A New Algorithm For Community Detection In Large Social Networks

Natalia Dibbern, Juan Neri, Ph.D Thomas Laurent

November 28, 2017

Networks provide a useful representation for the investigation of complex systems, and make it possible to examine the intermediate-scale structure of such systems. Consequently, networks have attracted considerable attention in sociology, biology, computer science, and many other disciplines. The majority of intermediate-scale structure investigations have focused on community structure, where one decomposes the network into cohesive groups of nodes referred to as communities, which have a higher density of connections within than between them. With the rise of social media (e.g Facebook and Twitter), algorithms for community detection in large social networks have become increasingly important. In this work we propose a new algorithm for community detection. The algorithm proceeds by alternating three simple routines in an iterative fashion: diffusion, thresholding, and random sampling. We use our algorithm to detect communities in a large social network of 4.8 million users known as the LiveJournal Network. We compare the performance of our algorithm with a state of the art algorithm developed by Facebook. We also conduct a thorough study of the mathematical properties of the proposed algorithm. We show that the algorithm monotonically increases some quality function describing how good is the partition of the network.

1 Introduction

The goal of our algorithm is to take a graph $G = (V, E)$ and produce a partition $\{f_1, \ldots, f_R\}$ of V , subject to explicitly defined size constraints. The algorithm is capable of enforcing arbitrary size constraints in the form of lower bounds S_r and upper bounds T_r, such that $S_r \leq |f_r| \leq T_r$, $\forall r$. To initialize the algorithm, we begin by randomly assigning nodes to clusters, in proportions that are feasible with respect to these sizing constraints.

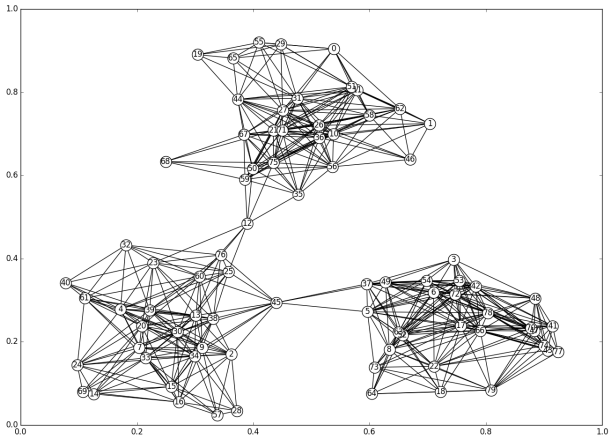

Figure 1: Inputs: A graph G, the target number of clusters R, and size constraints S_r, T_r for each cluster

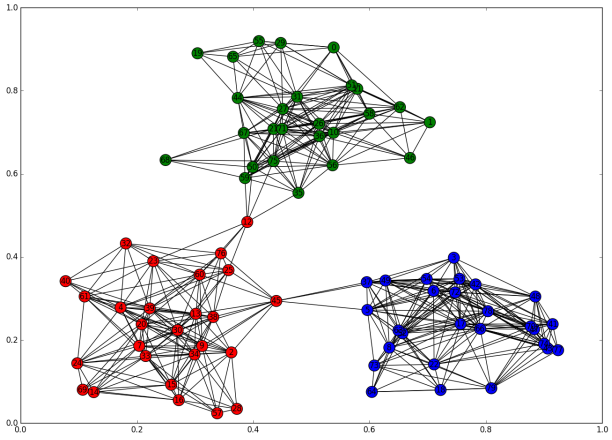

Figure 2: Output: A partition of the graph G into R clusters $\{f_1, \ldots, f_R\}$, subject to the constraint $S_r \leq |f_r| \leq T_r$

2 Mathematical Formulation of the Optimization Problem

Partitioning graphs at scale is a key challenge for any application that involves distributing a graph across disks, machines, or data centers. Graph partitioning is a very well studied problem with a rich literature, but existing algorithms typically can not scale to billions of edges, or can not provide guarantees about partition sizes [**?**].
We introduce an algorithm, for precisely partitioning massive graphs. Our algorithm is motivated by the challenge of performing graph predictions in a distributed system. Because this requires assigning each node in a graph to a physical machine with memory limitations, it is critically necessary to ensure the resulting partition shards do not overload any single machine. This is what they need to do in the Facebook Friends You May Know (FYMK) graph [**?**]. Due to the size of the graph and high query volume, Facebook has built a customized system for performing this task.

2.1 Notation

Let $G = (V, E)$ be an undirected graph with vertex set $V = \{v_1, \ldots, v_n\}$. In the following we assume that the graph G is weighted, that is each edge between two vertices v_i and v_j carries a non-negative weight $w_{ij} \geq 0$. The weighted adjacency matrix of the graph is the matrix $W = (w_{ij})$ $i, j = 1, \ldots, n$, where

$$w_{ij} = \begin{cases} 1 & \text{if } v_i \text{ adjacent to } v_j \\ 0 & \text{otherwise.} \end{cases},$$

As G is undirected we require $w_{ij} = w_{ji}$. The degree of a vertex $v_i \in V$ is defined as

$$d_i = \sum_{j=1}^{n} w_{ij}.$$

Note that, in fact, this sum only runs over all vertices adjacent to v_i, as for all other vertices v_j the weight w_{ij} is 0. The degree matrix D is defined as the diagonal matrix with the degrees $d_1, \ldots, d_n$ on the diagonal.

2.2 Minimizing the cut with size constraints

2.2.1 Set theoretical definition of the problem

For every $A \subset V$ we define

$$Cut(A, A^c) = \sum_{i \in A} \sum_{j \in A^c} w_{ij}.$$

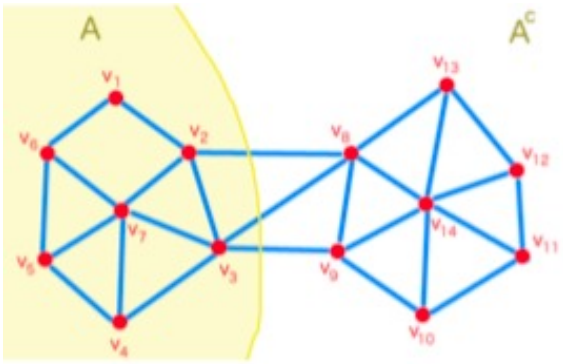

Figure 3: $Cut(A.A^C) = 3$

The cut of a subset A of the vertexes is the number of edges with one vertex in A and one vertex in A^c. In the context of graph partitioning, low values of cut correspond to subsets that accurately represent a cluster in the graph. In Fig. 2.2.1 for example, note that $Cut(A, A^C) = 3$ since there are three edges that connect vertices that are in A with vertices that are in A^C. It is intuitive then to note that what we want to do is to minimize this cut.

2.2.2 Importance of size constraints

One of the keys for the functionality and efficiency of our algorithm is that for each cluster r we have a set amount of vertices that it can contain.

Imagine we had no size constraints. Then our problem would simply be

$$\text{Minimize } \sum_r \text{Cut}(A_r, A_r^c) \tag{1}$$

$$\text{over all partitions } (A_1, \ldots, A_R) \text{of the vertex set } V \tag{2}$$

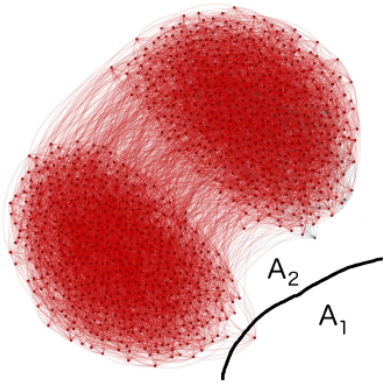

Figure 4: Incorrect partition solution

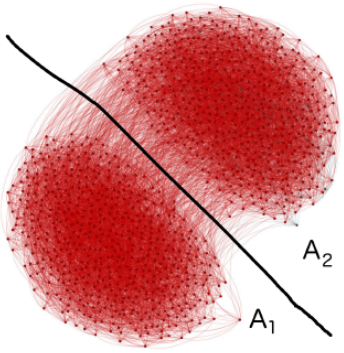

Figure 5: Correct partition solution

Without the size constraint our algorithm we might want to separate the partitions where one of the shards contains only one vertex (an outlier vertex to be more precise). This is what is happening in Figure 4. It is clear that the Cut value will be much lower than that in Figure 5, but it is also clear how Figure 5 contains the correct solution since Figure 4 fails to accurately partition the two shards. In general, we want our shards to be similar of size since this will avoid this problem. Hence, our model needs to enforce a size constraint.

Moreover adding a size constraint to the model is good for any application that involves distributing a graph across disks, machines, or data centers. The Facebook partitioning problem requires exactly this, where they need to partition their graph to be stored in different machines[**?**].

With this in mind, it is clear now that given S_r and T_r the size constraints of each cluster r, the objective of our algorithm can be expressed as

$$\text{Minimize } \sum_r \text{Cut}(A_r, A_r^c) \tag{3}$$

$$\text{such that } S_r \leq |A_r| \leq T_r \tag{4}$$

2.2.3 Vectorial formulation

Recall that a partition of V is defined by

$$V = A_1 \cup \cdots \cup A_r$$

$$\text{where } A_r \cap A_s = \emptyset \text{ if } r \neq s.$$

In this paper to each partition of the set V we associate a $n \times R$ partition matrix $F = (f_1, \ldots, f_R)$ such that

$$f_{ir} = \begin{cases} 1 & \text{if } v_i \in A_r \\ 0 & \text{if } v_i \notin A_r. \end{cases}$$

We will denote by f_r the r^{th} column of F.

With this definition, and noting that:

$$Cut(A, A^c) = \sum_{i \in A} \sum_{j \in A^c} w_{ij} = \sum_{i=1}^{n} \sum_{j=1}^{n} F_{ir}(1 - Fjr) w_{ij}$$

and that

$$\sum_r F_{ir} = 1$$

then we can rewrite (3)-(4) as:

$$\sum_r \mathrm{Cut}(f_r, f_r^c) = \sum_r \sum_{ij} w_{ij} F_{ir}(1 - f_{jr}) = \sum_r \sum_{ij} w_{ij} F_{ir} - \sum_r \sum_{ij} w_{ij} F_{ir} F_{jr} = \sum_{ij} w_{ij} - \sum_r \sum_{ij} w_{ij} F_{ir} F_{jr},$$

so problem (3) is equivalent to minimizing:

$$\sum_{ij} w_{ij} - \sum_r \sum_{ij} w_{ij} F_{ir} F_{jr}.$$

Since $\sum_{ij} w_{ij}$ is constant, then the above can be obtained by maximizing:

$$\sum_r \sum_{ij} w_{ij} F_{ir} F_{jr} = \sum_r \langle f_r, W f_r \rangle ,$$

where $\langle \cdot, \cdot \rangle$ denotes the dot product.

Therefore (3)-(4) becomes

$$\boxed{\begin{array}{c} \text{Maximize } \sum_r \langle f_r, W f_r \rangle \\ \text{such that } S_r \leq \sum_i f_{ir} \leq T_r \\ F = (f_1, \ldots, f_R) \text{ a partition of } V \end{array}} \quad \begin{array}{l} (5) \\ (6) \\ (7) \end{array}$$

2.3 Constructing a new similarity matrix Ω

Even though the above defines a good model the problem in question, moving forward, we are not going to use the W similarity matrix in our model. We will use a "smooth" matrix to replace it. We will call it Ω and define it as follows.

Definition 1. $\Omega_\alpha = (\mathbb{I} + \frac{\alpha}{1-\alpha} L_{sym})^{-1}$, *where* $L_{sym} = \mathbb{I} - D^{-1/2} W D^{-1/2}$ *be the symmetric normalized graph Laplacian [?].*

Remark: For convenience, we will refer to Ω_α as Ω.

Therefore, instead of solving problems (3)-(5) we will solve the following:

$$\boxed{\begin{array}{c} \text{Maximize } \sum_r \langle f_r, \Omega f_r \rangle \\ \text{such that } S_r \leq \sum_i f_{ir} \leq T_r \\ F = (f_1, \ldots, f_R) \text{ a partition of } V \end{array}} \quad \begin{array}{l} (8) \\ (9) \\ (10) \end{array}$$

The objective of this new matrix is to accurately represent a measure of similarity between the vertexes of the graph G, while at the same time being positive definite. Let us now explain the motivation behind replacing W by Ω.

We are going to show that Ω_{ij} gives a better notion of similarity between vertex v_i and v_j. In order to do so we begin by analyzing the steady state stochastic process defined in the following theorem:

Theorem 1. *Let,*

$$\begin{cases} u^{k+1} = \alpha \ \left(D^{-1/2} W D^{-1/2}\right) \ u^k + (1-\alpha) \ \frac{\mathbb{I}_A}{|A|}, \\ u^0 = \textit{any vector.} \end{cases}$$

where u^k *is a subset of the vertexes of the graph* G, $\alpha \in [0,1]$ *and* $A \subset V$.

This process represents a combination of the following two processes:

1. $\alpha \ \left(D^{-1/2} W D^{-1/2}\right) \ u^k$ *the operator that gets applied to the vertexes with probability* α
2. $(1-\alpha)$ A *the vertexes get transported to the set* $A \subset V$ *with probability* $(1-\alpha)$

If we let $u^\infty = \lim_{k\to\infty} u^k$, *then*

$$u^\infty = \Omega \mathbb{I}_A.$$

Proof.

$$u^\infty = \alpha \ (D^{-1/2} W D^{-1/2}) \ u^\infty + (1-\alpha) \ \frac{\mathbb{I}_A}{|A|}$$

$$\left(\mathbb{I} - \alpha \ \left(D^{-1/2} W D^{-1/2}\right)\right) u^\infty = (1-\alpha) \ \frac{\mathbb{I}_A}{|A|}$$

$$\frac{1}{(1-\alpha)} \left(\mathbb{I} - \alpha \ \left(D^{-1/2} W D^{-1/2}\right)\right) u^\infty = \frac{\mathbb{I}_A}{|A|}$$

$$u^\infty = \left(\frac{\mathbb{I} - \alpha \ \left(D^{-1/2} W D^{-1/2}\right)}{1-\alpha}\right)^{-1} \frac{\mathbb{I}_A}{|A|}$$

$$u^\infty = \left(\frac{\mathbb{I}}{1-\alpha} - \frac{\alpha}{1-\alpha}(D^{-1/2} W D^{-1/2})\right)^{-1} \frac{\mathbb{I}_A}{|A|}$$

$$u^\infty = \left(\mathbb{I}\frac{1-\alpha+\alpha}{1-\alpha} - \frac{\alpha}{1-\alpha}(D^{-1/2} W D^{-1/2})\right)^{-1} \frac{\mathbb{I}_A}{|A|}$$

$$u^{\infty} = \left(\mathbb{I}(1+\frac{\alpha}{1-\alpha}) - \frac{\alpha}{1-\alpha}(D^{-1/2}WD^{-1/2})\right)^{-1} \frac{\mathbb{I}_A}{|A|}$$

$$u^{\infty} = \left(\mathbb{I} + \frac{\alpha}{1-\alpha}L_{sym}\right)^{-1} \frac{\mathbb{I}_A}{|A|}$$

$$u^{\infty} = \Omega\frac{\mathbb{I}_A}{|A|}$$

□

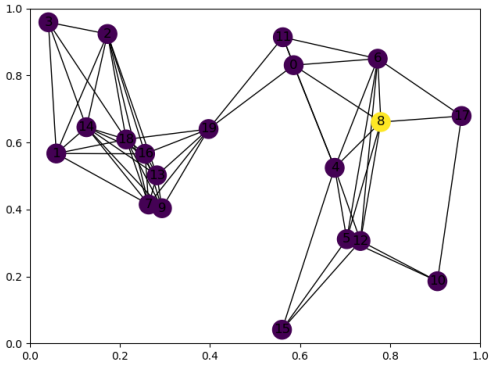

Figure 6: u^0, $A = \{v_8\}$

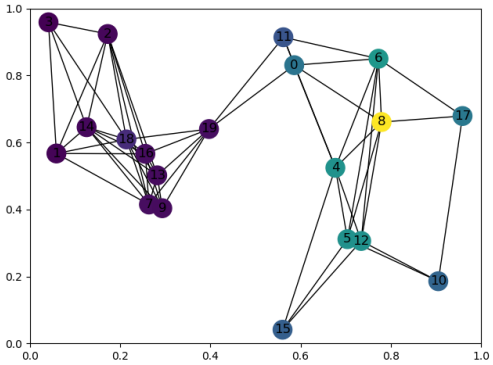

Figure 7: u^{20}, $A = \{v_8\}$

Let $v_j \in V$ and note that $\Omega_{ij} = (\Omega\mathbb{I}_{\{v_j\}})_i$. Hence, by making the use of Theorem 1 and letting $A = \{v_j\}$. Then

$$u^{\infty} = \lim_{k\to\infty} \left(D^{-1/2}WD^{-1/2}\right)\ u^k + (1-\alpha)\ \mathbb{I}_{v_j} = \Omega\mathbb{I}_{\{v_j\}}$$

So then:

$$u_i^{\infty} = (\Omega\mathbb{I}_{\{v_j\}})_i = \Omega_{ij}$$

It is important to note that the more "jumps" we do away from vertex v_j (meaning the more edges we need to be travel to arrive to vertex v_i) the lower the value of u^{∞} will be (see Figure 3 and 4). Hence it is intuitive to say that Ω encodes a notion of distance between vertex v_j and vertex v_i on the graph. This is a superior notion of similarity than the one we had with W. The reason for this is that W only tells you weather two vertices are adjacent or not, whereas Ω represents a measure of "smooth" distance on the graph, hence providing a relationship of every vertex to every other vertex and not only to its neighbors.

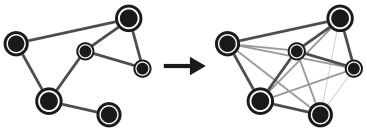

Figure 8: W (left) vs Ω (right)

3 $E(F) = \sum_r \langle f_r, \Omega f_r \rangle$ is convex

In this section we are going to show that the energy $E(F) = \sum_r \langle f_r, \Omega f_r \rangle$ is convex. This will be key to derive an efficient algorithm. In order for the energy to be convex we will show that our matrix Ω is positive definite, which is another motivation for using Ω instead of W since the latter is usually not positive definite in practice.

In order to do this we will need the following definition and Proposition from [**?**]. We reproduce the proof for completeness

3.0.1 Normalized Graph Laplacian

The symmetric normalized Laplacian is defined as follows:

$$L_{sym} = \mathbb{I} - D^{-1/2} W D^{-1/2}$$

and is a symmetric matrix defined in [**?**].

Since the degree matrix D is diagonal and positive, its reciprocal square root $D^{-1/2}$ is just the diagonal matrix whose diagonal entries are the reciprocals of the positive square roots of the diagonal entries of D.

Proposition 1. *The normalized Laplacians satisfy the following property:*

$$f' L_{sym} f = \frac{1}{2} \sum_{i,j=1}^{n} w_{ij} \left(\frac{f_i}{\sqrt{d_i}} - \frac{f_j}{\sqrt{d_j}} \right)^2$$

Proof. Part (1)

$$\begin{aligned}
f' L_{sym} f &= f'(\mathbb{I} - D^{-\frac{1}{2}} W D^{-\frac{1}{2}}) f \\
&= f' \mathbb{I} f - f'(D^{-\frac{1}{2}} W D^{-\frac{1}{2}}) f \\
&= \sum_{i=1}^{n} f_i^2 - \sum_{i,j=1}^{n} \frac{f_i w_{ij} f_j}{\sqrt{d_i}\sqrt{d_j}} \\
&= \frac{1}{2} \sum_{i,j=1} \frac{f_i^2}{d_i} w_{ij} + \frac{f_j^2}{d_j} w_{ij} - \sum_{i,j=1}^{n} \frac{f_i w_{ij} f_j}{\sqrt{d_i}\sqrt{d_j}} \\
&= \frac{1}{2} \sum_{i,j=1} w_{ij} \left(\frac{f_i^2}{d_i} - 2 \frac{f_i f_j}{\sqrt{d_i}\sqrt{d_j}} + \frac{f_j^2}{d_j} \right) \\
&= \frac{1}{2} \sum_{i,j=1} w_{ij} \left(\frac{f_i}{\sqrt{d_i}} - \frac{f_j}{\sqrt{d_j}} \right)^2
\end{aligned}$$

□

Proposition 2. Ω *is positive definite.*

Proof. Let λ be an eigenvalue of L_{sym} with corresponding eigenvector v. Then, the following holds:

$$L_{sym} v = \lambda v$$

Recall that in general a matrix $A \in \mathbb{R}^n \times \mathbb{R}^n$ is positive definite if $v^T A v \geq 0$ for all v.

Hence, multiplying both sides by v^T and with the use of Prop. 1, we get

$$v^T L_{sym} v = v^T(\lambda v) = \lambda \|v\|^2 > 0 \tag{11}$$

Next, consider the following

$$(\mathbb{I} + \frac{\alpha}{1-\alpha} L_{sym})v = v + \frac{\alpha}{1-\alpha} L_{sym} v = v + \frac{\alpha}{1-\alpha}\lambda v = (1 + \frac{\alpha}{1-\alpha}\lambda)v$$

Therefore, $(1 + \frac{\alpha}{1-\alpha}\lambda)$ is an egenvalue of Ω^{-1} which is positive since $\alpha \in [0, 1)$

Finally, let $\gamma = (1 + \frac{\alpha}{1-\alpha}\lambda)$. Then if we let x be its corresponding eigenvector,

$$\Omega^{-1} x = \gamma x$$
$$x = \gamma(\Omega^{-1})^{-1}\ x = \gamma \Omega x$$
$$x\frac{1}{\gamma} = \Omega\ x$$

So $\frac{1}{\gamma} > 0$ is an eigenvalue of Ω.

We then conclude that Ω is positive definite.

□

In order to prove convexity of $E(F)$ we need to show that its Hessian is positive semidefinite. Therefore, we first need to make use of the following lemma:

Lemma 1. $\nabla E(F) = \begin{bmatrix} \Omega f_1 \\ \vdots \\ \Omega f_r \end{bmatrix}$ *and* $\nabla^2 E(F) = \begin{bmatrix} \Omega & 0 & \dots & 0 \\ \vdots & \Omega & \dots & 0 \\ \vdots & 0 & \ddots & \vdots \\ 0 & \dots & 0 & \Omega \end{bmatrix}$

Proof. In general, if we let $f : \mathbb{R}^n \to R$ then for $h \in \mathbb{R}^n$

$$f(x + h) = f(x) + \vec{g} \cdot h + h^T A h + O(h^3)$$
$$\vec{g} = \nabla f(x)$$
$$\text{and } A = \nabla^2 f(x)$$

Recall that in our case we have

$$E(F) = \sum_{r=1}^{R} \langle f_r, \Omega f_r \rangle$$

where $E : \mathbb{R}^{n \times n} \to \mathbb{R}$.

Now consider the following

$$E(f_1 + df_1, \dots, f_r + df_r) = \sum_{r=1}^{R} \langle f_r + df_r, \Omega(f_r + df_r) \rangle$$

$$= E(f_1, \dots, f_r) + 2\sum_{r=1}^{R} \langle \Omega f_r, df_r \rangle + \sum_{r=1}^{R} \langle df_r, \Omega df_r \rangle$$

$$= E(f_1, \ldots, f_r) + \begin{bmatrix} 2\ \Omega\ f_1 \\ \vdots \\ \vdots \\ 2\ \Omega\ f_r \end{bmatrix} \cdot \begin{bmatrix} df_1 \\ \vdots \\ \vdots \\ df_r \end{bmatrix} + \begin{bmatrix} df_1 \\ \vdots \\ \vdots \\ df_r \end{bmatrix}^T \cdot \begin{bmatrix} \Omega & 0 & \ldots & 0 \\ \vdots & \Omega & \ldots & 0 \\ \vdots & 0 & \ddots & \vdots \\ 0 & \ldots & 0 & \Omega \end{bmatrix} \cdot \begin{bmatrix} df_1 \\ \vdots \\ \vdots \\ df_r \end{bmatrix}$$

From this we conclude that $\nabla E(F) = \begin{bmatrix} \Omega f_1 \\ \vdots \\ \Omega f_r \end{bmatrix}$ and $\nabla^2 E(F) = \begin{bmatrix} \Omega & 0 & \ldots & 0 \\ \vdots & \Omega & \ldots & 0 \\ \vdots & 0 & \ddots & \vdots \\ 0 & \ldots & 0 & \Omega \end{bmatrix}$ □

From the above result we can now prove that the Energy is convex.

Theorem 2. *$E(F) = \sum_{r=1}^{R} \langle f_r, \Omega f_r \rangle$ where $E : \mathbb{R}^{n \times n} \to \mathbb{R}$ is convex.*

Proof. Recall that for a smooth function to be convex, its Hessian must be semi-positive definite. Therefore, we just need to show that

$$\nabla^2 E(F) = \begin{bmatrix} \Omega & 0 & \ldots & 0 \\ \vdots & \Omega & \ldots & 0 \\ \vdots & 0 & \ddots & \vdots \\ 0 & \ldots & 0 & \Omega \end{bmatrix}$$

is positive semi-definite.

It suffices to show it for the case where $R = 2$ because its generalization to arbitrary values of R is straight forward. Therefore, let $\nabla^2 E(F) = \begin{bmatrix} \Omega & 0 \\ 0 & \Omega \end{bmatrix}$ and let $x \in \mathbb{R}^n$. Then

$$\begin{bmatrix} x \\ x \end{bmatrix}^T \cdot \begin{bmatrix} \Omega & 0 \\ 0 & \Omega \end{bmatrix} \cdot \begin{bmatrix} x \\ x \end{bmatrix} = 2x^T \Omega\ x > 0 \text{ since } \Omega \text{ is positive semi-definite.}$$

We conclude then that $\nabla^2 E(F)$ is positive semi-definite, implying that the energy is convex. □

$E(F)$ being convex implies that the value of E is always above its linearized approximation. We therefore need an algorithm that can guarantee that the linearized energy of a new partition $E(F^{k+1})$ be greater than the energy of the previous partition $E(F^k)$.

4 A monotonic algorithm

4.1 Maximizing a convex function subject to constraints

Define energy of a partition $F = (f_1, \ldots, f_r)$ as:

$$E(F) = \sum_r \langle f_r, W f_r \rangle .$$

If $E(F)$ were a smooth convex function, by definition, at each time step k we have:

$$E(F^{k+1}) \geq E(F^k) + \left\langle \nabla E(F^k), F^{k+1} - F^k \right\rangle$$

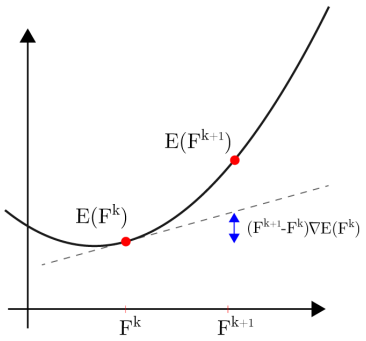

Figure 9: Low dimensionality convex function

So as long as $F^{k+1} = F^k + v$ where v is an ascent direction, that is $\langle \nabla E(f^k), v \rangle \geq 0$, then we have $E(F^{k+1}) \geq E(F^k)$.

Now let's denote by $\mathcal{C}$ the set of constraints (9)-(10), the following algorithm is monotonic:

$$\text{Given } F^k \text{ satisfying } \mathcal{C}, \text{ find } F^{k+1} \text{ satisfying } \mathcal{C} \text{ such that } \langle \nabla E(F^k), F^{k+1} - F^k \rangle \geq 0,$$

which can be expressed as:

$$\boxed{\begin{array}{c} \text{Given } F^k = (f_1^k, \ldots, f_R^k) \in \mathcal{C}, \text{ find } F^{k+1} = (f_1^{k+1}, \ldots, f_R^{k+1}) \in \mathcal{C} \text{ such that} \quad (12) \\ \langle \nabla E(F^k), F^{k+1} \rangle \geq \langle \nabla E(F^k), F^k \rangle \quad (13) \end{array}}$$

4.2 Maximizing the Energy

Recall that if $E(F)$ is a smooth convex function to maximize over the constraint set $\mathcal{C}$, the following holds:

$$E(f^{k+1}) \geq E(f^k) + \langle \nabla E(f^k), f^{k+1} - f^k \rangle,$$

which led to the conclusion that the problem can be described as:

Given $F^k = (f_1^k, \ldots, f_R^k) \in \mathcal{C}$, find $F^{k+1} = (f_1^{k+1}, \ldots, f_R^{k+1}) \in \mathcal{C}$ such that $\langle \nabla E(F^k), F^{k+1} \rangle \geq \langle \nabla E(F^k), F^k \rangle$

4.3 Greedy algorithm

We define $H^k = \nabla E(F^k)$ the heat bump at step k as shown in (3.1). By (Lemma 3) $H^k = \Omega F^k$ and $H_r^k = \nabla E(f_r^k)$ is the linearized energy at step k for the class r.

From Prop.**??**, the value of H^k can be obtained by repeated application of the stochastic process detailed in 3.1 until convergence.

We adopt the notation $\pi^k : \{1, \ldots, n\} \to \{1, \ldots, R\}$ for the membership function associated with the

partition F^k, where $\pi^k(i) = r$ indicates that i belongs to class r. With this, condition (13) becomes:

$$\sum_i H^k_{i,\pi^{k+1}(i)} \geq \sum_i H^k_{i,\pi^k(i)} \tag{14}$$

Theorem 3. $\sum_r \left\langle H^k_r, f^{k+1}_r \right\rangle \geq \sum_i \left\langle H^k_r, f^k_r \right\rangle \iff \sum_i H^k_{i,\pi^{k+1}(i)} \geq \sum_i H^k_{i,\pi^k(i)}$

Proof. It suffices to show that $\sum_r \langle H_r, f_r \rangle = \sum_i H_{i,\pi(i)}$

$$\sum_r \left\langle H^k_r, f^{k+1}_r \right\rangle \geq \sum_r \left\langle H^k_r, f^k_r \right\rangle$$

$$\sum_r \sum_i H^k_r(i) f^{k+1}_r(i) \geq \sum_r \sum_i H^k_r(i) f^k_r(i)$$

$$\sum_i \sum_r H^k_{i,r} f^{k+1}_r(i) \geq \sum_i \sum_r H^k_{i,r} f^k_r(i)$$

Now letting $\pi(i) : \{1, \ldots, n\} \to \{1, \ldots, R\}$ be the membership function, we have

$$\sum_i H^k_{i,\pi^{k+1}(i)} \geq \sum_i H^k_{i,\pi^k(i)}$$

□

The algorithm should therefore define a new membership function π^{k+1} that maps each vertex with entries in H to a class where the heat bump for that class in H are high. Some vertexes will be mapped to the same class by both π^k and π^{k+1}, however some subset $\Delta \subset V$ will "want" to switch class, for the value of H^k is higher in the new class than the old class.

$$\Delta = \{i \in V : \pi^k(i) \notin \arg\max_r H^k_{ir}\}$$

In general $\sum_i H^k_{i,\pi^{k+1}(i)}$ would be maximized by switching a nonempty subset of $\tilde{\Delta} \subset \Delta$ to the class that has the highest value, as follows:

$$\pi^{k+1}(i) = \begin{cases} \arg\max_r H^k_{ir} & \text{if } i \in \tilde{\Delta} \\ \pi^k(i) & \text{otherwise} \end{cases}$$

Note that the above satisfies (14) with strict inequality. The question now is to find a subset $\tilde{\Delta} \subset \Delta$ such that the new partition still satisfies the size constraints. We also want this subset $\tilde{\Delta}$ to be as big as possible, otherwise the algorithm is going to stagnate.

In other words, Δ is the subset of vertexes which would change class under plain thresholding. The problem is that if all vertexes change classes typically we will break the balance constraint. The nice thing is that we don't need all the vertexes in Δ to change class: any subset of Δ, even if it consist on only one vertex, will lead to a decrease in the cut value. The goal is then to look for the subset $\tilde{\Delta} \subset \Delta$ that leads to the biggest gain in equation (14) while not breaking the size constraint. So this lead to the optimization problem:

$$\text{Find } \tilde{\Delta} \subset \Delta \text{ that maximize } \sum_i H^k_{i,\pi^{k+1}(i)} \tag{15}$$

$$\text{while not breaking the size constraint} \tag{16}$$

And this problem can be solved with a small linear program with $R(R-1)$ unknowns (the number of vertices that are allowed to switch classes).

4.4 The algorithm in short

Choose F^0 to be a random partition of the graph that satisfies the size constraints.

At each step k compute:

- Find $H^k = \Omega F^k$, the heat bump of the current partition F^k by repeated application of (5)
- Threshold the matrix H^k, to find Δ, the vertexes that would increase the E if switched to a new class
- Find $\tilde{\Delta} \subset \Delta$ which contains the nodes that when switched classes will guarantee that the size constraints are met
- Get F^{k+1} by assigning the new class to all the vertexes in $\tilde{\Delta}$

Since Ω is positive definite, this algorithm gives a monotonic algorithm for the minimization problem (3)-(4) and F is guarantee to converge to a partition that minimizes the cut.