CSE 521: Design and Analysis of Algorithms I **Fall 2025**

Lecture 16: Introduction to Linear Programming

Lecturer: Shayan Oveis Gharan *11/19/25*

Disclaimer: *These notes have not been subjected to the usual scrutiny reserved for formal publications.*

16.1 Expander Graphs

Let $G = (V, E)$ be a d-regular graph. Recall that G is d regular if the degree of all vertices of G is exactly equal to d. Let $\tilde{L} = L/d$ be the normalized Laplacian matrix of G with eigenvalues $0 = \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$.

We say G is a (weak) expander if $\lambda_2 \geq \Omega(1)$, i.e., λ_2 is bounded away from 0; it is an absolute constant independent of the size of G. In general, this definition can be a bit confusing because for a fixed graph the number of vertices is a constant and λ_2 will also be a fixed constant. So, a more rigorous definition is the following: An infinite sequence $G_1, G_2, \ldots,$ of d-regular graphs where the size goes to infinity is a sequence of expander graphs if there exists $c > 0$ such that for all i the second smallest eigenvalue of the normalized Laplacian of G_i is at least c.

Expander graph can be seen as sparse version of the complete graph. Let $G = (V, E)$ be an expander graph (with n vertices). Then it has the following properties:

i) The diameter of an expander graph is $O(\log n)$. In particular, for any pair of vertices i, j of G, the shortest path connecting i to j has length at most $O(\log n)$.

ii) $\phi(G) \geq \Omega(1)$. Recall that by Cheeger's inequality $\phi(G) \geq \lambda_2/2$. So, since $\lambda_2 \geq \Omega(1)$, $\phi(G) \geq \Omega(1)$. Therefore, for any set S such that $\text{vol}(S) \leq \text{vol}(V)/2$, $\phi(S) \geq \Omega(1)$. In other words, a constant fraction of edges incident to vertices of S leave the set S. That in a sense means that G is unclusterable. Note that there are expander graphs with $d = 3$. These graphs are locally very sparse; every vertex has only 3 neighbors. Nonetheless, globally, there are very connected. If we want to break the network into two sets of almost equal size we have to cut a constant fraction of edges of G.

iii) Random walks mix rapidly in expander graphs. A simple random walk on an (undirected) graph G is defined as follows: Given a vertex v move to a uniformly random neighbor of v. It follows that if G is an expander, then a random walk started at an arbitrary vertex of G will be at an (almost) uniformly random vertex of G at time $O(\log n)$. In other words, the walk completely forgets where it has started from.

For the sake of comparison note that a cycle of length n has none of the above properties. It has diameter n and the random walk started at vertex 0 takes time $O(n^2)$ to be at a uniformly random vertex of it.

We say that a graph G is a strong expander if $\lambda_2 \geq 1 - O(1)/\sqrt{d}$. The best bound that we can hope for is

$$\lambda_2 \geq 1 - \frac{2\sqrt{d-1}}{d}.$$

These family of graphs are called *Ramanujan* and they are the best expanders that we can hope for. It turns out that these graphs are not very rare, as a random d regular is almost Ramanujan with high probability, this is a result due to Friedman.

Expander mixing lemma shows that strong expanders are very similar to d regular graphs, in the following sense: Let G be a strong d-regular strong expander and let H be a graph where for every pair i, j of vertices there is an edge connecting i to j with probability d/n. Then, for any two disjoint sets S, T of vertices of size $|S|, |T| \geq \Omega(n)$, the number of edges between S, T in G and H are almost the same. See Problem 2 of PS4 for the exact statement.

Another important fact about expander graphs is that many NP-hard optimization problems become easier on expander graphs in the sense that we can design algorithms with significantly better approximation factors if the underlying graph is an expander. We see one such example in Problem 2 of PS4. But the general fact follows from the expander mixing lemma: Expander graphs are similar to random graphs and complete graphs. Many problems such as max cut, min conductance cut, etc. are easier on random graphs and complete graphs and henceforth they are easier on expander graphs. For a concrete example, note that since $\lambda_2 \geq \Omega(1)$ in an expander graphs, and by Cheeger's inequality, $\lambda_2/2 \leq \phi(G) \leq \sqrt{2\lambda_2}$, we can obtain a constant factor approximation algorithm for $\phi(G)$ if G is an expander simply by outputing λ_2 as an estimate of $\phi(G)$. This is despite the fact the best known approximation algorithm for $\phi(G)$ in general graph is an $O(\sqrt{\log n})$ approximation algorithm of Arora, Rao and Vazirani [ARV09].

16.2 Linear Programming

Linear systems are easiest class of optimization problems. Roughly speaking, any linear system of equations can be solved efficiently. The easiest examples are linear systems of equalities. This systems can be solved by matrix inversion. Given a (full rank) matrix A, to solve $Ax = b$, it is enough to compute $A^{-1}b$. In linear programming we study system of linear inequalities.

In the first few lectures we study linear systems of inequalities, also known as Linear Programming (LP). We also allow for a linear cost (or objective) function. LPs can be used to solve problems in a wide range of disciplines from engineering to nutrition and the stock market, At some point it was estimated that half of all computational tasks in the world correspond to solving linear programs.

A general LP can be formalized as follows:

$$\begin{aligned} &\min_x \langle c, x \rangle \\ &\text{s.t. } Ax \leq b, \end{aligned} \tag{16.1}$$

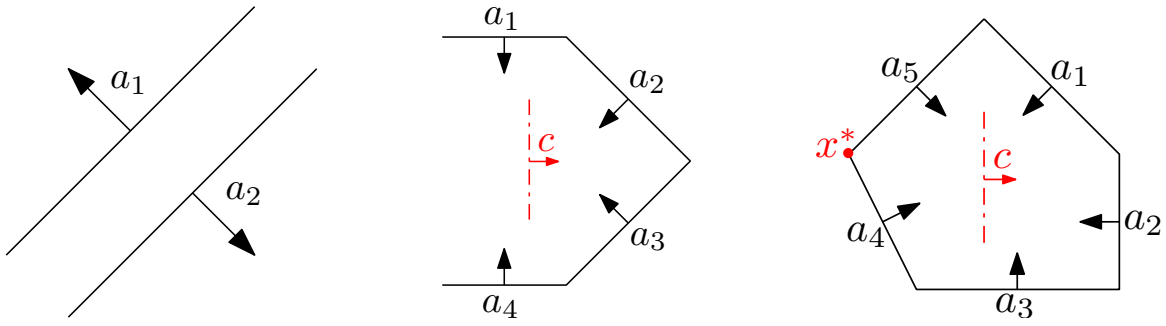
where $x \in \mathbb{R}^n$ is represents the variables, $c \in \mathbb{R}^n$ defines the objective function, and $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$ define the constraints. The above form is fairly general; one can model various types of constraints in this form. For example, a constraint $\langle a_1, x \rangle \geq b_1$ can be written as $\langle -a_1, x \rangle \leq -b_1$. Or, a constraint $\langle a_1, x \rangle = b_1$ can be written as $\langle a_1, x \rangle \leq b_1$ and $\langle -a_1, x \rangle \leq -b_1$.

The objective function can be viewed as a hyperplane in $\mathbb{R}^n$ with normal vector c. Further, one can express the constraint matrix A as a series of row vectors:

$$A = \begin{bmatrix} a_1^{\mathrm{T}} \\ a_2^{\mathrm{T}} \\ \vdots \\ a_m^{\mathrm{T}} \end{bmatrix}.$$

When viewed from this perspective, it is easy to see the constraint set $a_i^{\mathrm{T}} x \leq b_i,\ i \in \{1, 2, \ldots, m\}$ as the intersection of finitely many half-spaces. The feasible set an LP is called a polyhedron. Thus, the solution will always be one of the following three cases shown in Figure 16.1.

There are many well known ways of solving LPs, but two of the most well known and used are Interior Point Methods (IPMs) and the simplex method. The simplex method is based on a very geometrically intuitive



(a) Infeasible domain of x: No solution.

(b) Unbounded domain of x: $x^* = \pm\infty$

(c) Closed, bounded and compact domain of x: at least one vertex of the domain of x is in the solution set x^*.

Figure 16.1: The possible domains of the solution variable x for an LP, along with the solution.

idea: start at a vertex of the domain of x and move to an adjacent vertex with lower objective - continue until an optimal solution is reached. Theoretically, the worst case time complexity of the simplex method is exponential; however, the simplex method is very fast in practice, and toolboxes such as CPLEX and Gurobi can solve very large LPs (millions of solution variables) in a few seconds. Interior point methods have better theoretical bounds; roughly speaking, using an interior point method, one can solve a general linear program with n variables by solving $\tilde{O}(\sqrt{n})$ many systems of linear equalities.

16.3 Convex Programming

A function $f : \mathbb{R}^n \to \mathbb{R}$ is convex on a set $S \subseteq \mathbb{R}^n$ if for any two points $x, y \in S$, we have

$$f\left(\frac{x+y}{2}\right) \leq \frac{1}{2}(f(x) + f(y)).$$

We say f is concave if for any such $x, y \in S$, we have

$$f\left(f(\alpha x + (1-\alpha)y\right) \geq \alpha f(x) + (1-\alpha)f(y),$$

for any $0 \leq \alpha \leq 1$. There is an equivalent definition of convexity: For a function $f : \mathbb{R}^n \to \mathbb{R}$, the Hessian of f, $\nabla^2 f$ is a $n \times n$ matrix defined as follows:

$$(\nabla^2 f)_{i,j} = \partial_{x_i}\partial_{x_j} f$$

for all $1 \leq i, j \leq n$. We can show that f is convex over S if and only if for all $a \in S$,

$$\nabla^2 f\Big|_{x=a} \succeq 0.$$

For example, consider the function $f(x) = x^T A x$ for $x \in \mathbb{R}^n$ and $A \in \mathbb{R}^{n\times n}$. Then, $\nabla^2 f = A$. So, f is convex (over $\mathbb{R}^n$) if and only if $A \succeq 0$.

For another example, let $f : \mathbb{R} \to \mathbb{R}$ be $f(x) = x^k$ for some integer $k \geq 2$. Then, $f''(x) = k(k-1)x^{k-2}$. If k is an even integer, $f''(x) \geq 0$ over all $x \in \mathbb{R}$, so f is convex over all real numbers. On the other hand, if k is an odd integer then $f''(x) \geq 0$ if and only if $x \geq 0$. So, in this f is convex only over non-negative reals.

Similarly, f is concave over S, if $\nabla^2 f\Big|_{x=a} \preceq 0$ for all $a \in S$. For example, $x \mapsto \log x$ is concave over all positive reals.

We can generalize linear programming to convex programming as follows:

$$\begin{aligned} \min & f(x) \\ \text{s.t.}, \quad & f_1(x) \le b_1 \\ & \vdots \\ & f_m(x) \le b_m \end{aligned}$$

where $f, f_1, \ldots, f_m$ are convex functions over the domain of x. Note that if f_i's are convex only over a convex set S of $\mathbb{R}^n$, we can limit x to bein

Convex set We say a set $S \subseteq \mathbb{R}^n$ is convex if for any pair of points $x, y \in S$, the line segment connecting x to y is in S.

For example, let $f : \mathbb{R}^n \to \mathbb{R}$ be a convex function over a set $S \subseteq \mathbb{R}^n$. Let $t \in \mathbb{R}$, and define

$$T = \{x \in \mathbb{R}^n : f(x) \le t\}.$$

Then, T is convex. This is because if $x, y \in T$, then for any $0 \le \alpha \le 1$,

$$f(\alpha x + (1-\alpha)y) \le \alpha f(x) + (1-\alpha) f(y) \le \alpha t + (1-\alpha)t = t$$

where the first inequality follows by convexity of f. So, $\alpha x + (1-\alpha) \in T$ and T is convex.

Norms are Convex functions A norm $\|\cdot\|$ is defined as a function that maps $\mathbb{R}^n$ to $\mathbb{R}$ and satisfies the following three properties,

i) $\|x\| \ge 0$ for all $x \in \mathbb{R}^n$,

ii) $\|\alpha x\| = \alpha \|x\|$ for all $\alpha \ge 0$ and $x \in \mathbb{R}^n$,

iii) Triangle inequality: $\|x + y\| \le \|x\| + \|y\|$ for all $x, y \in \mathbb{R}^n$.

It is easy to see that any norm function is a convex function: This is because for any $x, y \in R^n$, and $0 \le \alpha \le 1$,

$$\|\alpha x + (1-\alpha)y\| \le \|\alpha x\| + \|(1-\alpha)y\| = \alpha \|x\| + (1-\alpha) \|y\| .$$

16.4 Semidefinite Programming

References

[ARV09] S. Arora, S. Rao, and U. Vazirani. "Expander Flows, Geometric Embeddings and Graph Partitioning". In: *J. ACM* 56.2 (Apr. 2009), 5:1–5:37 (cit. on p. 16-2).