

CSC 5450 Randomness and Computation

Week 12: Randomized Rounding

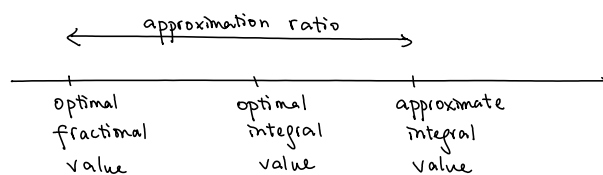
- Plan
- ① LP: set cover, group Steiner tree
 - ② Spectral: Cheeger's inequality
 - ③ SDP: maximum cut, correlation clustering.
-

Randomized Rounding

A popular method in designing approximation algorithms is to formulate the problem by a mathematical program (e.g. integer linear program, quadratic program), and then consider a relaxation of this program so that it can be solved in polynomial time.

Since it is just a relaxation, the optimal value of the relaxation could be smaller than the optimal value of the original minimization problem, and the optimal solutions may be fractional.

But we can use the optimal value of the relaxation as a lower bound, and try to find an integral solution that is "close" to it, and in this way we could prove that the integral solution found is a good approximate solution.



If we can upper bound the ratio between the approximate integral solution to the fractional optimal solution, then we can upper bound the approximation ratio.

Generally speaking, rounding is a procedure to turn a fractional solution into an integral solution while having some performance guarantee.

Randomized rounding is one way of doing rounding, in which we use a random procedure to turn the fractional solution into an integral solution. For some problems, the algorithm and the analysis are simple and the bound obtained is optimal.

Today, we will see three different types of relaxations (LP, spectral, SDP) and see how randomized rounding can be used to obtain very interesting results.

Set Cover [1, Chapter 14] [2, Chapter 1]

In the minimum set cover problem, we are given a ground set $U = \{u_1, u_2, \dots, u_n\}$ of n elements, and a collection of subsets $\mathcal{S} = \{S_1, S_2, \dots, S_m\}$ where each $S_i \subseteq U$, and each S_i has a cost c_{S_i} . The objective is to pick a subcollection of minimum total cost such that each element is contained in some subset in the subcollection.

This is a general problem that captures many covering problems as special cases, e.g. vertex cover.

The problem can be formulated as an integer linear program, where each subset S has an indicator variable x_S :

$$\begin{aligned} \min \quad & \sum_S c_S x_S \\ \text{subject to: } & \sum_{S: u \in S} x_S \geq 1 \quad \text{for each element } u. \\ & x_S \in \{0, 1\} \quad \text{for each subset } S. \end{aligned}$$

This is an integer linear program and is NP-complete to solve. So, instead of writing the constraints $x_S \in \{0, 1\}$, we replace it by the constraint that $0 \leq x_S \leq 1$.

Then, the resulting linear program can be solved in polynomial time.

Given an optimal fractional solution of total cost $C := \sum_S c_S x_S$, we would like to design a rounding algorithm to turn it into an integral solution, while the total cost is still small.

The idea here is to think of each x_S as the probability that S is chosen.

So, the algorithm is to pick each subset S with probability x_S , independently for each S .

We need to analyze two things: (1) the cost (2) the feasibility.

First, we calculate the expected total cost of the algorithm.

By linearity of expectation, $E[\text{total cost}] = \sum_{S \in \mathcal{S}} c_S \cdot \Pr(S \text{ is chosen}) = \sum c_S \cdot x_S = C$.

So, the expected total cost is equal to the optimal fractional value.

Next, we calculate the probability that an element is covered:

$$\begin{aligned} \Pr(u \text{ is not covered}) &= \prod_{S: u \in S} (1 - \Pr(S \text{ is chosen})) && \text{by the constraint} \\ &= \prod_{S: u \in S} (1 - x_S) \leq \prod_{S: u \in S} e^{-x_S} = e^{-\sum_{S: u \in S} x_S} \leq e^{-1}. \end{aligned}$$

So, $\Pr(u \text{ is covered}) \geq 1 - \frac{1}{e}$, and we expect a constant fraction of the elements is covered.

To cover all the elements, we just repeat the above procedure $O(\log n)$ times and use the union of the solutions. Then, the total cost is $O(\log n) \cdot C$, an $\log(n)$ -approximation.

By repeating $O(\log n)$ times, $\Pr(u \text{ is not covered in the union}) \leq (e^{-1})^{O(\log n)} = 1/\text{poly}(n)$.

By a union bound, $\Pr(\text{all elements are covered}) \geq 1 - \frac{1}{\text{poly}(n)}$.

By a union bound, $\Pr(\text{all elements are covered}) \geq 1 - \frac{1}{\text{poly}(n)}$.

Therefore, with good probability, we can find a set cover of total cost $O(\log n)$ times the optimal in polynomial time.

Two remarks: ① There is a simple greedy algorithm having the same guarantee.

② This approximation guarantee is optimal assuming $P \neq NP$, i.e. for some constant c , it is NP-hard to obtain a $c \cdot \log n$ approximation algorithm for the minimum set cover problem.

Group Steiner Tree (sketch) [3, Chapter 3.3]

One advantage of the randomized rounding approach is its flexibility in solving more general problems. We briefly mention one interesting example.

In the group Steiner tree problem, we are given an undirected graph $G=(V,E)$ and some subsets $S_1, S_2, \dots, S_K \subseteq V$, and each edge e has a cost c_e . The objective is to find a tree T of minimum total cost such that $T \cap S_i \neq \emptyset$ for all i , i.e. T intersects every group S_i .

This is a generalization of minimum set cover.

Consider a special case where the graph itself is a tree, every group is a subset of the leaf nodes, and the objective is to find a min-cost tree connecting each group to the root.

(The general case can be reduced to this special case with some loss in the approximation ratio.)

Consider the following linear programming relaxation for the problem.

$$\min \sum_e c_e x_e$$

$$\sum_{e \in \text{cut}} x_e \geq 1 \quad \text{for each cut that separates the root from some group.}$$

$$x_e \geq 0$$

The randomized rounding algorithm picks each edge e with probability x_e / x_f , where f is the unique parent edge connecting e to the root. Then, an edge is included in the tree if it has a path to the root.

To compute the expected cost, note that an edge e is included if the whole path

$e = e_1, e_2, \dots, e_p$ to the root has been picked. This happens with probability:

$$\frac{x_{e_1}}{x_{e_2}} \cdot \frac{x_{e_2}}{x_{e_3}} \cdot \dots \cdot \frac{x_{e_{p-1}}}{x_{e_p}} \cdot x_{e_p} = x_e.$$

Hence, the expected cost is just $\sum c_e x_e$, which is the optimal value of the linear program.

Suppose each edge is included in the tree with probability x_e independently, then the same analysis as in set cover would show that $\Pr(\text{group } g \text{ is not covered}) \leq \frac{1}{e}$, and repeating $O(\log(n))$ times would do as in the set cover problem.

However, they are not independent, since two edges may share some edges in their paths to the root.

It is still possible to show that the algorithm will work, by bounding the dependence of the variables and applying some inequality that we have not discussed (Jensen's inequality in this example). See [3,4] for details.

There are several general techniques to deal with (limited) dependent random variables.

These techniques could be very useful in some problems; see [3].

Cheeger's Inequality

This is the third time we talk about Cheeger's inequality, and we will finally see a proof.

We will only introduce just enough definitions, and also we assume the graph is d-regular.

Recall that the conductance of a set S is defined as $\phi(S) = \frac{|\delta(S)|}{d \cdot |S|}$ where $\delta(S)$ is the set of edges with one endpoint in S and another endpoint in $V-S$.

And the conductance of a graph is defined as $\phi(G) = \min_{S: |S| \leq |V|/2} \phi(S)$.

Let A be the adjacency matrix of an undirected graph.

Let $L = I - \frac{A}{d}$ be the normalized Laplacian matrix of the graph.

Let $\lambda_1 \leq \lambda_2 \leq \dots \leq \lambda_n$ be the eigenvalues of L . It is known that $\lambda_1 = 0$ and $\lambda_n \leq 2$.

In fact, the all-one vector $\vec{1}$ is an eigenvector of L with eigenvalue 0.

Theorem (Cheeger's inequality) $\frac{\lambda_2}{2} \leq \phi(G) \leq \sqrt{2\lambda_2}$.

(This is different from what we have seen before, because we use a different matrix.)

There is also an efficient algorithm to achieve the performance guarantee, and it is called the spectral partitioning algorithm.

① Compute the second eigenvector x of L (the eigenvector corresponding to the second smallest eigenvalue)

② Sort the vertices so that $x(i) \geq x(j) \geq \dots \geq x(n)$ (where $n = |V|$ is the number of vertices)

$$S_i = \begin{cases} \{1, \dots, i\} & \text{if } i \leq n/2 \\ \{i+1, \dots, n\} & \text{if } i > n/2 \end{cases}$$

③ Return $\min_i \phi(S_i)$.

This algorithm can be implemented easily and efficiently, and surprisingly it also has good performance in practical applications (e.g. clustering, image segmentation, etc).

Rayleigh Quotient

The main tool in connecting eigenvalues and eigenvectors to optimization problem is the Rayleigh quotient,

$$\text{which is defined as } R(x) := \frac{x^T M x}{x^T x} = \frac{\sum_{i,j} M(i,j) x(i) x(j)}{\sum_i x(i)^2}$$

Let M be a real symmetric matrix with eigenvalues $\lambda_1 \leq \lambda_2 \leq \dots \leq \lambda_n$ with corresponding orthonormal eigenvectors x_1, x_2, \dots, x_n .

Claim $\lambda_1 = \min \frac{x^T M x}{x^T x}$

proof Let $x \in \mathbb{R}^n$. We can write $x = c_1 x_1 + c_2 x_2 + \dots + c_n x_n$, as x_i form a basis.

$$\begin{aligned} \text{Then } x^T M x &= (c_1 x_1 + \dots + c_n x_n)^T M (c_1 x_1 + \dots + c_n x_n) \\ &= (c_1 x_1 + \dots + c_n x_n)^T (c_1 \lambda_1 x_1 + \dots + c_n \lambda_n x_n) \quad \text{since } x_i \text{ is an eigenvector with eigenvalue } \lambda_i \\ &= c_1^2 \lambda_1 + c_2^2 \lambda_2 + \dots + c_n^2 \lambda_n \quad \text{since } \langle x_i, x_j \rangle = 0 \text{ for } i \neq j \text{ and } \langle x_i, x_i \rangle = 1 \quad \forall i \end{aligned}$$

$$\text{Similarly, } x^T x = (c_1 x_1 + \dots + c_n x_n)^T (c_1 x_1 + \dots + c_n x_n) = c_1^2 + \dots + c_n^2.$$

$$\text{So, } \frac{x^T M x}{x^T x} = \frac{c_1^2 \lambda_1 + \dots + c_n^2 \lambda_n}{c_1^2 + \dots + c_n^2} \geq \frac{\lambda_1 (c_1^2 + \dots + c_n^2)}{c_1^2 + \dots + c_n^2} = \lambda_1. \quad \square$$

The advantage of this characterization is that it can be extended to other eigenvalues.

Let T_k be the set of vectors that are orthogonal to x_1, x_2, \dots, x_{k-1} .

$$\lambda_k = \min_{x \in T_k} \frac{x^T M x}{x^T x}.$$

proof Let $x \in T_k$. Write $x = c_1 x_1 + \dots + c_n x_n$.

Observe that $c_i = \langle x, x_i \rangle$. Since $x \in T_k$, $c_1 = c_2 = \dots = c_{k-1} = 0$.

$$\text{Then } \frac{x^T M x}{x^T x} = \frac{\sum_{i=k}^n c_i^2 \lambda_i}{\sum_{i=k}^n c_i^2} \geq \lambda_k. \quad \square$$

Laplacian Matrix

$$x^T L x$$

We will use the above claim to study the second eigenvalue, i.e. $\lambda_2 = \min_{x \perp \vec{1}} \frac{x^T L x}{x^T x}$.

Before that, let's see why we prefer Laplacian matrices over adjacency matrices.

There are at least two reasons:

- ① For any graph (not only d -regular graph), the all-one vector $\vec{1}$ is an eigenvector of L with eigenvalue 0 (the smallest eigenvalue of L).

Then, $\lambda_2 = \min_{x \perp \vec{1}} \frac{x^T L x}{x^T x}$. So we know we are looking for minimizer over all vectors x with $\sum x(i) = 0$.

- ② $x^T L x$ has a very nice quadratic form: $x^T L x = \frac{1}{d} \sum_{ij \in E} (x(i) - x(j))^2$.

There is a good way to see it, although you can directly check it.

Let $e = ij$ be an edge. Let $L_e = \frac{1}{d} \begin{bmatrix} & i & j \\ i & \frac{1}{d} & -\frac{1}{d} \\ j & -\frac{1}{d} & \frac{1}{d} \end{bmatrix}$

Then $L = \sum_{e \in E} L_e$. Then $x^T L x = x^T \left(\sum_{e \in E} L_e \right) x = \sum_{e \in E} x^T L_e x = \frac{1}{d} \sum_{ij \in E} (x(i) - x(j))^2$.

Spectral Relaxation

The first inequality is called the "easy" direction, and the second inequality is called the "hard" direction.

So, naturally we prove the easy direction first.

Recall that $\lambda_2 = \min_{x \perp \vec{1}} \frac{x^T L x}{x^T x} = \min_{x \perp \vec{1}} \frac{\sum_{ij \in E} (x(i) - x(j))^2}{d x^T x}$

To upper bound λ_2 , we just need to find a vector $x \perp \vec{1}$ and compute its Rayleigh quotient.

To get some intuition, let say $\phi(G) = \phi(S)$ and $|S| = n/2$.

We consider the "binary" solution: $x(i) = \begin{cases} +1 & \text{if } i \in S \\ -1 & \text{if } i \notin S \end{cases}$

Since $|S| = n/2$, $\sum_{i \in V} x(i) = 0$, and thus $x \perp \vec{1}$.

Then $\lambda_2 \leq \frac{\sum_{ij \in E} (x(i) - x(j))^2}{d \sum_{i \in V} x(i)^2} = \frac{4|E(S)|}{d|V|} \leq \frac{2|E(S)|}{d|S|} = 2\phi(S)$.

For general S , we consider the binary solution: $x(i) = \begin{cases} +1/|S| & \text{if } i \in S \\ -1/(|V|-|S|) & \text{if } i \notin S \end{cases}$

Then $x \perp \vec{1}$, and $\lambda_2 \leq \frac{\sum_{ij \in E} (x(i) - x(j))^2}{d \sum_{i \in V} x(i)^2} = \frac{|E(S)| \cdot \left(\frac{1}{|S|} + \frac{1}{|V|-|S|} \right)^2}{d \left(|S| \cdot \frac{1}{|S|^2} + (|V|-|S|) \cdot \frac{1}{(|V|-|S|)^2} \right)} = \frac{|E(S)| \cdot |V|}{d \cdot |S| \cdot |V-S|} \leq 2\phi(S)$.

This proves the easy direction.

To summarize, if there is a sparse cut, then λ_2 is small.

A consequence is that if λ_2 is large, then we know that G has no sparse cut.

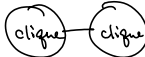
The Hard Direction: Intuition

The Hard Direction: Intuition

In the minimization problem $\min_{x \perp \vec{1}} \frac{\sum_{i,j \in E} (x(i) - x(j))^2}{d \sum_i x(i)^2}$, if we can only search for "binary" solutions,

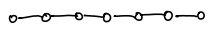
then we are essentially optimizing over the conductances.

Unfortunately, we are optimizing over a much larger domain (otherwise the problem is not efficiently solvable), and there could be some very non-binary solutions (very "smooth" vector), for which it is not clear how to find a sparse cut from it.

To get some feeling, suppose we are given a graph like .

Observe that the optimizer tries to minimize the average $(x(i) - x(j))^2 / (x(i)^2 + x(j)^2)$.

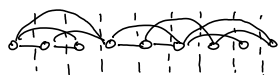
In this case, it is not good to "split" the vertices in a clique, because there are so many edges within it. So, we would expect that the values in each clique are very similar, while the two cliques would have different values so that $x \perp \vec{1}$. Hence, we expect that the minimizer would look very similar to a binary vector, and we can easily find a good cut with $\phi(S) \approx \lambda_2$.

Now, suppose we are given a graph like , then the minimizer can do much better by making each edge very short, while the values decrease smoothly from +1 to -1, in which case $\lambda_2 \ll \phi(G)$.

The key of Cheeger's inequality is to show that λ_2 cannot be much smaller than $\phi(G)$.

In other words, if λ_2 is small, then we can extract a somewhat sparse cut from the eigenvector.

We can think of the optimizer "embedding" the graph into a line, while most edges are short.



Then it should be the case that some threshold gives a sparse cut.

The Hard Direction: Proof

The first step is to preprocess the second eigenvector so that at most half the entries are nonzero.

This would guarantee that the output set S satisfies $|S| \leq |V|/2$.

This step is simple. Without loss of generality we assume there are fewer positive entries in x than negative entries.

Consider the following vector y :
$$y(i) = \begin{cases} x(i) & \text{if } x(i) \geq 0 \\ 0 & \text{if } x(i) < 0 \end{cases}$$

Claim $R(y) \leq R(x)$.

proof $(Ly)(i) = y(i) - \sum_{j \in N(i)} \frac{y(j)}{d} \leq x(i) - \sum_{j \in N(i)} \frac{x(j)}{d} = (Lx)(i) = \lambda_2 \cdot x(i) \quad \forall i \text{ with } y(i) > 0$.

Therefore, $y^T L y = \sum_{i \in V} y(i) \cdot (Ly)(i) \leq \sum \lambda_2 x(i)^2 = \sum_i \lambda_2 y(i)^2$, proving the claim. \square

There is a very elegant argument to make the above intuition precise: just pick a random threshold.

Lemma Given any y , there exists a subset $S \subseteq \text{supp}(y)$ such that $\phi(S) \leq \sqrt{2R(y)}$, where $\text{supp}(y) = \{i \mid y(i) \neq 0\}$.

Proof We can assume that $-1 \leq y_i \leq +1$ for all i , by scaling y if necessary.

Let $t \in (0, 1]$ be chosen uniformly at random.

Let $S_t = \{i \mid y_i^2 \geq t\}$. Then $S_t \subseteq \text{supp}(y)$ by construction.

We analyze the expected value of $|S(S_t)|$ and the expected value of $|S_t|$.

$$\begin{aligned} E(|S(S_t)|) &= \sum_{ij \in E} [\Pr(\text{the edge } ij \text{ is cut})] \quad \text{by linearity of expectation} \\ &= \sum_{ij \in E} [\Pr(y_i^2 < t \leq y_j^2)] \\ &= \sum_{ij \in E} |y_j^2 - y_i^2| \\ &= \sum_{ij \in E} |y_i - y_j| |y_i + y_j| \\ &\leq \sqrt{\sum_{ij \in E} (y_i - y_j)^2} \sqrt{\sum_{ij \in E} (y_i + y_j)^2} \quad \text{by Cauchy-Schwarz } \langle a, b \rangle \leq \|a\| \cdot \|b\| \\ &\leq \sqrt{\sum_{ij \in E} (y_i - y_j)^2} \sqrt{2 \sum_{ij \in E} (y_i^2 + y_j^2)} \\ &= \sqrt{\sum_{ij \in E} (y_i - y_j)^2} \sqrt{2d \sum_{i \in V} y_i^2} \\ &= \sqrt{2R(y)} \cdot \left(d \sum_{i \in V} y_i^2 \right)^{1/2}. \end{aligned}$$

$$\text{Also, } E[|S_t|] = \sum_{i \in V} \Pr[y_i^2 \geq t] = \sum_{i \in V} y_i^2$$

$$\text{Therefore, } \frac{E[|S(S_t)|]}{E[d|S_t|]} \leq \sqrt{2R(y)}.$$

$$\text{This means that } E[|S(S_t)| - \sqrt{2R(y)} \cdot d \cdot |S_t|] \leq 0.$$

$$\text{Hence, there exists } t \text{ such that } \frac{|S(S_t)|}{d \cdot |S_t|} \leq \sqrt{2R(y)} \leq \sqrt{2R(x)} = \sqrt{2\lambda_2}. \quad \square$$

Combining the claim and the lemma proves Cheeger's inequality.

And the proof shows that the spectral partitioning algorithm achieves the performance guarantee,

because the output set S_t is a "threshold" set that the algorithm searches.

Maximum Cut [1, Chapter 26] [2, Chapter 6]

There is a simple $\frac{1}{2}$ -approximation algorithm for the maximum cut problem.

Goemans and Williamson introduced semidefinite programming to the design of approximation algorithms.

One can write a quadratic program for the maximum cut problem.

$$\begin{aligned} \max \quad & \frac{1}{2} \sum_{i,j \in E} (1 - x_i \cdot x_j) \\ & x_i^2 = 1 \quad \text{for } 1 \leq i \leq n \end{aligned}$$

The optimal solutions to this quadratic program correspond to maximum cuts in the graph.

This shows nothing, just that this quadratic program is NP-hard.

The idea of Goemans and Williamson is to relax this quadratic program as a semidefinite

$$\begin{aligned} \max \quad & \frac{1}{2} \sum_{i,j \in E} (1 - y_{ij}) \\ & y_{ii} = 1 \quad \text{for } 1 \leq i \leq n \\ & Y \succeq 0 \quad \text{where } Y \text{ is the } n \times n \text{ matrix whose } (i,j)\text{-th entry is } y_{ij} \end{aligned}$$

The notation $Y \succeq 0$ means that Y is a symmetric positive semidefinite matrix (PSD matrix).

The following are three equivalent definitions for a matrix Y to be positive semidefinite:

- ① all eigenvalues of Y are non-negative;
- ② $x^T Y x \geq 0$ for all $x \in \mathbb{R}^n$
- ③ $Y = V^T V$ for some $V \in \mathbb{R}^{n \times n}$.

This is a convex program, and can be solved in polynomial time to any precision, by the ellipsoid method or the interior point method.

To analyse this program, it is more convenient to think of it as a vector program.

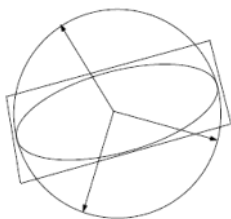
The condition $Y \succeq 0$ can be rewritten as $Y = V^T V$. Let v_i be the i -th column of V .

Then the above program is equivalent to:

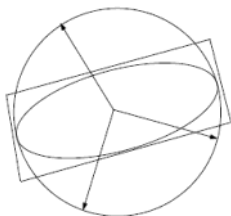
$$\begin{aligned} \max \quad & \frac{1}{2} \sum_{i,j \in E} (1 - v_i \cdot v_j) \\ & v_i^2 = 1 \quad \text{for } 1 \leq i \leq n \\ & v_i \in \mathbb{R}^n \quad \text{for } 1 \leq i \leq n \end{aligned}$$

So each vertex is associated an n -dimensional vector.

Geometrically, the vectors are "spread out" as much as possible.



(picture from [2])



(picture from [2])

Goemans and Williamson showed how to construct a cut with at least $0.878 \text{OPT}_{\text{SDP}}$ edges,

OPT_{SDP} denotes the optimal value of the SDP.

The algorithm is very simple, again by randomized rounding.

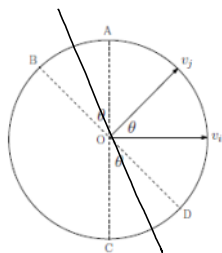
Let r be a random n -dimensional unit vector.

Let $S = \{i \mid v_i \cdot r \geq 0\}$ and return $\delta(S)$ as the solution.

To analyse the algorithm, we compute $E[\# \text{ of edges in } \delta(S)]$ and compare it to OPT_{SDP} .

By linearity of expectation, it is equal to $\sum_{ij \in E} \Pr(ij \text{ is separated}) = \sum_{ij \in E} \theta_{ij} / \pi$.

Pictorially, the probability that ij is separated is equal to the probability r is "between" the arc AB and the arc CD (see the picture from [2]).



On the other hand, the objective can be written as $\sum_{ij} \frac{1}{2} (1 - \cos \theta_{ij})$.

So the approximation ratio is $\frac{\sum_{ij \in E} \theta_{ij} / \pi}{\sum_{ij \in E} \frac{1}{2} (1 - \cos \theta_{ij})} \geq \min_{ij \in E} \frac{2 \theta_{ij}}{\pi (1 - \cos \theta_{ij})} > 0.87856$.

This algorithm can be derandomized to give a polytime deterministic 0.878 -approximation.

Surprisingly, this is optimal unless $P=NP$, assuming the "unique games conjecture".

Correlation Clustering [2, chapter 6.4]

Semidefinite programming is useful for constraint satisfaction problems and graph partitioning problems.

Let us see one more example.

In the correlation clustering problem, we are given an undirected graph, and each edge ij has two nonnegative weights, $w_{ij}^+ \geq 0$ and $w_{ij}^- \geq 0$.

The goal is to cluster the vertices into sets of similar vertices; the degree to which i and j are similar are given by w_{ij}^+ , and the degree to which they are different are given by w_{ij}^- .

Let S be a partition of the vertices, $\delta(S)$ be the set of edges with endpoints in different sets

Let S be a partition of the vertices, $\delta(S)$ be the set of edges with endpoints in different sets of the partition, and $E(S)$ be the set of edges with endpoints in the same set of the partition.

The objective is to maximize $\sum_{ij \in E(S)} w_{ij}^+ + \sum_{ij \in \delta(S)} w_{ij}^-$.

Let e_k be the k -th unit vector. Let x_i be a vector for vertex i .

One can model the problem as follows.

$$\max \sum_{ij \in E} (w_{ij}^+ (x_i \cdot x_j) + w_{ij}^- (1 - x_i \cdot x_j))$$

subject to $x_i \in \{e_1, \dots, e_n\}$ for all i .

We can relax the problem by a semidefinite program.

$$\max \sum_{ij \in E} (w_{ij}^+ (v_i \cdot v_j) + w_{ij}^- (1 - v_i \cdot v_j))$$

subject to $v_i \cdot v_i = 1 \quad \forall i$

$$v_i \cdot v_j \geq 0 \quad \forall i, j$$

$$v_i \in \mathbb{R}^n \quad \forall i$$

This semidefinite program can be solved in polynomial time.

Let OPT_{SDP} be its objective value.

The algorithm is simple. It uses two random hyperplanes to partition the vertices into 4 sets.

$$R_1 = \{i \in V : r_1 \cdot v_i \geq 0, r_2 \cdot v_i \geq 0\}$$

$$R_2 = \{i \in V : r_1 \cdot v_i \geq 0, r_2 \cdot v_i < 0\}$$

$$R_3 = \{i \in V : r_1 \cdot v_i < 0, r_2 \cdot v_i \geq 0\}$$

$$R_4 = \{i \in V : r_1 \cdot v_i < 0, r_2 \cdot v_i < 0\}.$$

We will show that it is a $3/4$ -approximation.

Let W be the weight of the random partition.

$$\begin{aligned} \text{Then } E[W] &= \sum_{ij \in E} (w_{ij}^+ \Pr(i \& j \text{ in same set}) + w_{ij}^- \Pr(i \& j \text{ in different sets})) \\ &= \sum_{ij \in E} \left(w_{ij}^+ \left(1 - \frac{\theta_{ij}^2}{\pi}\right) + w_{ij}^- \left(1 - \left(1 - \frac{\theta_{ij}^2}{\pi}\right)\right) \right) \\ &\geq \frac{3}{4} \sum_{ij \in E} (w_{ij}^+ \cos \theta_{ij} + w_{ij}^- (1 - \cos \theta_{ij})) \\ &= \frac{3}{4} \sum_{ij \in E} (w_{ij}^+ (v_i \cdot v_j) + w_{ij}^- (1 - v_i \cdot v_j)) \\ &= \text{OPT}_{\text{SDP}}, \end{aligned}$$

where the second line is true because i and j are in the same set if both hyperplanes do not separate them, and the inequality follows from some calculus.

References

- [1] Vazirani. Approximation algorithms.
- [2] Williamson, Shmoys. The design of approximation algorithms.
- [3] Dubhashi, Panconesi. Concentration of measure for the analysis of randomized algorithms.