Hierarchical Directed Spectral Graph Partitioning

MS&E 337 - Information Networks

Stanford University
Fall 2005

David Gleich

February 1, 2006

Abstract

In this report, we examine the generalization of the Laplacian of a graph due to Fan Chung. We show that Fan Chung's generalization reduces to examining one particular symmetrization of the adjacency matrix for a directed graph. From this result, the directed Cheeger bounds trivially follow. Additionally, we implement and examine the benefits of directed hierarchical spectral clustering empirically on a dataset from Wikipedia. Finally, we examine a set of competing heuristic methods on the same dataset.

1 Clustering for Directed Graphs

Clustering problems often arise when looking at graphs and networks. At the highest level, the problem of clustering is to partition a set of objects such that each partition contains similar objects. The choice of how to define the similarity between objects is the key component to a clustering algorithm.

If we restrict ourselves to clustering graphs, then a natural way to define a clustering is a graph partition. That is, each of the objects is a vertex in our graph and we want to partition the vertices of the graph to optimize a function on the edges and vertices. For example, one common function is the *normalized cut* objective function. Given a set of vertices S,

$$ncut(S) = \operatorname{vol} \partial S \left(\frac{1}{\operatorname{vol} S} + \frac{1}{\operatorname{vol} \bar{S}} \right),$$

where

$$\bar{S} = V - S, \quad \operatorname{vol} \partial S = \sum_{(u,v) \in E | u \in S, v \in \bar{S}} w_{i,j}, \text{ and } \operatorname{vol} S = \sum_{u \in S} \sum_{(u,v) | u \to v} w_{u,v}.$$

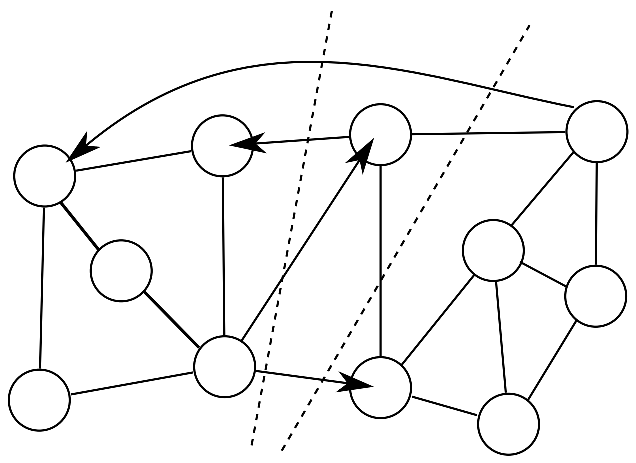


Figure 1: An example where an "optimal" directed cut differs from an "optimal" symmetrized cut. Edges without arrows are bi-directional, and this graph has two natural groupings – the leftmost 5 nodes and the rightmost 6 nodes. With directionality intact, the obvious cut is the set of directed edges (the left cut). Without directionality, the two cuts are indistinguishable when looking at the ratio of cut edges to vertices.

For those readers to whom these definitions are not yet entirely clear, we'll revisit them soon.

This definition, however, only applies to undirected graphs. Given a directed graph, how do we generalize this property? In [1], Fan Chung provides an answer. We'll see her proposal in section 2. Before we delve into the details of her method, however, let's examine the question of why?

That is, why look at directed graph partitioning? Given any directed graph, we can *symmetrize* the graph to an undirected graph by dropping the direction of each edge. We can then use any of the numerous partitioning methods for undirected graphs. The problem with this approach is that it loses information. Consider the graph in figure 1, which shows an example where the loss of information is critical to making a correct choice. In this sense, directed clustering is critical to preserving the information in the original graph.

2 The Directed Laplacian

We begin this section by summarizing previously known results about the Laplacian of a graph. Next, we introduce the normalized Laplacian and Fan Chung's directed Laplacian. We conclude the section with a proof of the directed Cheeger bounds for the directed Laplacian by reducing to the undirected case.

Let $G = (V, E, w)$ be a weighted, undirected graph (we'll handle the directed case soon!). We restrict ourselves to positive weights $w > 0$. For an unweighted graph, let

$w_{i,j} = 1$ if (i,j) is an edge. Let W be the weighted adjacency matrix for G,

$$W_{i,j} = \begin{cases} w_{i,j} & \text{if } (i,j) \in E \\ 0 & \text{otherwise.} \end{cases}$$

Definition. *The* Laplacian *of G is the matrix*

$$L = \operatorname{Diag}(We) - W,$$

where $\operatorname{Diag}(We)$ *is a diagonal matrix with the row-sums of W along the diagonal.*

The Laplacian has a particularly nice set of properties. First, the matrix L is symmetric because W is symmetric for an undirected graph. For an unweighted graph,

$$L_{i,j} = \begin{cases} d_i & \text{if } i = j \\ -1 & \text{if } i \text{ is adjacent to } j \\ 0 & \text{otherwise,} \end{cases}$$

where d_i is the degree of vertex i. For a weighted graph, we have

$$L_{i,j} = \begin{cases} \sum_j W_{i,j} & \text{if } i = j \\ -W_{i,j} & \text{if } i \text{ is adjacent to } j \\ 0 & \text{otherwise,} \end{cases}$$

From these facts, we have that L is a singular matrix with null-space (at least) e, the vector of all ones. From now on, we will only consider the weighted case, but mention useful reductions to the unweighted case.

Doing a little bit of algebra (which is readily available in most references on the Laplacian), we have

$$x^T L x = \sum_{(i,j)\in E} w_{i,j}(x_i - x_j)^2.$$

From this fact, it follows that L is positive semi-definite. We'll now state more properties of the Laplacian.[1]

Claim. *[2] Let $0 = \lambda_0 \leq \lambda_1 \leq \ldots \leq \lambda_{n-1}$ be the eigenvalues of L. Then L is connected if and only if $\lambda_1 > 0$. Additionally, the number of connected components of G is equal to the dimension of the null-space of L (the multiplicity of the zero eigenvalue).*

While this claim appears to provide an algorithm for computing the number of connected components of a graph, it should not be used in this manner. Computing the connected components is a trivial operation using a breadth first search of the graph. In contrast, computing the eigenvalues is an inherently iterative procedure (due to the insolvability of the quintic and higher order polynomials) and plagued with numerical roundoff issues.

[1] For proofs of these facts, see the references.

Claim. *[3] Let G_1 be a subgraph of G with the same nodes and a subset of edges. Let L denote the Laplacian matrix for G and L_1 denote the Laplacian of G_1, then*

$$\lambda_1(L_1) \leq \lambda_1(L).$$

This definition is one motivation for calling λ_1 the *algebraic connectivity* of G. As we decrease the number of edges in a graph, λ_1 decreases; that is to say, as we decrease the connectivity of G, λ_1 decreases.

Perhaps the most important theorem with the Laplacian are the Cheeger inequalities. However, first we need a few more definitions. The first definition is a formalization of a graph partition and the induced graph cut. Following that, we define the volume associated with a graph and a cut. Finally, we define the conductance of a cut and the conductance of a graph.

Definition. *$S \subset V$ is called a* cut *of a graph because it induces a partition of V into S and $\bar{S}$.*

Definition. *The* volume *of a vertex v is defined*

$$\operatorname{vol} v = \sum_u W_{v,u}.$$

Likewise, the volume of a set is

$$\operatorname{vol} S = \sum_{v \in S} \operatorname{vol} v.$$

Finally, the volume of a cut is

$$\operatorname{vol} \partial S = \sum_{u \in S, v \in \bar{S}} W_{u,v}.$$

One important property of the $\operatorname{vol} \partial S$ is that

$$\operatorname{vol} \partial S = \operatorname{vol} \partial \bar{S}.$$

That is, the volume of the cut is symmetric for each side of the partition. This fact follows straightforwardly from the symmetry of W.

$$\operatorname{vol} \partial S \sum_{u \in S, v \in \bar{S}} W_{u,v} = \sum_{u \in S, v \in \bar{S}} W_{v,u} = \operatorname{vol} \partial \bar{S}.$$

Definition. *The* conductance *of a cut S is*

$$\phi_G(S) = \frac{\operatorname{vol} \partial S}{\min(\operatorname{vol} S, \operatorname{vol} \bar{S})}.$$

The conductance *of a graph G is*

$$\phi_G = \min_{S \subset V} \rho_G(S).$$

Instead of the Laplacian, we often examine the normalized Laplacian of a graph.

Definition. *The* normalized Laplacian *of a graph is the matrix*

$$\mathcal{L} = D^{-1/2} L D^{-1/2} = I - D^{-1/2} W D^{-1/2},$$

where $D = \mathrm{Diag}(We)$.

Finally, we can state the a key theorem.[2]

Theorem 2.1. *[2] The second eigenvalue of the normalized Laplacian of the graph,* λ_1 *is related to the conductance* ϕ_G *by*

$$\frac{\phi_G^2}{2} \le \lambda_1 \le 2\phi_G.$$

This theorem relates the conductance of the graph to the second eigenvalue. This statement is further motivation for calling λ_1 the algebraic connectivity of G.

2.1 Directed Generalization

In [1], Fan Chung generalized many of the results from the previous section to a directed graph. We'll first examine some properties of a random walk. Then we'll define a circulation which allows us to define a symmetrization of the graph. Finally, we'll show that Chung's definitions of *volume* directly correspond to the definitions from the previous section when applied to the adjacency matrix $\Pi P + P^T \Pi$ (the matrices used will be defined later). Thus, the Cheeger bounds for the directed Laplacian follow.

Given a weighted directed graph $G = (V, E, w)$, a random walk on G is a Markov process with transition matrix

$$P = D^{-1} W,$$

where $D = \mathrm{Diag}(We)$ as always. If G is strongly connected, by the Perron-Frobenius theorem, we know that P has at least one left eigenvector which is strictly positive with eigenvalue 1 (because $\rho(P) \le 1$ and $Pe = e$). P has a unique left eigenvector with eigenvalue 1 if G is aperiodic. Henceforth, we'll assume that G is strongly connected and aperiodic and discuss what happens if this is not the case in section 2.3.

Let π be the unique left eigenvector such that

$$\pi P = \pi.$$

The row-vector π corresponds to the stationary distribution of the random walk. Looking at the previous equation we have that

$$\pi(u) = \sum_{v, v \to u} \pi(v) P(v, u),$$

that is, the probability of finding the random walk at u is the sum of all the incoming probabilities from vertices v that have a directed edges to u.

[2]Note to Amin, I believe this is the correct form of this theorem. If it isn't, one of my subsequent results will not follow.

On an undirected graph, we have that

$$\pi(u) = \frac{\operatorname{vol} u}{\operatorname{vol} V},$$

because

$$\sum_{v, v\to u} \frac{\operatorname{vol} v}{\operatorname{vol} V} \frac{W_{v,u}}{\operatorname{vol} v} = \sum_{v, v\to u} \frac{\operatorname{vol} v}{\operatorname{vol} V} \frac{W_{u,v}}{\operatorname{vol} v} = \frac{\operatorname{vol} u}{\operatorname{vol} V}.$$

We now define a circulation on a directed graph G.

Definition. *[1] A function $F : E \to \mathbb{R}^+ \cup \{0\}$ that assigns each directed edge to a non-negative value is called a circulation if*

$$\sum_{u, u\to v} F(u, v) = \sum_{w, v\to w} F(v, w),$$

for each vertex v.

One interpretation of a circulation is a flow in the graph. The flow at each vertex must be conserved, hence, the flow in is equal to the flow out.

Now, we'll demonstrate one circulation on a graph. Let

$$F_\pi(u, v) = \pi(u)P(u, v).$$

The conservation property the circulation follows directly from the stationarity property of π.

$$\sum_{u, u\to v} F_\pi(u, v) = \sum_{u, u\to v} \pi(u)P(u, v) = \pi(v) \cdot 1 = \sum_{w, v\to w} \pi(v)P(v, w).$$

Using the circulation F_π, we define the directed volume. These specific definitions come from [4] though Fan Chung calls them flows on a directed graph.

Definition. *The volume of a vertex v in a directed graph is*

$$\operatorname{vol} v = \sum_{u, u\to v} F(u, v).$$

The definition of the volume of a set generalizes in the same way. The volume crossing a cut is

$$\operatorname{vol} \partial S = \sum_{u\in S, v\in \bar{S}} F(u, v).$$

One critical property of this definition is that

$$\operatorname{vol} \partial S = \operatorname{vol} \partial \bar{S}.$$

Intuitively, this follows because of the conservation of circulation property. Because the circulation out of each vertex equals the circulation into each vertex, the net circulation out of a subset of vertices ($\operatorname{vol}\partial S$) must equal the net circulation into the subset ($\operatorname{vol}\partial \bar{S}$). Formally, but not intuitively, we have

$$\begin{aligned}\operatorname{vol}\partial S &= \sum_{u\in S, v\in \bar{S}} F(u,v)\\ &= \sum_{u\in S, v\in \bar{S}} F(u,v) + \sum_{u\in S, v\in S} F(u,v) - \sum_{u\in S, v\in S} F(u,v)\\ &= \operatorname{vol}(S) - \sum_{u\in S, v\in S} F(u,v).\end{aligned}$$

Now, we appeal to the definition of a circulation on $\operatorname{vol} S$, namely,

$$\operatorname{vol} S = \sum_{v\in S}\sum_{w, v\to w} F(v,w) = \sum_{v\in V, u\in S} F(v,u).$$

To conclude,

$$\begin{aligned}\operatorname{vol}\partial S &= \sum_{v\in V, u\in S} F(v,u) - \sum_{u\in S, v\in S} F(u,v)\\ &= \sum_{v\in \bar{S}, u\in S} F(v,u) + \sum_{u\in S, v\in S} F(u,v) - \sum_{u\in S, v\in S} F(u,v)\\ &= \operatorname{vol}\partial \bar{S}.\end{aligned}$$

At this point, we have a symmetric function over a cut in the graph. Hence, we can reuse our previous definition of ϕ_G and directly apply it to a directed graph. It remains to prove that the same Cheeger inequalities hold for this directed graph.

To accomplish this final step, we examine the matrix

$$\tilde{W} = \frac{\Pi P + P^T \Pi}{2},$$

where $\Pi = \operatorname{Diag}(\pi)$. In contrast to W, this matrix is symmetric. Let $\tilde{G}$ be the associated undirected graph corresponding to W. In the following lemma, we'll show $\phi_G = \phi_{\tilde{G}}$.

Lemma. *Let G be a directed graph and let $\tilde{G}$ be the weighted symmetrization of G such that $\tilde{G}$ has adjacency matrix $\tilde{W}$, then*

$$\operatorname{vol}_G S = \operatorname{vol}_{\tilde{G}} S,$$

and

$$\operatorname{vol}_G \partial S = \operatorname{vol}_{\tilde{G}} \partial S,$$

where vol_G is the directed volume as previously defined and $\operatorname{vol}_{\tilde{G}}$ is the undirected volume.

Proof. The first equality is trivial. Let e be the vector of all ones, and e_S be the vector with a 1 corresponding to a vertex in S and 0 otherwise, then

$$\begin{aligned}
\operatorname{vol}_{\tilde{G}} S &= e_S \tilde{W} e \\
&= \frac{e_S \Pi P e + e_S P^T \Pi e}{2} \\
&= \frac{e_S \Pi e + e_S P^T \pi}{2} \\
&= \frac{e_S \pi + e_S \pi}{2} \\
&= \sum_{u \in S} \pi(u).
\end{aligned}$$

Recall that

$$\operatorname{vol}_G S = \sum_{u \in S} \sum_{v, v \to u} F(v, u).$$

In this case, $F(u, v) = \pi(v) P(v, u)$, and we have that

$$\sum_{v, v \to u} F(v, u) = \pi(u),$$

because of the stationarity of random walk. Thus, the first inequality holds.

The second equality is also simple,

$$\begin{aligned}
\operatorname{vol}_{\tilde{G}} \partial S &= \sum_{u \in S, v \in \bar{S}} \frac{\pi(u) P(u, v) + P(v, u) \pi(v)}{2} \\
&= \frac{\operatorname{vol}_G \partial S}{2} + \frac{\operatorname{vol}_G \partial \bar{S}}{2}.
\end{aligned}$$

We are done because $\operatorname{vol}_G \partial S = \operatorname{vol}_G \partial \bar{S}$. □

Theorem 2.2. *Let G be a directed graph and let $\tilde{G}$ be the weighted symmetrization of G such that $\tilde{G}$ has adjacency matrix $\tilde{W}$. Then $\phi_G = \phi_{\tilde{G}}$.*

Proof. This result follows directly from the previous lemma, because we equated all the quantities in the definition. □

Corollary. *The Cheeger inequalities,*

$$\frac{\phi_G^2}{2} \leq \lambda_1 \leq 2\phi_G,$$

hold for the directed graph G when we define the directed Laplacian,

$$\mathcal{L} = \mathcal{L}(\tilde{\mathcal{G}}) = I - \frac{1}{2}\left(\Pi^{1/2} P \Pi^{-1/2} + \Pi^{-1/2} P^T \Pi^{1/2}\right),$$

and λ_1 is the first non-trivial eigenvalue of $\mathcal{L}$.

Proof. This follows from Theorem 2.1 and the previous theorem. □

The proof presented here differs from Fan Chung's proof in that we do not directly prove the Cheeger inequalities for the directed case. Rather, we show the equivalence between symmetrizing the graph using the stationary distribution and the undirected weighted analysis. This suggests a few ideas that are discussed in the conclusion.

2.2 Directed Laplacian on Undirected Graphs

If we happened to "forget" that a graph is undirected and use the directed Laplacian algorithm on G instead of the undirected version, nothing would change. The definition of the directed Laplacian is thus a generalization of the undirected Laplacian.

This fact follows from the stationary distribution of an undirected random walk. In the previous section, we showed that, for an undirected graph,

$$\pi(u) = \frac{\operatorname{vol} u}{\operatorname{vol} V}.$$

Let $D = \operatorname{Diag}(We)$, then $D_{ii} = \operatorname{vol} i$ and $\Pi^{1/2} = \frac{1}{\sqrt{\operatorname{vol} V}} D^{1/2}$. From these, we have that

$$\begin{aligned}
\mathcal{L} &= I - \frac{1}{2}\left(\Pi^{1/2} D^{-1} W \Pi^{-1/2} + \Pi^{-1/2} W^T D^{-1} \Pi^{1/2}\right) \\
&= I - \frac{1}{2}\left(\frac{1}{\sqrt{\operatorname{vol} V}} D^{1/2} D^{-1} W \sqrt{\operatorname{vol} V} D^{-1/2} + \sqrt{\operatorname{vol} V} D^{-1/2} W^T D^{-1} D^{1/2} \frac{1}{\sqrt{\operatorname{vol} V}}\right) \\
&= I - D^{-1/2} W D^{1/2}.
\end{aligned}$$

Thus, we get back the original normalized Laplacian.

For the original Laplacian, observe that

$$\operatorname{vol} V \cdot \Pi P = W.$$

Hence,

$$\begin{aligned}
\operatorname{vol} V \cdot L &= \operatorname{vol} V \cdot \Pi - \frac{1}{2}\left(\operatorname{vol} V \cdot \Pi P + \operatorname{vol} V \cdot P^T \Pi\right) \\
&= \operatorname{Diag}(We) - \frac{1}{2}(W + W^T),
\end{aligned}$$

and L is a rescaled version of the undirected Laplacian.

2.3 Not Strongly Connected or Aperiodic

At the beginning of the analysis for the directed Laplacian, we assumed that G was strongly connected and aperiodic. This allowed us to assume that G has a unique stationary distribution π.

If G is not aperiodic but is strongly connected, the matrix P will not have a unique stationary distribution. One remedy is to introduce the *lazy random walk*. Given the transition matrix for a random walk P on a strongly connected graph, the lazy random walk has a transition matrix

$$P_{\text{lazy}} = \frac{I + P}{2}.$$

Clearly the lazy random walk is aperiodic (it has self-loops), thus it has a unique stationary distribution. In [5], they suggest this modification.

If the graph G is not strongly connected, all is not lost! Instead, we can define the PageRank transition matrix which is a modification of the transition matrix $D^{-1}W$ to ensure aperiodicity and strong connection. We consider, instead,

$$P_{\text{PR}} = \alpha P + \frac{(1-\alpha)}{n} ee^T.$$

In Zhou et al [4], they use this modification to enable directed graph regularization on non-strongly connected graphs with $\alpha = 0.99$.

Each of these modifications changes the underlying graph G and we must be careful when using them to ensure that the results remain sensible on the original graph G.

3 Directed Spectral Clustering

The spectral graph partitioning algorithm is originally due to Fiedler. We interpret this algorithm as a clustering algorithm in that graph partitioning often corresponds to clustering. For an undirected graph, the algorithm is simple. We need to define a few more concepts before we can state the algorithm. Using our definition of the directed Laplacian from the previous section, we can immediately generalize this algorithm to directed graphs.

First, the *normalized cut* of a partition S of a graph G is

$$ncut(S) = \operatorname{vol}\partial S \left(\frac{1}{\operatorname{vol} S} + \frac{1}{\operatorname{vol} \bar{S}} \right).$$

The normalized cut of a graph is the minimum over all subsets S. Second, the expansion of a partition S of a graph G is

$$\rho(S) = \frac{\operatorname{vol}\partial S}{\min(|S|, |\bar{S}|)}.$$

Given the definition of volume from the previous section, these definitions generalize to a directed graph as well.

Finally, we observe that a permutation of the vertices of G induces $n-1$ partitions of the graph G. Further, we can compute the value of $ncut, \rho$, and ϕ for each of the $n-1$ partitions in time $O(|E|)$.

We formally state the recursive spectral clustering algorithm as figure 2. We provide an "algorithm run visualization" in figure 3. The idea with the algorithm is to use v_1 corresponding to λ_1 as an ordering of the vertices of the graph G that reveals a good cut. In the proof of the Cheeger inequalities, we use the fact that a sweep through all possible partitions induced by the ordering given in v_1 yields a good cut.

The origin of the normalized cut metric is [6]. Shi and Malik show that relaxing the integer program for normalized cuts reduces to a generalized eigenvalue problem,

$$\begin{array}{ll} \min_x ncut(x) = & \min_y \frac{y^T(\operatorname{Diag}(We)-W)y}{y^T \operatorname{Diag}(We) y}, \\ & \text{s.t. } y^T \operatorname{Diag}(We)^{-1} e = 0, y \in \{1, -b\} \end{array}$$

for a particular (but irrelevant here) defined constant b. Relaxing y to be real-valued gives that y the eigenvector with minimum eigenvalue of the generalized eigenvalue problem

$$\begin{array}{l} (\operatorname{Diag}(We) - W)y = \lambda \operatorname{Diag}(We) y, \\ \text{s.t. } y^T \operatorname{Diag}(We)^{-1} e = 0. \end{array}$$

```
Data: graph $G = (V, E, w)$, minimum partition size $p$
Result: a recursive partitioning of $G$
if $|V| < p$ then
    stop and return.
end
if $G$ has more than one connected component then
    divide $G$ into its components and recurse on each partition.
end
Compute $v_1$ corresponding to $\lambda_1$ of $L$ or $\mathcal{L}$ for $G$.
Sort $v_1$ into permutation $P$.
Sweep through all cuts induced by the ordering $P$ and compute one of
$ncut(P(1:i))$, $\rho_G(P(1:i))$ , or $\phi_G(P(1:i))$ and let $I$ correspond to the index
achieving the minimum value.
Recurse on $G(P(1:I))$ and $G(P(I+1:|V|))$ (the subgraphs induced by the
subset of vertices).
```

Figure 2: The (directed) version of the recursive spectral graph partitioning algorithm. If G is directed, then we must interpret G implicitly as one of the standard modifications for possibly aperiodic, not strongly connected graphs.

The second criteria merely states that that y is the first non-trivial eigenvector. Transforming the generalized eigenvalue problem into a standard eigenvalue problem by substituting $\tilde{y} = \text{Diag}(We)^{1/2}y$ and left-multiplying by $\text{Diag}(We)^{-1/2}$ shows that

$$\mathcal{L}\tilde{y} = \tilde{y}.$$

Hence

$$y = D^{-1/2}v_1,$$

where v_1 is the first non-trivial eigenvector of $\mathcal{L}$.

In [4], Zhou et al show that using v_1 corresponding to the normalized directed Laplacian is a real-valued relaxation of computing the optimal normalized cut as an integer program. With our analysis in this paper, this result is trivial and does not require proof. Instead, it follows directly from the definition of normalized cut and the symmetrization of the directed problem.

4 Other Directed Clustering

We can generalize our spectral clustering framework for any permutation vector P that may reveal a good cut in the graph. That is, we relax our requirement that v_1 is the second smallest eigenvector of one of the Laplacian matrices. Sprinkled throughout the

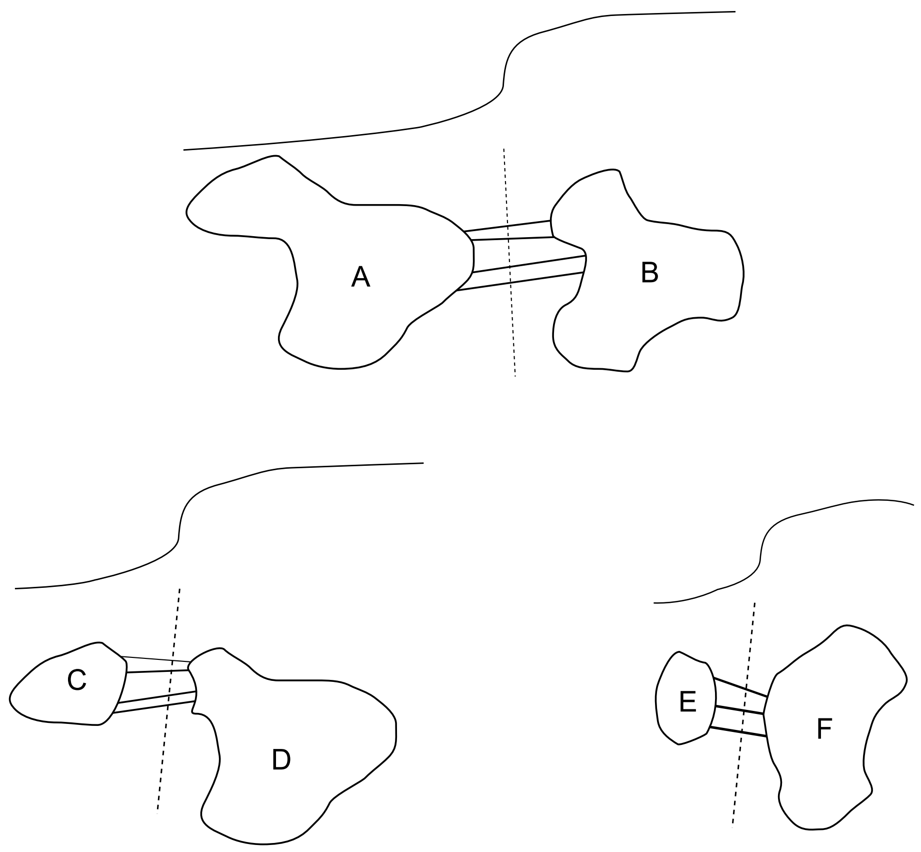


Figure 3: An illustration of hierarchical spectral clustering. We start with an initial graph G and compute the second smallest eigenvector of L of $\mathcal{L}$. If we look at the graph "along" this eigenvector, we see that the jump in the eigenvector corresponds to a good partition of the graph. In this case, we separate A from B. Next, we recurse on A and B and divide them into C, D and E, F, respectively. We continue this process until a suitable stopping criteria (e.g. less than p nodes in a partition) terminates the procedure. The eigenvector is shown above the each graph.

literature are ideas that may yield a vector with the property that sorting the graph by that vector reveals a good cut.

Our requirements for the algorithm and vector are simple:

- it must yield one or more vectors v that yield a sorting of the graph,
- it must work on directed graphs, and
- it should be computationally similar to computing v_1.

4.1 Higher non-trivial eigenvectors

The first obvious relaxation is to consider additional eigenvectors of the Laplacian. The motivation for this approach is direct in that each higher eigenvector is the next best solution to the normalized cut problem, subject to orthogonality with all previous solutions. Thus, we use algorithms that also examine higher eigenvectors and, potentially, partition using these instead of v_1.

The work required for this method is nearly equivalent to computing v_1. Empirically, we find that we often need many dimensions for the Arnoldi subspaces used in the ARPACK algorithm. These additional dimensions correspond to additional eigenvectors. Thus, if we have a cluster of eigenvalues near 0, we must find all the corresponding eigenvectors before they will converge. In fact, the benefits of splitting with these eigenvalues can be empirically motivated.

Consider the graph in figure 4. In that graph, the higher eigenvectors correspond to slightly worse, but similar cuts. In fact, on a recursive application, the spectral algorithm cuts these small clusters off one at a time.[3]

4.2 Cosine Similarity

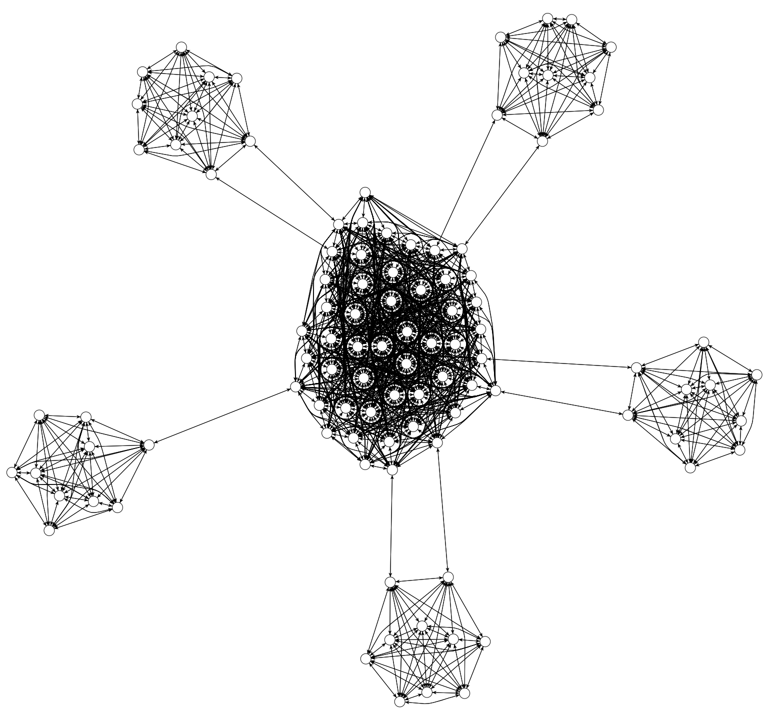
Next, we come to the idea of cosine similarity. In a high-dimensional space, the *cosine* of the angle between two vectors a_i, a_j is

$$\cos(a_i, a_j) = \frac{a_i^T a_j}{||a_i|| \cdot ||a_j||}.$$

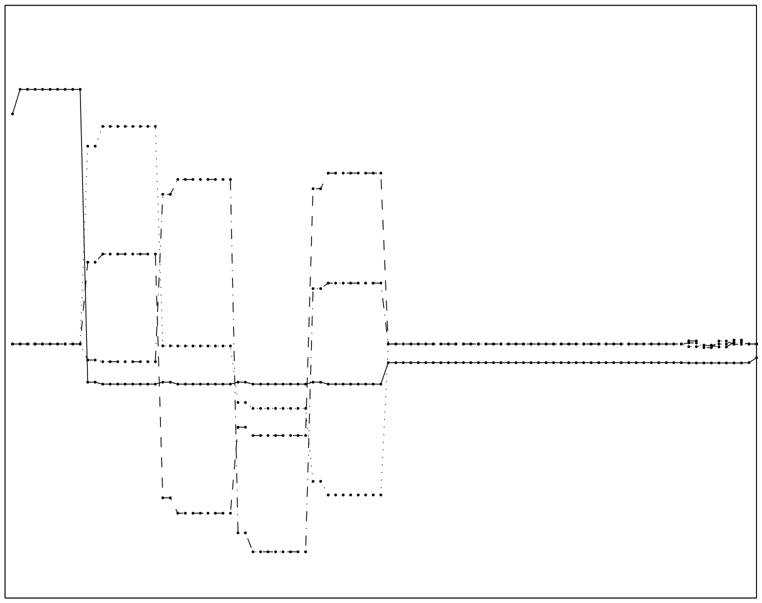
If we consider a particular node in the graph, i, we view that node as a row-vector of the adjacency matrix, a_i^T. We then compute the cosine similarity between that node and the rest of the graph. Trivially, $\cos(a_i, a_i) = 1$. The dot-product $a_i^T a_j$ counts the number out-links shared between nodes i and j. If these nodes share many outlinks, then they may be assigned to the same cluster. Hence, cosine similarity may correspond to some indication of clustering in the graph.

McSherry alluded to formalization of a similar algorithm for clustering in [7]. His algorithm naturally extends to bounding cosine similarity from below instead of bounding Euclidean distance above. However, instead of depending on the cosine value to directly correlate with clusters in the graph, we directly minimize one of the objective functions instead.

[3]This motivates using multiple eigenvectors to divide the graph, which is often used in practice [4].



(a) The graph



(b) Eigenvectors

Figure 4: An example of where higher eigenvectors yield additional partitioning information. The 5 small clusters are the first 50 vertices of the graph (10 vertices each) and are composed of a clique. The remainder of the graph is a 50 vertex clique. The first 5 eigenvectors shown here all have a similar form and separate the small clusters.

Using the vector of cosine similarities to a particular node is significantly faster than computing the second smallest eigenvector. Further, this method has no problems with directed graphs.

4.3 PageRank and Personalized PageRank

We've examined PageRank as a stationary distribution for a non-strongly connected, periodic graph. However, here we view it differently. PageRank roughly corresponds to the popularity of a given page. Hopefully, pages that are somehow similar may have similar PageRank values. Effectively, this idea is wishful thinking. Nevertheless, it is a trivial extension and leads to the idea of personalized PageRank.

In PageRank, we use the transition matrix

$$P_{\text{PR}} = \alpha P + \frac{(1-\alpha)}{n} ee^T.$$

This modification corresponds to adding a clique to our graph between all vertices with a low transition probability, $\frac{1-\alpha}{n}$, to each page in the graph. In contrast, with Personal PageRank, we add transitions from each vertex in our graph to one particular vertex i. Thus,

$$P_{\text{PPR}_i} = \alpha P + (1-\alpha) e e_i^T.$$

For Personal PageRank, we interpret the random walk as a process that begins at node i and proceeds according to the transition matrix P with probability α and resets with probability $1-\alpha$. Personalized PageRank, then, ranks the out-neighborhood of node i.

Personalized PageRank seems nearly ideal for this application, because it also defines a stationary distribution over the Personal PageRank modified graph. Thus, when we use Personalized PageRank as a sorting vector, it seems natural to use the Personalized PageRank symmetrized adjacency matrix to evaluate the cut metrics for choosing the split.

4.4 Right Hand Eigenvectors

In [8] Stewart states that the subdominant right hand eigenvectors are an indication of state clustering. That is, given a Markov chain transition matrix P, the dominant left eigenvector is the stationary distribution. The dominant right eigenvector is the vector e. Stewart shows, through straightforward algebra, that the coordinates of the subdominant right hand eigenvectors indicate how far the state is from the stationary distribution.

Naturally, these vectors fit nicely into our framework. Again, we can compute eigenvectors of the transition matrix P for our graphs at least as easily as computing the smallest eigenvectors for the Laplacian matrix. Thus, the computational time is equivalent. Again, there is no problem with directed graphs. For this method, we have to assume that P is strongly connected and aperiodic, however, as with the Laplacian, this is not a problem.

4.5 Random Permutations

Finally, in [2], we see that any random vector has a similar property to v_1 for the Laplacian matrix. Thus, guessing a random vector may yield a good ordering of the states.[4]

5 Implementation

We implemented all of the directed spectral clustering algorithm in a Matlab program. The usage for the program follows.

```
SPECTRAL Use recursive spectral graph partitioning to cluster a graph.

[ci p] = spectral(A, opts)

ci gives the cluster index of all results
p is a ordering permuation of the adjacency matrix A

The algorithm works slightly different than a standard second eigenvetor
splitting technique.  If possible for the desired problem type, the
algorithm will compute opts.nv vectors and use each vector as a linear
ordering over the vertices of the graph.  The algorithm chooses the
cut that maximizes the smallest partition size (i.e. best bisection).

opts.verbose: extra output information [{0} | 1]
opts.maxlevel: maximum recursion level [integer | {500}]
opts.minsize: minimum cluster size [integer | {5}]
opts.maxfull: largest size to use for full eigensolve [integer | {150}]
opts.split: a string specifying the split type
  [{'ncut'} | 'expansion' | 'conductance']
opts.directed: directed spectral clustering [{0} | 1]
opts.nv: the number of eigenvectors to check [integer | {1}]

options for directed spectral clustering
opts.directed_distribution: stationary distribution for graph
  [{'pagerank'} | 'lazywalk' | 'exact']
opts.pagerank_options: options structure for PageRank (see pagerank.m)
    pagerank_options.v is ignored
opts.laziness: laziness parameter is not using PageRank [float | {0.5}]
opts.lazywalk_options: options structure for lazywalk (see lazywalk.m)

opts.problem: which problem to solve for the sorting vector
 ['minbis' | {'ncut'} | 'cos' | 'prank' | 'pprank' | 'random' | 'reig']
```

The implementation in Matlab uses ARPACK for eigenvalue computations. We either use the stationary distribution for the exact graph G, or one of the two modifications – the lazy walk or PageRank vector. Second, we allow the users to choose

[4]This fact is in the notes, but I really don't see how it can be correct. From my experiments, this algorithm tends to bisect the graph in virtually every case because the cut values are monotone in the size of the smallest partition.

Abbv.	**|V|**	**|E|**	**Description**
wb-cs	9914	36854	All pages from 2001 Webbase crawl on `cs.stanford.edu`.
wp-100	18933	908412	Strongly connected component of Wikipedia articles with at least 100 inlinks.
neuro	202	2540	Neuron map from C. Elegans.

Table 1: A summary of statistics on our datasets.

which "problem" they use to generate the sorting vector v (or vectors if $nv > 1$ and the method produces multiple vectors). There is nothing particularly slick about the implementation.

The current implementation scales to roughly 100,000 vertex graphs. Around this point, the inefficiencies that result from Matlab begin to dominate. In theory, nothing prohibits a distributed data implementation of these ideas.

6 Evaluation

[Note: I intended to have a more thorough evaluation. However, due to time constraints, it has been abbreviated.]

We empirically evaluate the implementation of the previous section on a few graphs. The graphs are summarized in table 1.

We begin with wb-cs. This graph demonstrates a few interesting properties of the algorithms. First, the graph is directed and is not strongly connected. Thus, we use the PageRank modification of the graph with $\alpha = 0.99$. If we try using only one eigenvector of the directed Laplacian, the algorithms fail and we only divide off very small portions of the graph. Using multiple eigenvectors $nv = 25$ helps, although, the process still tends to stagnate. See figure 5 for a summary of the results from each run.

Next, we examine the wiki dataset. This dataset comes from a Wikipedia dump in November. We processed the dataset by removing all pages without at least 100 inlinks and computing the largest strongly connected component of what remained. This procedure yields a nice core subset of the wikipedia pages.

In contrast to the previous dataset, the partitioning algorithm worked well on the Wikipedia dataset with few modifications. At the end of the algorithm, we concatenated all the results into a permutation of the adjacency matrix. The results are presented in figure 6 and table 2. In figure 7, we show details on two of the apparent clusters in the dataset, the "time" cluster, which has links to many nodes, and the "baseball" cluster, which is tight with few links elsewhere.

Finally, we partition the neuro dataset. Like wiki, there were no problems with the algorithms on this dataset. This directed dataset corresponds to the neural connections in C. Elegans. For these nodes, a cluster is a set of highly interconnected neurons. It is particularly important to obey directionality in the clustering as the direction indicates the flow of information. The permutation shown in figure 8 should help biologists identify functional units among the neurons.

No progress with $nv = 1$

```
>> [ci p] = spectral(A, struct('directed_distribution', 'pagerank', ...
      'directed', 1, 'pagerank_options', struct('c', 0.99), 'nv', 1));
  level=0; split=[1, 9913]; cv=0.01 str=lambda_1=-9.72e-017; lambda_2=7.38e-003
  level=1; split=[1, 9912]; cv=0.01 str=lambda_1=1.88e-015; lambda_2=7.38e-003
  level=2; split=[9909, 3]; cv=0.01 str=lambda_1=-2.19e-015; lambda_2=7.38e-003
  level=3; split=[9908, 1]; cv=0.01 str=lambda_1=-3.69e-015; lambda_2=7.38e-003
  level=4; split=[5, 9903]; cv=0.01 str=lambda_1=3.16e-015; lambda_2=7.38e-003
```

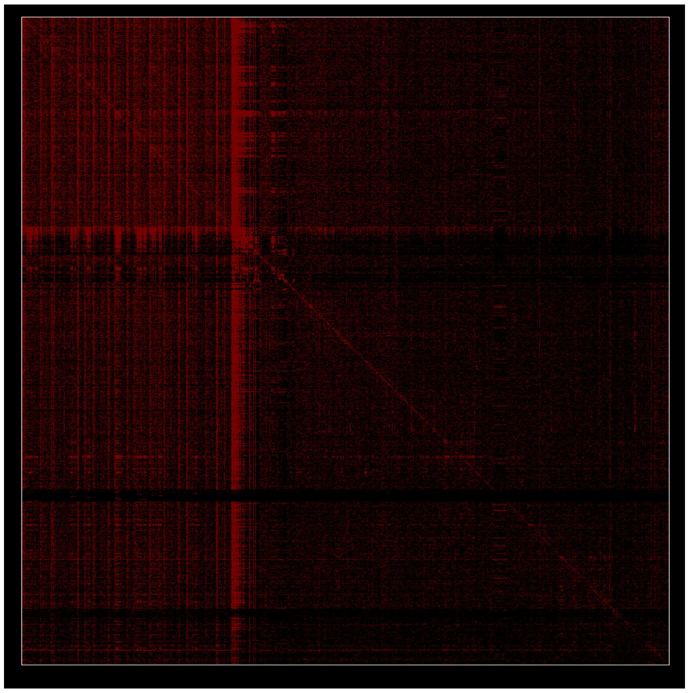
Better with $nv = 25$

```
>> [ci p] = spectral(A, struct('directed_distribution', 'pagerank', ...
      'directed', 1, 'pagerank_options', struct('c', 0.99), 'nv', 25));
  level=0; split=[240, 9674]; cv=0.01 str=lambda_1=3.05e-017; lambda_2=7.38e-00
  level=1; split=[276, 9398]; cv=0.01 str=lambda_1=2.13e-015; lambda_2=7.13e-0033
  level=2; split=[9340, 58]; cv=0.01 str=lambda_1=2.70e-015; lambda_2=8.29e-003
  level=3; split=[20, 9320]; cv=0.01 str=lambda_1=-3.31e-015; lambda_2=8.42e-003
  level=4; split=[20, 9300]; cv=0.01 str=lambda_1=-2.92e-015; lambda_2=8.42e-003
  ...
```

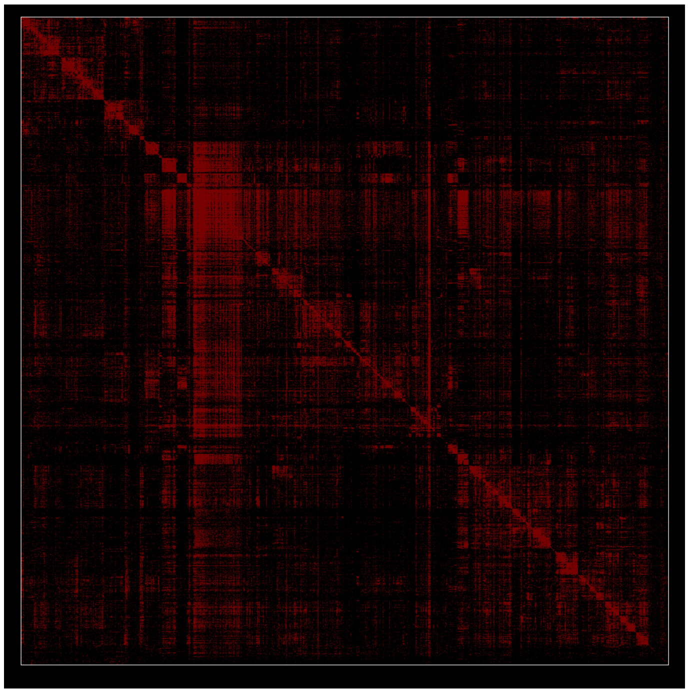
Best with $nv = 25$, using right-hand eigenvectors

```
>> [ci p] = spectral(A, struct('directed_distribution', 'pagerank', ...
      'directed', 1, 'pagerank_options', struct('c', 0.99), 'nv', 1, ...
      'problem', 'reig'));
  level=0; split=[1271, 8643]; cv=0.02 str=lambda_1=1.00e+000; lambda_2=9.88e-001
  level=1; split=[614, 657]; cv=0.01 str=lambda_1=1.00e+000; lambda_2=9.86e-001
  level=1; split=[7805, 838]; cv=0.03 str=lambda_1=1.00e+000; lambda_2=9.89e-001
  level=2; split=[1131, 6674]; cv=0.01 str=lambda_1=1.00e+000; lambda_2=9.89e-001
  level=3; split=[622, 509]; cv=0.01 str=lambda_1=1.00e+000; lambda_2=9.90e-001
  ...
```

Figure 5: The simple algorithm stagnates and cannot cut the webgraph in the first two instances. The "split" indicates the size of each partition. In the first case, we partition individual nodes – hardly worth solving an eigenvalue problem. By changing the permutation vector for the graph, we can improve the results and make significant cuts in the graph.



(a) Original Adjacency Matrix

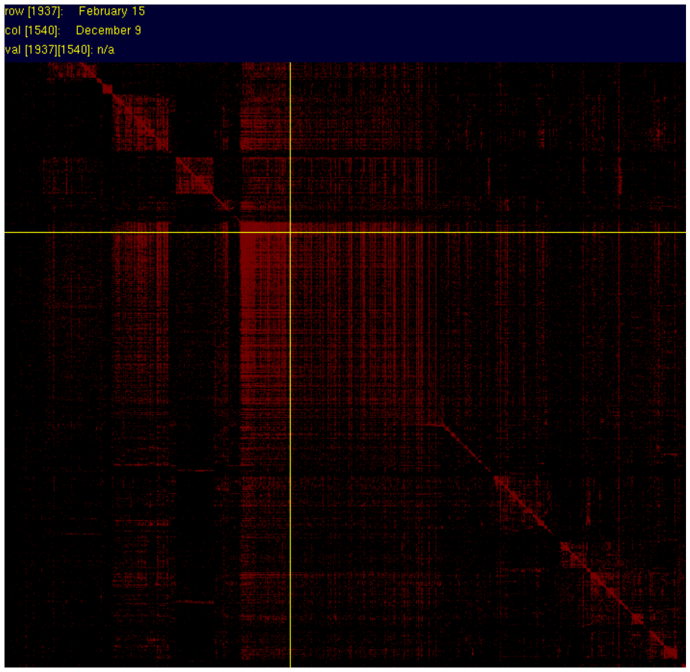


(b) Permuted Adjacency Matrix

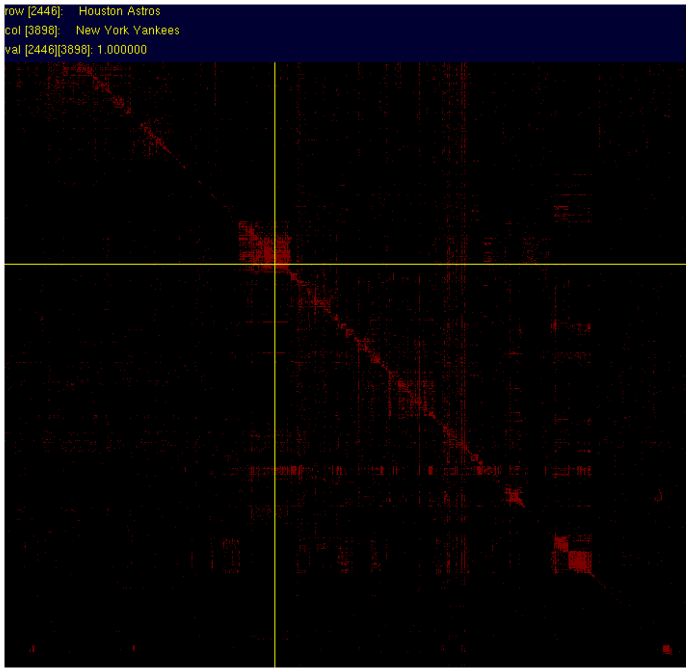
Figure 6: An ordering of the adjacency matrix for the neuro dataset. The ordering generates small blocks on the diagonal which correspond to more highly interconnected groups of neurons.

'Sahel'	'Distribution'	'Spear'
'Niger'	'Parameter'	'Nymph'
'Benin'	'Leonhard Euler'	'Aphrodite'
'Drought'	'Riemann zeta function'	'Dionysus'
'Niger River'	'Function (mathematics)'	'Hermes'
'West Africa'	'Complex number'	'Artemis'
'Mali'	'Square root'	'Persephone'
'Senegal'	'Neal Stephenson'	'Apollo'
'Chad'	'Limit (mathematics)'	'Oracle'
'Sahara'	'Natural logarithm'	'Zeus'
'Sahara Desert'	'Continuous function'	'Helen'
'Oasis'	'Impedance'	'Theseus'
'Muammar al-Qaddafi'	'Complex analysis'	'Demeter'
'Mauritania'	'Real number'	'Hera'
'Libya'	'Mathematical'	'Hades'
'Maghreb'	'Trigonometric function'	'Helios'
'Hashish'	'Distance'	'Hercules'
'Tangier'	'Fractal'	'Heracles'
'Nomad'	'Derivative'	'Gaia (mythology)'
'Bedouin'	'Category:Mathematical analysis'	'Centaur'
'Algeria'	'Vector'	'Ares'

Table 2: A set of proximate articles according to the final permutation of the wikipedia dataset. We see that each linear list of articles is topically coherent indicating a good permutation of the adjacency matrix.

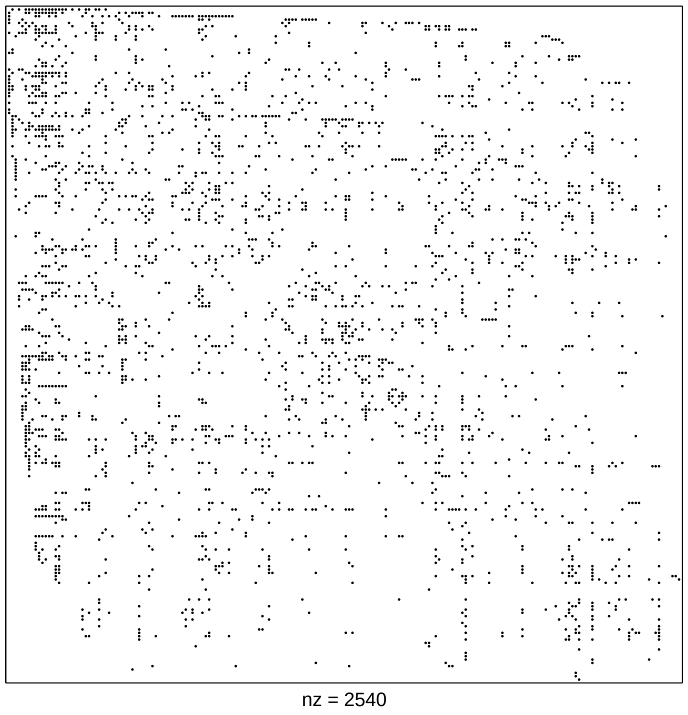

row [1937]: February 15
col [1540]: December 9
val [1937][1540]: n/a

(a) “Universal” Time Cluster

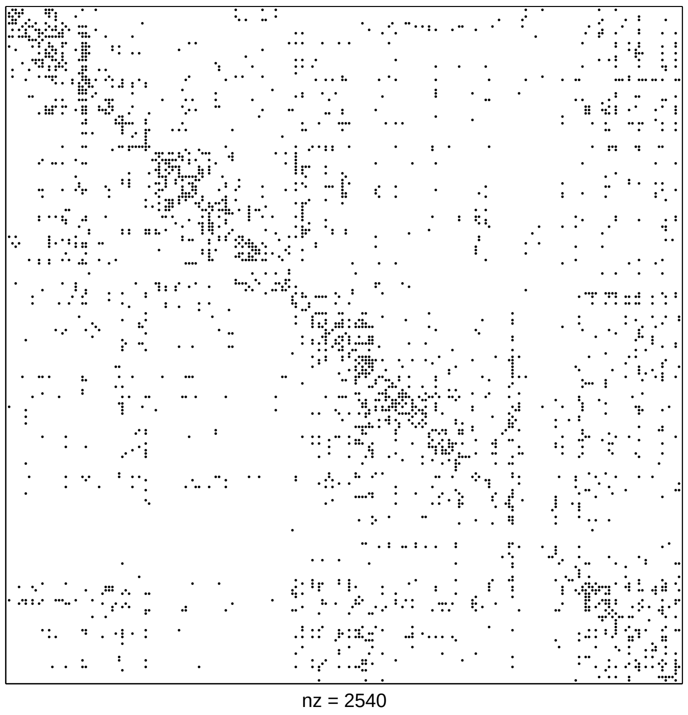

row [2446]: Houston Astros
col [3898]: New York Yankees
val [2446][3898]: 1.000000

(b) Baseball Cluster

Figure 7: An ordering of the adjacency matrix for the neuro dataset. The ordering generates small blocks on the diagonal which correspond to more highly interconnected groups of neurons.



(a) Original Adjacency Matrix



(b) Permuted Adjacency Matrix

Figure 8: An ordering of the adjacency matrix for the neuro dataset. The ordering generates small blocks on the diagonal which correspond to more highly interconnected groups of neurons.

7 Conclusion and Future Ideas

To recap, we explored the directed Laplacian as defined by a circulation on a graph. We proved that the directed Laplacian obeys the Cheeger inequalities based on a particular symmetrization of the adjacency matrix of a graph. Second, we built an implementation of the directed spectral clustering idea that follows from the directed Laplacian and used non-Laplacian eigenvectors to help partition the graphs. One of these ideas worked well on a graph where the eigenvectors the Laplacian were not helpful.

The directed Laplacian is promising. However, it tends to exhibit the same problems as the undirected Laplacian in that it chooses small cuts in many cases. One possibility is to use a semidefinite embedding of the directed graph to correct this problem. Recent empirical results have show that using the semidefinite embedding yields more balanced cuts of power-law graphs [9].

Our results suggest future work in a few directions. First, Fan Chung choose to preserve the symmetry of the volume operator on a graph cut,

$$\operatorname{vol} \partial S = \operatorname{vol} \partial \bar{S}.$$

For any circulation, this directly leads to computing the eigenvalues of a symmetric matrix,

$$\frac{F + F^T}{2}.$$

Instead, we might define a non-symmetric volume, or choose another property of the undirected volume operation to preserve. A second idea is to relate the concept of a circulation more directly with flows on the graph. Clearly, any $s - t$ flow gives rise to a circulation on the graph where we add an edge from t to s with the max-flow capacity. This fact immediately suggests a new algorithm for clustering under the constraint that a small set of nodes are in separate clusters. (Setup an $s - t$ max flow between one pair, solve, partition, and then repeat on each partition until all vertices are in different clusters.)[5]

References

[1] Fan Chung. Laplacians and the Cheeger inequality for directed graphs. *Annals of Combinatorics.*

[2] Amin Saberi, Paul Constantine. Lecture Notes for MS&E 337. October 31, 2005.

[3] James Demmel. CS267: Notes for Lecture 23, Graph Partitioning 2. April 9, 1999. `http://www.cs.berkeley.edu/~demmel/cs267/lecture20/lecture20.html`, accessed on 30 January 2006.

[4] Dengyong Zhou, Jiayuan Huang, and Bernhard Sch olkopf. Learning from labeled and unlabled data on a directed graph. In *Proceedings of the 22nd International Conference on Machine Learning.* 2005.

[5]It seems like there should be a more direct way to solve this problem and I would be surprised if there wasn't something known about it already.

[5] Fan Chung. Random walks and cuts in directed graphs. Accessed from `http://www.math.ucsd.edu/~fan/research.html` during December.

[6] Normalized Cuts and Image Segmentation. IEEE Transactions on Pattern Analysis and Machine Intelligence. Volume 22, Issue 8, 2005. Accessed from `http://www.cs.berkeley.edu/~malik/papers/SM-ncut.pdf`.

[7] Frank McSherry. Lecture for MS&E 337. Accessed from `http://www.stanford.edu/class/msande337/notes/talk.pdf`.

[8] William J. Stewart. *Introduction to the Numerical Solution of Markov Chains.* Princeton University Press, 1994.

[9] Kevin Lang. Finding good nearly balanced cuts in power-law graphs. Accessed from `http://research.yahoo.com/publication/YRL-2004-036.pdf`.