CSE 521: Design and Analysis of Algorithms I **Fall 2020**

Lecture 19: Linear Programming Relaxation, Duality and Applications

Lecturer: Shayan Oveis Gharan *12/09/20*

1.1 Optimization

Consider an optimization problem where we are trying to find a solution with minimum cost among a set of feasible solutions. We say an algorithm, ALG, gives an α-approximation for the problem if for any possible input to the problem, we have

$$\frac{\text{cost(ALG)}}{\text{cost(OPT)}} \leq \alpha \tag{1.1}$$

Here, OPT denotes the optimum solution to the problem.

To prove that a given algorithm is an α-approximation, it is sufficient to find a lower-bound for cost(OPT), and then prove that the ratio between cost(ALG) and this lower-bound for any input is upper-bounded by α.

1.1.1 Example: Vertex Cover

Here, we give an application of linear programming in designing an approximation algorithm for a graph problem called vertex cover. We will design a 2-approximation algorithm. This is the best known result for the vertex cover problem. It is a fundamental open problem to beat the factor 2 approximation for the vertex cover problem. In the next lecture we will discuss a generalization of vertex cover called the set cover problem and we see some applications.

Given a graph $G = (V, E)$, we want to find a set $S \subset V$ such that every edge in E is incident to at least one vertex in S. Obviously, we can let $S = V$. But, here among all such sets S we want to choose a one of minimal cost, where cost(S) is defined as $\sum_{i \in S} c_i$ if every vertex i has associated cost c_i, and $|S|$ if vertices do not have any cost.

In the first step we write a (integer) program which characterizes the optimum solution. Then, we use this program to give a lower bound on the optimum solution. We define this problem with a set of variables $x_i \; \forall\, i \in V$, where x_i is defined as

$$x_i = \begin{cases} 1 & i \in S \\ 0 & i \notin S \end{cases} \tag{1.2}$$

Our constraint that every edge must be incident to at least one vertex in S can be written as $x_i + x_j \geq 1 \; \forall\, i \sim j \in E$. So, the question is to find values for all x_i's that minimize the cost of the set S subject to the aforementioned constraint. This can be defined as the following optimization problem

$$\begin{aligned} \min \quad & \sum_{i \in V} c_i x_i \\ \text{s.t.,} \quad & x_i + x_j \geq 1, \; \forall\, i \sim j \in E \\ & x_i \in \{0, 1\}, \; \forall\, i \in V \end{aligned} \tag{1.3}$$

Observe that the optimum solution of the above program is exactly equal to the optimum set cover. Note that this is not a linear program, since we have that $x_i \in \{0, 1\}$ for every vertex i, rather than allowing x_i to be a continuous-valued variable. Since the vertex cover problem is NP-hard in general, we do not expect to ever find a general solver to efficiently solve the above integer program. However, there are commercial integer programming solver that work great in practice. They solve a set of linear inequality subject to the each of the underlying variables being 0/1. For many practical applications these program actually find the optimum solution very fast. So, one should always keep them in mind if we are trying to solve an optimization problem in practice.

We can relax the above (integer) program by replacing the integer constraint with the constraint that $0 \leq x_i \leq 1 \; \forall i \in V$. This turns the problem into a linear program. Since this is optimizing over a set of x_i's that includes the optimum set cover, the optimal value of this linear program will be less than or equal to the optimal value of the set cover problem, i.e. OPT LP $\leq$ OPT. The resulting linear program can be written as

$$\begin{aligned} \min \quad & \sum_{i \in V} c_i x_i \\ \text{s.t.,} \quad & x_i + x_j \geq 1, \; \forall i \sim j \in E \\ & 0 \leq x_i \leq 1, \; \forall i \in V \end{aligned} \tag{1.4}$$

Suppose we have an optimal solution of the above program. We want to round this solution into a set cover such that the cost of the cover that we produce is within a small factor of the cost of the LP solution.

The idea is to ue a simple thresholding idea: For each vertex i, if $x_i \geq 0.5$, then we add i to S, otherwise we don't include i in S.

Claim 1.1. *For any solution x of linear program (1.4), the resulting set S, is a vertex cover*

Proof. For a feasible solution x to the linear program, we know that $x_i + x_j \geq 1 \; \forall i \sim j \in E$. This means that for every edge $i \sim j$, at least one of x_i, x_j is at least 0.5.Therefore, for any edge $i \sim j$ at least one of i, j is in S. So, S is a vertex cover. □

Claim 1.2. *For any solution x of linear program (1.4) the resulting set S satisfies*

$$\sum_{i \in S} c_i \leq 2 \sum_i c_i x_i = OPT\ LP.$$

This implies that the above algorithm is a 2 approximation for the vertex cover problem.

Proof.

$$\sum_{i \in S} c_i = \sum_{i: x_i \geq 0.5} c_i \leq \sum_{i: x_i \geq 0.5} 2 c_i x_i \leq \sum_i c_i x_i.$$

□

Note that in the worst case $x_i = 0.5$ for all vertices i and the above claim is tight.

1.1.2 Set Cover

Given a set of n elements $V = \{1, 2, ..., n\}$ and a collection of n sets $\{S_1, S_2, ..., S_n\}$ whose union equals the ground set V, the set cover problem is to choose a set $T \subseteq [n]$ with a minimum cost and subject to a constraint that $T \cap S_i \neq \phi, \forall i$. The problem is formulated as (1.5).

$$\begin{aligned} \min \quad & \sum_i x_i c_i \\ \text{s.t.,} \quad & \sum_{i: i \in S_j} x_i \geq 1, \ \forall j. \\ & x_i \in \{0, 1\} \end{aligned} \tag{1.5}$$

Since the problem (1.5) is an NP-hard problem, it can be relaxed via the Linear Programming, where the constraint $x_i \in \{0, 1\}$ is relaxed to $x \geq 0$, to find an optimal point x_{lp}^* such that the optimal value corresponding to x_{lp}^* is a lower bound to the the original problem. Next, a randomized rounding is used, that is

$$Y_i = \begin{cases} 1, & \text{w.p. } \alpha x_i \\ 0, & \text{otherwise} \end{cases} \tag{1.6}$$

The analysis of the randomized rounding

$$\begin{aligned} \mathbb{P}\left[\sum_{i \in S_j} Y_i = 0\right] &= \mathbb{P}\left[Y_i = 0, \forall i \in S_j\right] \\ &= \prod_{i \in S_j} \mathbb{P}\left[Y_i = 0\right] \\ &= \prod_{i \in S_j} (1 - \alpha x_j) \\ &\leq \prod_{i \in S_j} e^{-\alpha x_i} \\ &\leq e^{-\sum_{i \in S_j} \alpha x_i} \leq e^{-\alpha} \end{aligned}$$

If we choose $\alpha = \log 2m$, we have $\mathbb{P}\left[\sum_{i \in S_j} Y_i = 0\right] \leq \frac{1}{2m}$. So, $\mathbb{P}\left[\sum_{i \in S_j} Y_i \geq 0\right] \geq 1 - \frac{1}{2m}$, which means with union bound in every set w.p. $\frac{1}{2}$, we have a probability 1. Furthermore, by the Markov inequality,

$$\mathbb{E}\left[\sum_i c_i Y_i\right] = \alpha \sum_i x_i c_i \leq 2\alpha \cdot \text{OPT LP} \leq 2\alpha \cdot \text{OPT} \tag{1.7}$$

1.2 Spectral Sparsifiers

For two symmetric matrix $A, B \in \mathbb{R}^{n \times n}$ we write

$$A \preceq B$$

iff $B - A \succeq 0$, i.e., $B - A$ is a PSD matrix. In other words, $A \preceq B$ iff for any vector $x \in \mathbb{R}^n$,

$$x^T Ax \leq x^T Bx$$

Let $\lambda_1 \leq \cdots \leq \lambda_n$ be the eigenvalues of A and $\tilde{\lambda}_1 \leq \cdots \leq \tilde{\lambda}_n$ be the eigenvalues of B. It follows that if $A \preceq B$, then for all i, $\lambda_i \leq \tilde{\lambda}_i$.

Definition 1.3. *Given a graph $G = (V, E)$ and $\epsilon > 0$, we say a (weighted) graph $H = (V, E')$ is a $1 \pm \epsilon$-spectral sparisifier of G if*

$$(1 - \epsilon)L_G \preceq L_H \preceq (1 + \epsilon)L_G.$$

Ideally, we want H to be a subgraph of G which has much fewer edges than G. An immediate consequence of the above definition is that all eigenvalues of H approximate eigenvalues of H up to multiplicative $1 \pm \epsilon$ error.

It is also not hard to see that if H is a $1 \pm \epsilon$-spectral sparisifer of G then it preserves the size of all cuts of G. In particular, for a set $S \subseteq V$, recall $\mathbf{1}^S$ is the indicator vector of the set S. It follows that for a graph G,

$$\mathbf{1}^S L_G \mathbf{1}^S = \sum_{i \sim j} (\mathbf{1}^S_i - \mathbf{1}^S_j)^2 = \sum_{i \sim j} \mathbb{I}\left[|\{i, j\} \cap S| = 1\right] = 2|E(S, \overline{S})|$$

So, if H is a $1 \pm \epsilon$-spectral sparsifier of G we have

$$(1 - \epsilon)\mathbf{1}^S L_G \mathbf{1}^S \leq \mathbf{1}^S L_H \mathbf{1}^S \leq (1 + \epsilon)\mathbf{1}^S L_G \mathbf{1}^S,$$

so the (weighted) size of every cut in H is within $1 \pm \epsilon$ multiplicative factor of the same cut in G.

Theorem 1.4 (Speilman-Srivastava)**.** *For every graph $G = (V, E)$ and $\epsilon > 0$, there is a weighted graph H that is a subgraph of G such that H is a $1 \pm \epsilon$-spectral sparsifier of G and that H has at most $O(n \log n/\epsilon^2)$ many edges.*

The first idea that come to mind is to construct an unbiased estimator: Let X be a random matrix defined as follows: For every edge $e \in E$, $X = L_e/p_e$ with probability p_e, Then, observe that

$$\mathbb{E}[X] = \sum_e p_e \frac{L_e}{p_e} = \sum_e L_e = L_G.$$

So, X is an unbiased estimator. And, the main question is how to choose the probabilities such that concentration bounds can kick in and imply $X \approx \mathbb{E}[X]$.

Let us start with a simple case of a complete graph. If G is a complete graph, we can simply let $p_e = 1/\binom{n}{2}$ for all edges. It then follows that $O(n \log n/\epsilon^2)$ many samples are enough to approximate the complete graph. However, it turns out that a uniform distribution does not necessarily work out in a general graph. For example, if G is a *Barbell graph*, i.e., union of two K_n connected by an edge (see Figure 1.1), then, if we want to down-size G to $O(n \log n)$ edges we need to let $p_e = O(\log n)/n$ for all edges, but then the single edge connecting the two complete graphs won't be chosen with high probability. So, H is disconnected with high probability and it cannot be a spectral sparsifier of G for any $\epsilon < 1$. In the rest of this section we will see how to choose the edge probabilities p_e.

1.2.1 Reduction to Isotropic Case

First, it turns out that we can reduce the graph sparsification problem to a linear algebraic problem. First, let us recall the generalized eigenvalue problem. In the generalized eigenvalue problem we are given a symmetric matrix A and a PSD matrix B and we want to find

$$\max_x \frac{x^T Ax}{x^T Bx}$$

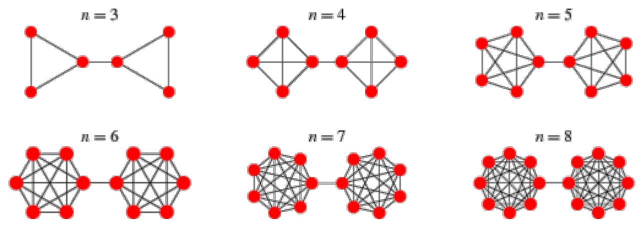

Figure 1.1: Barbell Graph

In the special case that B is the identity matrix, the solution of the above problem is exactly the largest eigenvector of A. We can solve the above problem by reducing it to an eigenvalue problem.

$$\max_x \frac{x^TAx}{x^TBx} = \max_x \frac{x^TB^{1/2}B^{-1/2}AB^{-1/2}B^{1/2}x}{x^TB^{1/2}B^{1/2}x} = \max_{x:y=B^{1/2}x} \frac{y^TB^{-1/2}AB^{-1/2}y}{y^Ty} = \max_y \frac{y^TB^{-1/2}AB^{-1/2}y}{y^Ty}$$

So, to find the solution to the generalized eigenvalue problem it is enough to find the largest eigenvector y of the matrix $B^{-1/2}AB^{-1/2}$ and then let $x = B^{-1/2}y$. Note that, here we are using the fact that B is PSD; otherwise $B^{-1/2}$ is not well defined.

Now, let us go back to the spectral sparsifier problem. Suppose H is a $1 \pm \epsilon$-spectral sparsifier of G. It follows that for all $x \in \mathbb{R}^n$.

$$1 - \epsilon \leq \frac{x^TL_Hx}{x^TL_Gx} \leq 1 + \epsilon$$

By a similar analogy, it follows that for all y,

$$1 - \epsilon \leq \frac{y^TL_G^{-1/2}L_HL_G^{-1/2}y}{y^Ty} \leq 1 + \epsilon$$

So, the above inequality implies that the matrix $L_G^{-1/2}L_HL_G^{-1/2}$ is approximately equal to the identity matrix.

Remark 1.5. *There is a technical problem here: since L_G has a zero eigenvalue the inverse of L_G is not well-defined. In the above calculation, we take the inverse with respect to positive eigenvalues of G; in particular if $L_G = \sum_i \lambda_i v_i v_i^T$, we let $L_G^{-1/2} = \sum_{i:\lambda_i>0} \frac{1}{\sqrt{\lambda_i}} v_i v_i^T$. We ignore this fact in the rest of our calculations for the simplicity of the argument.*

Now, we reformulate the spectral sparsification problem as follows:

Theorem 1.6. *Given $n \times n$ PSD matrices, $E_1, \ldots, E_m$ such that*

$$\sum_{i=1}^m E_i = I,$$

For any $\epsilon > 0$, there is a subset S of them of size $O(n \log n/\epsilon^2)$ and a set of weights w_i for each $i \in S$ such that

$$(1-\epsilon)I \preceq \sum_{i\in S} w_iE_i \preceq (1+\epsilon)I$$

Let us discuss how we can reduce the sparsification problem to the above theorem. Say our graph G has m edges. For edge e_i define

$$E_i = L_G^{-1/2}L_{e_i}L_G^{-1/2}.$$

First, observe that each E_i is a PSD matrix, and furthermore,

$$\sum_{i=1}^{m} E_i = \sum_{i=1}^{m} L_G^{-1/2} L_{e_i} L_G^{-1/2} = L_G^{-1/2} \left(\sum_{i=1}^{m} L_{e_i} \right) L_G^{-1/2} = L_G^{-1/2} L_G L_G^{-1/2} = I.$$

So, roughly speaking by multiplying the Laplacians of the edges of G by $L_G^{-1/2}$ on both sides we are normalizing the space such that every direction look the same. We are reducing the graph spectral sparsification problem to a linear algebraic problem of finding a sparsifier of the sum of PSD matrices that add up to the identity matrix.

1.2.2 Finding the Spectral Sparsifier

Now, as before, let

$$X = \frac{E_i}{p_i}$$

with probability p_i. Similar to before, $\mathbb{E}[X] = I$. So, X is an unbiased estimator. To prove the concentration we used the following generalization of the Chernoff bound which is known as matrix Chernoff bound

Theorem 1.7. *Let X be a random $n \times n$ PSD matrix. Suppose that $X \preceq \alpha \mathbb{E}[X]$ with probability 1. Let $X_1, \dots, X_k$ be independent copies of X, then for any $\epsilon > 0$,*

$$\mathbb{P}\left[(1-\epsilon)\mathbb{E}[X] \preceq \frac{1}{k}(X_1 + \cdots + X_k) \preceq (1+\epsilon)\right] \geq 1 - 2ne^{-\epsilon^2 k/4\alpha}.$$

So, this says that to prove Theorem 1.6 it is enough to choose $k = O(\alpha \log n/\epsilon^2)$ many copies of X. Finally, to finish the proof we need to choose the probabilities p_i such that $\alpha \leq O(n)$.

First, suppose we let p_i be uniform, i.e., $p_i = 1/m$ for all i. Then, we need to choose α such that for all i,

$$\frac{E_i}{1/m} \preceq \alpha I.$$

But it turns out that in the worst case we have to let $\alpha = m$.

The idea is to let $p_i \propto \mathrm{Tr}(E_i)$. Let us first find the normalizing constant: Suppose $p_i = \beta \, \mathrm{Tr}(E_i)$. Then,

$$\sum_i p_i = \beta \sum_i \mathrm{Tr}(E_i) = \beta \, \mathrm{Tr}\left(\sum_i E_i\right) = \beta n$$

So, we should let $\beta = 1/n$. It follows that $p_i = \beta \, \mathrm{Tr}(E_i) = \mathrm{Tr}(E_i)/n$.

Now, we claim that for all i,

$$\frac{E_i}{\mathrm{Tr}(E_i)/n} \preceq \alpha I$$

for $\alpha = n$. This will complete the proof of Theorem 1.6. To show the above it is enough to show

$$\frac{E_i}{\mathrm{Tr}(E_i)} \preceq I$$

To show this we only use the fact that all eigenvalues of E_i are in the range $[0, 1]$ (this is true because E_i is PSD, and $\sum_j E_j = I$). So, it remains to prove the above inequality. Say $E_i = \sum_j \lambda_j v_j v_j^T$. For any arbitrary vector $x \in \mathbb{R}^n$,

$$x^T \frac{E_i}{\mathrm{Tr}(E_i)} x = \frac{\sum_j \lambda_j \langle x, v_j \rangle^2}{\sum_j \lambda_j} \leq \max_j \langle x, v_j \rangle^2 \leq \|x\|^2 = x^T I x.$$

1.2.3 Back to Spectral Sparsifiation

In the previous section we saw that we should choose each E_i with probability $\text{Tr}(E_i)/n$. Translating this back to the setting of graph sparsification; recall that for edge e_i, $E_i = L_G^{-1/2} L_{e_i} L_G^{-1/2}$. So, we should sample every edge e of G with probability

$$p_e = \frac{\text{Tr}(L_G^{-1/2} L_e L_G^{-1/2})}{n}$$

The quantity

$$\text{Tr}(L_G^{-1/2} L_e L_G^{-1/2}) = b_e^T L_G^{-1} b_e$$

is called the *effective resistance* of the edge e; here for an edge $e = \{u, v\}$, $b_e = \mathbf{1}_u - \mathbf{1}_v$ is the vector which is $+1$ at one endpoint of e and -1 at the other endpoint and 0 everywhere else. It is very well understood and there are fast algorithms to estimate it; one can also compute the inverse of the Laplacian and compute the effective resistance of all edges immediately.

The following simple algorithm can be used to construct a $1 \pm \epsilon$-spectral sparsifier of G:

1. For $i = 1$ to $O(n \log n/\epsilon^2)$
2. Sample each edge e of G with probability $p_e = \text{Tr}(L_G^{-1/2} L_e L_G^{-1/2})/n$. If the edge e is sampled weight it by $1/p_e$.