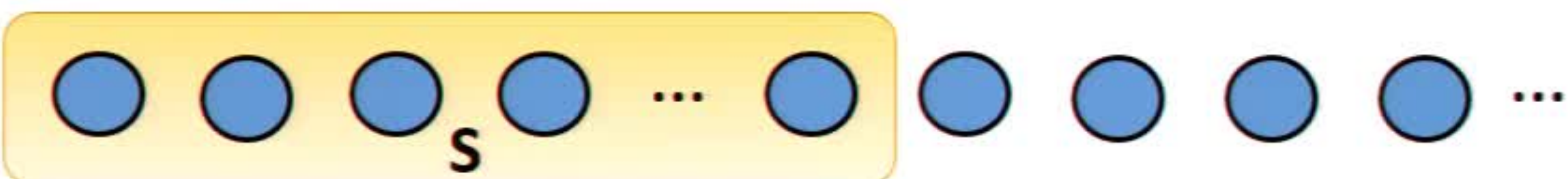
"Single sweep" algorithms:

let $p: \mathrm{V} \rightarrow \mathbb{R}$ be a probabilistic function over the nodes of the graph

1. Order the nodes according to

$$\frac{p(v_1)}{d_{v_1}} \geq \frac{p(v_2)}{d_{v_2}} \geq \cdots \geq \frac{p(v_n)}{d_{v_n}}$$

2. For $i = 1 \rightarrow n$:
    1. Let V_i be the set of the first i nodes
    2. Check if this set meets the desired volume, Cheeger ratio



Finding good cuts with stochastic processes

- Spectral methods
  - [Alon, Milman '85]: *the Cheeger inequalities* relating the Cheeger ratio to eigenvalues
  - Typically use eigenfunctions to find good cuts
- "Single sweep" algorithms:
  - [Lovász, Simonovitz '90, '93], [Spielman, Teng '04]: lazy random walks
  - [Chung '07]: Laplacian eigenvectors
  - [Andersen et al., '06]: PageRank (or reset) random walks
  - [Andersen, Peres '09]: evolving (cluster) sets with Markov chains
  - [Gharan, Trevisan '12]: lazy random walks + evolving sets
  - Random walk or probability diffusion based, **local**

PageRank as a probability distribution

$$\text{pr}_{\alpha,f} = \alpha \sum_{k=0}^{\infty} (1-\alpha)^k f P^k$$

- Parameters are α, a jumping probability, and f, a starting distribution
- Stationary distribution of a "reset random walk":
  - At every step:
    - with probability $1 - \alpha$ move from u to a neighbor v with probability $\frac{1}{d_u}$
    - with probability α jump to a node drawn from the starting distribution
- Common starting distributions: uniform, personalized
- Lower α means more diffusion of probability
- High α means more probability near the start node

Heat kernel pagerank as a probability distribution

$$\rho_{t,f} = e^{-t}\, fI + e^{-t}t\, fP + e^{-t}\frac{t^2}{2!}\, fP^2 + \cdots + e^{-t}\frac{t^k}{k!}\, fP^k + \cdots$$

- $\sum_{k=0}^{\infty} e^{-t}\frac{t^k}{k!} = 1$ ...
- Let X be the random variable that takes on value fP^k with probability $p_k \leftarrow e^{-t}\frac{t^k}{k!}$. Then $\mathbb{E}[X] = \rho_{t,f}$.
- If f is a probability distribution over the nodes, then fP^k is the distribution over the nodes after k random walk steps.
  - We'll take f to be χ_u, the indicator vector for seed node u, and use $\rho_{t,u} \coloneqq \rho_{t,\chi_u}$
- *Redefine the random process:* X takes on the probability distribution after k random walk steps starting from node u with probability $p_k \leftarrow e^{-t}\frac{t^k}{k!}$.

Computing heat kernel pagerank with random walks

- "heat kernel random walk":
  - take k random walk steps with probability $k \leftarrow Poiss(t)$
  - At every step:
    - move from u to a neighbor v with probability ${}^{1}/_{d_u}$
- For local clusters, can keep walks short
- ϵ-approximate heat kernel pagerank values:

  1. $(1-\epsilon)\rho_{t,s}(v) - \epsilon \leq \hat{\rho}_{t,s}(v) \leq (1+\epsilon)\rho_{t,s}(v)$, *and*
  2. *for each node* v *with* $\hat{\rho}_{t,s}(v) = 0$, *it must be that* $\rho_{t,s}(v) \leq \epsilon$

  "Relaxed" notion of approximation which captures nodes with high heat kernel pagerank value ($> \epsilon$) and ignores the rest

Heat kernel pagerank as a probability distribution

$$\rho_{t,f} = e^{-t} fI + e^{-t}t\, fP + e^{-t}\frac{t^2}{2!} fP^2 + \cdots + e^{-t}\frac{t^k}{k!} fP^k + \cdots$$

- $\sum_{k=0}^{\infty} e^{-t}\frac{t^k}{k!} = 1$ ...
- Let X be the random variable that takes on value fP^k with probability $p_k \leftarrow e^{-t}\frac{t^k}{k!}$. Then $\mathbb{E}[X] = \rho_{t,f}$.
- If f is a probability distribution over the nodes, then fP^k is the distribution over the nodes after k random walk steps.
  - We'll take f to be χ_u, the indicator vector for seed node u, and use $\rho_{t,u} \coloneqq \rho_{t,\chi_u}$
- *Redefine the random process:* X takes on the probability distribution after k random walk steps starting from node u with probability $p_k \leftarrow e^{-t}\frac{t^k}{k!}$.

Computing heat kernel pagerank with random walks

Approximate the estimated distribution $\mathbb{E}[X] = \rho_{t,u}$ with random walks:

```
ApproxHKPR(G, t, u, ϵ):
ρ̂ ← 0-vector of size n
K ← max_walk_length
for r rounds do:
      k ← min(Poiss(t),K)
      perform a k-step random walk from u
      v ← last node visited in the random walk
      ρ̂[v] += 1

return ρ̂/r
```

sublinear

Running time: $O(rK)$

- $r = 16\epsilon^{-3}\log n$
- $K = \frac{\log(1/\epsilon)}{\log\log(1/\epsilon)}$

Random walks can be performed with local, lightweight *RandomNeighbor()* queries

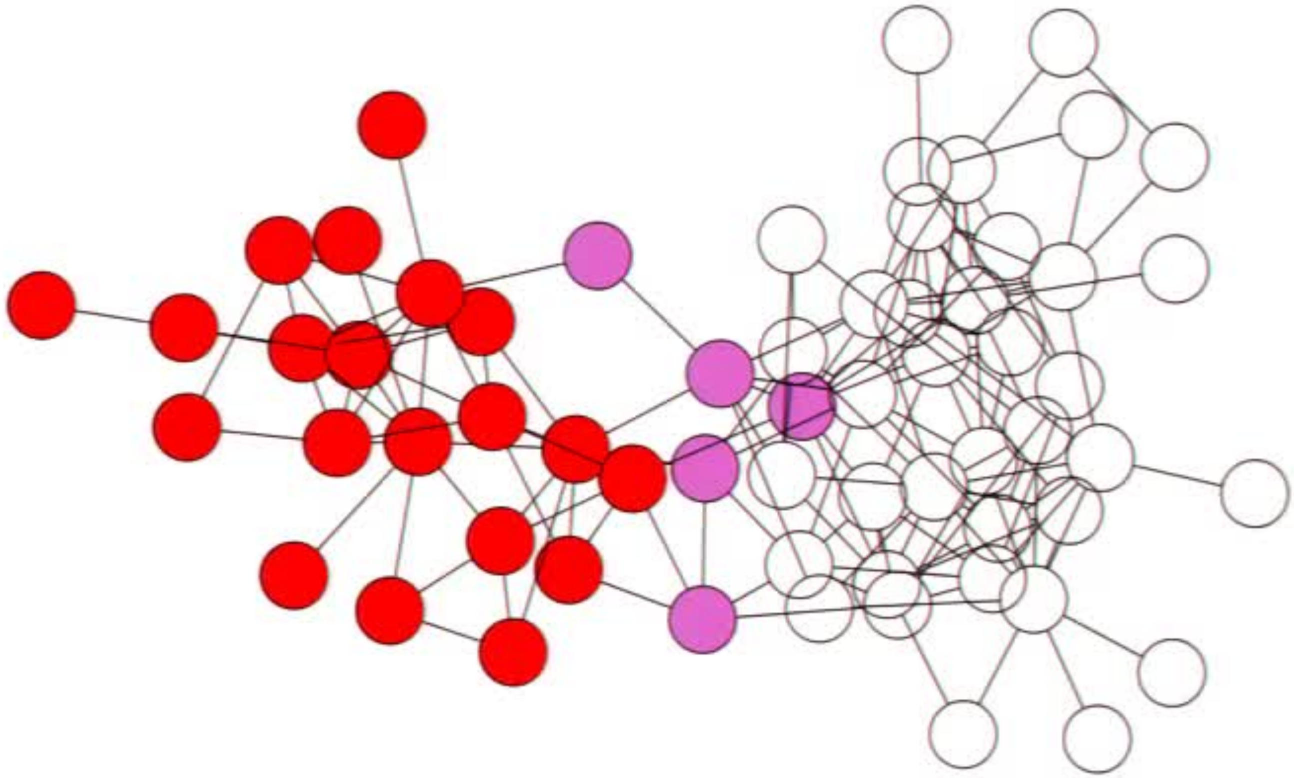
Finding local clusters with stochastic processes

- "Single sweep" algorithms:

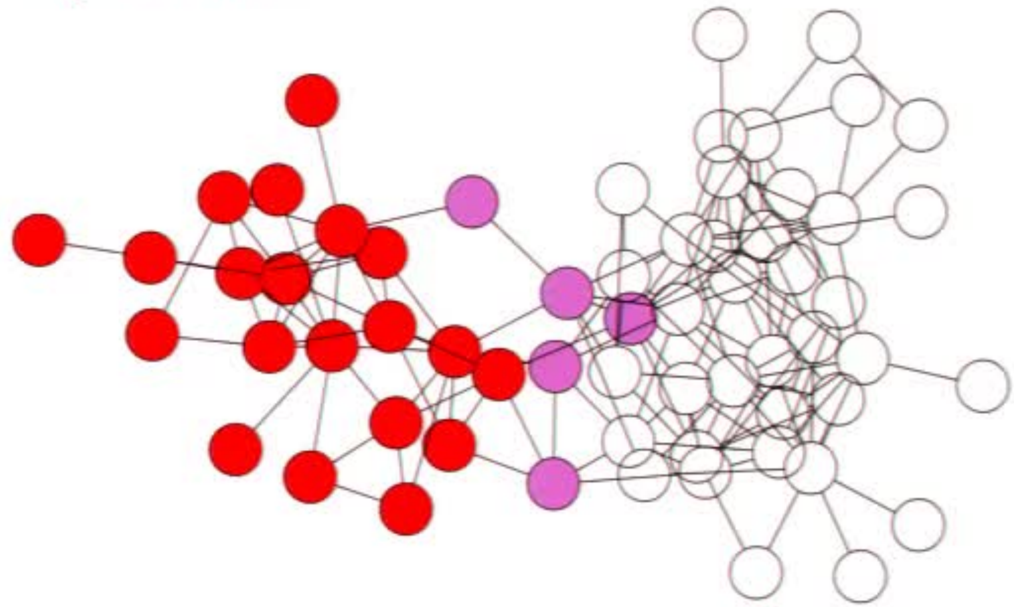
Algorithm	Conductance of output set	Work/volume ratio
[Spielman, Teng '04]	$O(\phi^{1/2} \log^{3/2} n)$	$O(\phi^{-2} \operatorname{polylog} n)$
[Andersen et al., '06]	$O(\phi^{1/2} \log^{1/2} n)$	$O(\phi^{-1} \operatorname{polylog} n)$
[Andersen, Peres '09]	$O(\phi^{1/2} \log^{1/2} n)$	$O(\phi^{-1/2} \operatorname{polylog} n)$
[Gharan, Trevisan '12]	$O(\epsilon^{-1/2} \phi^{1/2})$	$O(\varsigma^{\epsilon} \phi^{-1/2} \operatorname{polylog} n)$
[Chung, S. '14]	$O(\phi^{1/2})$	$O(\varsigma^{-1} \epsilon^{-3} \log n \log(\epsilon^{-1}) \log\log(\epsilon^{-1}))$

work/volume ratio: ratio between the computational complexity of the algorithm on a given run and the volume of the output set

Solving local linear systems



Local Laplacian linear systems



Local Laplacian linear systems

$$x_S = \mathcal{L}_S^{-1} b_1$$

$\approx$

$$\sum_{j=1}^{N} \rho_{S,jT/N,b_2} \frac{T}{N} D_S^{-1/2}$$

approximate the sum by sampling $r \leftarrow \gamma^{-2} \log(s\gamma^{-1})$
Dirichlet heat kernel pagerank vectors

[Chung, S. WAW‘13], [Chung, S. IM‘15]

Computing Dirichlet heat kernel pagerank with random walks

$$\rho_{S,t,f} = \sum_{k=0}^{\infty} e^{-t} \frac{t^k}{k!} f P_S^k$$

- "Dirichlet heat kernel random walk":
  - take k random walk steps with probability $k \leftarrow Poiss(t)$
  - At every step:
    - move from u to a neighbor v with probability $^1/_{d_u}$
    - If v is outside of S, abort the walk and ignore any contribution from it
  - Since we only want to consider probability diffusion *within* S, cannot allow any random walks which have left S to return any probability to it
- t parameter is more sensitive
  - Allow walks of length up to $^t/_\varepsilon$ (in practice $2t$ is fine)

Summary: local algorithms and applications of heat kernel pagerank

- Vector values can be computed by sampling random walks
- Sublinear number of random walks sufficient
- Can bound length of random walks by $t/_{\varepsilon}$
  - In practice, $2t$ is more than enough
  - In some cases, constant $K = \frac{\log(1/\varepsilon)}{\log\log(1/\varepsilon)}$ independent of t, n is enough
- A single vector can be used to compute a local cluster
- Vectors can be sampled to compute a local Laplacian linear solution