Computer Science 2: Advanced Algorithms Fall 2026

Lecture 9 — September 21, 2026

Prof. Jarosław Błasiok *Scribe: Lisa Sophie Albertelli, Suhanee Korpe*

In the previous lectures:

Linear programming and relaxation. A linear program optimizes a linear function over a polyhedron:

$$\max_x \langle c, x\rangle \qquad \text{subject to } Ax \le b.$$

It can be solved in polynomial time. Our strategy was to relax a combinatorial problem to a larger feasible set, solve the resulting LP, and then round its solution back to a combinatorial solution.

The sparsest cut problem. For a graph $G = (V, E)$, we seek a nontrivial cut minimizing

$$\phi(S) = \frac{|E(S, \bar{S})|}{|S|\,|\bar{S}|}, \qquad \phi^* = \min_{\varnothing \neq S \subsetneq V} \phi(S).$$

A cut is represented by the cut metric $d_S(u, v) = |\mathbf{1}_S(u) - \mathbf{1}_S(v)|$, which equals one across the cut and zero within either side. Thus,

$$\phi(S) = \frac{\sum_{\{u,v\}\in E} d_S(u, v)}{\sum_{u<v} d_S(u, v)}.$$

Relaxing cut metrics to arbitrary metrics and normalizing the denominator gives the LP

$$\begin{aligned} \lambda^* = \min_d \quad & \sum_{\{u,v\}\in E} d_{uv} \\ \text{subject to} \quad & \sum_{u<v} d_{uv} = 1, \\ & d_{uv} \ge 0, \\ & d_{uw} \le d_{uv} + d_{vw} \qquad \text{for all } u, v, w. \end{aligned}$$

Since this enlarges the feasible set, $\lambda^* \le \phi^*$.

Rounding through ℓ_1 metrics. The rounding uses two ingredients:

- **Bourgain's theorem:** every n-point metric embeds into ℓ_1 with distortion $\eta = O(\log n)$:

$$\frac{d(x, y)}{\eta} \le \|f(x) - f(y)\|_1 \le d(x, y).$$

  Distances never increase and shrink by at most a factor η.

- **Lemma A:** every ℓ_1 metric is a nonnegative combination of cut metrics:

$$d' = \sum_k \alpha_k d_{S_k}, \qquad \alpha_k \geq 0.$$

At least one of these cuts has sparsity no larger than the ratio associated with d'.

Embedding the LP metric into ℓ_1 and then extracting a cut therefore gives

$$\phi(S) \leq \eta\lambda^* \leq \eta\phi^*,$$

an $O(\log n)$-approximation.

Bourgain's embedding: the main idea. We first construct a random map into the real line using distance to a random set:

$$\pi_S(x) = d(x, S) = \min_{s \in S} d(x, s).$$

For nonempty S, the triangle inequality gives

$$|\pi_S(x) - \pi_S(y)| \leq d(x, y).$$

Choose a scale k uniformly from $\{1, \ldots, m\}$, where $m = \lfloor \log_2 n \rfloor + 2$, and include each point in S independently with probability 2^{-k}. The random-scale lemma shows that

$$\mathbb{E}|\pi_S(x) - \pi_S(y)| \geq \frac{c_0}{2m} d(x, y), \qquad c_0 = \frac{1}{4}(1 - e^{-1/4}).$$

The proof uses growing balls around x and y: at each relevant scale, a constant-probability event makes the sample hit one ball and miss the other. Summing the resulting radius increments gives the lower bound.

Finally, sample independent maps and place them in separate coordinates:

$$f(x) = \frac{1}{T}(\pi_1(x), \ldots, \pi_T(x)), \qquad \|f(x) - f(y)\|_1 = \frac{1}{T} \sum_{t=1}^{T} |\pi_t(x) - \pi_t(y)|.$$

Concentration ensures that these averages approximate their expectations simultaneously for all pairs, giving the required ℓ_1 embedding with distortion $O(\log n)$.

In this lecture:

From linear to semidefinite programming

Linear programming

A linear program minimizes a linear function over a feasible set defined by linear constraints:

$$\begin{aligned} \text{minimize} \quad & \langle c, x \rangle \\ \text{subject to} \quad & \langle a_i, x \rangle \geq b_i \qquad \text{for all } i. \end{aligned}$$

Here, $x \in \mathbb{R}^d$ is the vector of variables, $c, a_i \in \mathbb{R}^d$ are given coefficient vectors, and $b_i \in \mathbb{R}$ are given bounds. The objective is

$$\langle c, x \rangle = \sum_{j=1}^{d} c_j x_j.$$

The feasible set is a *polyhedron*: the intersection of the half-spaces specified by the linear inequalities. A bounded polyhedron is called a *polytope*. Linear programs can be solved in polynomial time.

Semidefinite programming: the same, with matrices

In a semidefinite program (SDP), the variable is a symmetric matrix $X \in \mathbb{R}^{d \times d}$ rather than a vector. The objective and the ordinary constraints remain linear in its entries, but we impose one additional requirement: X must be positive semidefinite, written $X \succeq 0$, (we will define it below).

$$\begin{aligned} \text{minimize} \quad & \langle C, X \rangle \\ \text{subject to} \quad & \langle A_i, X \rangle \geq b_i \qquad \text{for all } i, \\ & X \succeq 0. \end{aligned}$$

Here X, C, A_i are symmetric $d \times d$ matrices. Their matrix inner product is

$$\langle C, X \rangle = \sum_{j=1}^{d} \sum_{k=1}^{d} C_{jk} X_{jk} = \operatorname{tr}(C^\top X) = \operatorname{tr}(CX),$$

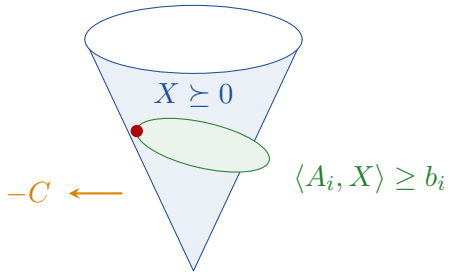
where the last equality uses symmetry. The trace, $\operatorname{tr}(M)$, is the sum of the diagonal entries of a square matrix M. Thus, the objective is simply a weighted sum of the entries of X; the same interpretation applies to each constraint $\langle A_i, X \rangle \geq b_i$.

Definition 1 (Positive semidefinite matrix)**.** A real symmetric matrix X is *positive semidefinite* (PSD), written $X \succeq 0$, if all its eigenvalues are nonnegative. Equivalently,

$$X \succeq 0 \quad \iff \quad v^\top X v \geq 0 \qquad \text{for every } v \in \mathbb{R}^d.$$

The notation $X \succeq 0$ does not mean that every entry of X is nonnegative. It is a condition on its eigenvalues, or equivalently on its quadratic forms.

Geometrically, the PSD matrices form a convex cone.



The feasible set of an SDP is the intersection of this cone with the polyhedron defined by the linear constraints. This intersection is convex but can have a curved boundary. Such a feasible set is called a *spectrahedron*.

There are several equivalent formulations of positive semidefiniteness. Many problems involving matrix optimization subject to spectral constraints can therefore be reformulated as SDPs.

Two facts we will use

Fact 2 (Gram matrices)**.** *A symmetric matrix X is positive semidefinite if and only if it is a Gram matrix:*

$$X \succeq 0 \quad \Longleftrightarrow \quad X = V^\top V \quad \Longleftrightarrow \quad X_{ij} = \langle v_i, v_j \rangle \quad \textit{for some vectors } v_1, \ldots, v_d.$$

By the spectral theorem, we can write

$$X = U^\top \Lambda U.$$

Since X is PSD, the eigenvalues in Λ are nonnegative. Thus,

$$X = U^\top \sqrt{\Lambda}\sqrt{\Lambda} U = (\sqrt{\Lambda}U)^\top (\sqrt{\Lambda}U) = V^\top V, \qquad V = \sqrt{\Lambda}U.$$

If v_i denotes the ith column of V, then the entries of X are exactly the inner products of these columns:

$$X_{ij} = \langle v_i, v_j \rangle.$$

Fact 3 (Efficiency)**.** *SDPs can be solved to accuracy ε in time polynomial in the input size and* $\log(1/\varepsilon)$*.*

The ellipsoid method allows us to handle an arbitrarily large family of linear constraints, provided we have an efficient *separation oracle*: given a candidate X, we must be able to find a violated constraint efficiently if X is infeasible.

We can express the PSD constraint as

$$X \succeq 0 \quad \Longleftrightarrow \quad \langle vv^\top, X \rangle \geq 0 \qquad \text{for every } v \in \mathbb{R}^d.$$

Using the cyclic property of the trace,

$$\langle vv^\top, X \rangle = \operatorname{tr}(vv^\top X) = \operatorname{tr}(v^\top X v) = v^\top X v.$$

Thus, the quadratic form $v^\top X v$ can be written as the matrix inner product of the outer product $vv^\top$ with X. There is no need to list all these constraints; what matters is finding a violated one efficiently. If $X \not\succeq 0$, an eigenvector corresponding to a negative eigenvalue supplies a violated constraint.

The Max Cut Problem

The Max-Cut problem

Given a graph $G = (V, E)$, we want to partition its vertices into two sets S and $\bar{S} = V \setminus S$ so that as many edges as possible have one endpoint in each set. Writing $E(S, \bar{S})$ for these crossing edges,

$$\mathrm{OPT} = \max_{S \subseteq V} |E(S, \bar{S})|.$$

Thus, OPT is the largest possible number of edges cut by a partition.

If a solution cut crosses every edge, the graph is bipartite. This perfect-solution case is easy to recognize: there are algorithms to check whether a graph is bipartite and find such a partition. Maximizing the number of cut edges in general is harder. A particularly interesting case is that of a *near-bipartite* graph, where some cut crosses almost all edges.

Max-Cut as a system of linear equations. Associate a variable $x_i \in \mathbb{F}_2 = \{0, 1\}$ with each vertex i. Its value indicates which side of the cut contains that vertex. For every edge $\{i, j\} \in E$, consider the equation

$$x_i \oplus x_j = 1,$$

This equation is satisfied exactly when x_i and x_j differ, meaning that the endpoints lie on opposite sides and the edge is cut. Max-Cut therefore asks us to satisfy as many of these equations as possible; every equation has exactly two variables.

Two approximation questions.

1. **Multiplicative approximation.** Given an optimum cut of size OPT, find a cut satisfying

   $$|E(S, \bar{S})| \geq \alpha \, \mathrm{OPT}\,.$$

   How large can α be? We want it as close to 1 as possible.

2. **Near-bipartite graphs.** Given the promise

   $$\mathrm{OPT} \geq (1 - \varepsilon)|E|,$$

   find a cut of size at least

   $$(1 - f(\varepsilon))|E|, \qquad f(\varepsilon) \longrightarrow 0 \quad \text{as } \varepsilon \longrightarrow 0.$$

   How quickly can $f(\varepsilon)$ approach zero?

A random cut already gives $\frac{1}{2}$

Put each vertex into S independently with probability $1/2$, and into $\bar{S}$ otherwise. Each edge is cut with probability $1/2$: whichever side contains its first endpoint, the second endpoint lies on the other side with probability $1/2$.

By linearity of expectation,

$$\mathbb{E}\big[|E(S, \bar{S})|\big] = \sum_{e \in E} \mathbb{P}[e \text{ is cut}] = \sum_{e \in E} \frac{1}{2} = \frac{|E|}{2}.$$

Since no cut can contain more than all the edges, $\mathrm{OPT} \leq |E|$. Therefore,

$$\mathbb{E}\big[|E(S, \bar{S})|\big] = \frac{|E|}{2} \geq \frac{\mathrm{OPT}}{2}.$$

A random cut thus gives a $1/2$-approximation in expectation, also described as an approximation within a factor of 2.

Derandomization: conditional expectations

We can find a cut with at least $|E|/2$ edges deterministically using the *method of conditional expectations*.

Fix an order $v_1, \ldots, v_n$ of the vertices and choose their sides one at a time. For the first vertex,

$$\mathbb{E}[|E(S,\bar{S})|] = \frac{1}{2}\,\mathbb{E}[|E(S,\bar{S})| \mid v_1 \in S] \quad + \frac{1}{2}\,\mathbb{E}[|E(S,\bar{S})| \mid v_1 \in \bar{S}].$$

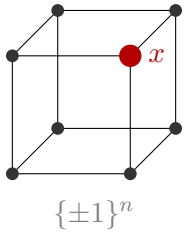
The expectation before fixing v_1 is the average of the expectations for its two possible placements. At least one of these two values is therefore as large as the current expectation. Place v_1 on the side with the larger value, leaving the remaining vertices unfixed.

Repeat this procedure for $v_2, v_3, \ldots, v_n$, keeping all earlier choices fixed. Each conditional expectation can be computed in $O(|E|)$ time. At every step, choose a side that does not decrease the expected cut size. Once all vertices have been assigned, the cut is fixed, and its size is at least the initial expectation:

$$|E(S,\bar{S})| \geq \frac{|E|}{2} \geq \frac{\mathrm{OPT}}{2}.$$

Max-Cut as a quadratic program

We can reformulate Max-Cut as the maximization of a quadratic function over the hypercube $\{-1,+1\}^n$.



Encode a cut $(S,\bar{S})$ by a vector $x \in \{-1,+1\}^n$, where $x_i = +1$ if $i \in S$ and $x_i = -1$ if $i \in \bar{S}$. For each edge $\{i,j\} \in E$,

$$\frac{1 - x_i x_j}{2} = \begin{cases} 1, & \text{if } i \text{ and } j \text{ lie on opposite sides of the cut,} \\ 0, & \text{if } i \text{ and } j \text{ lie on the same side.} \end{cases}$$

Indeed, opposite signs give $x_i x_j = -1$, so the expression equals $(1-(-1))/2 = 1$. Equal signs give $x_i x_j = 1$, so it equals $(1-1)/2 = 0$.

Summing over all edges therefore counts exactly the edges crossing the cut. Hence,

$$\mathrm{OPT} = \max_{x \in \{-1,+1\}^n} \sum_{\{i,j\} \in E} \frac{1 - x_i x_j}{2}.$$

Thus, Max-Cut is the maximization of a quadratic polynomial in the variables $x_1, \ldots, x_n$, constrained to the hypercube. This optimization problem is NP-hard.

The SDP relaxation

Recall the quadratic formulation of Max-Cut:

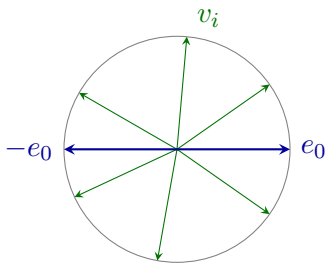
$$\mathrm{OPT} = \max_{x \in \{-1,+1\}^n} \sum_{\{i,j\} \in E} \frac{1 - x_i x_j}{2}.$$

Instead of assigning a sign $x_i \in \{-1, +1\}$ to each vertex, we assign a unit vector $v_i \in S^{n-1}$, where

$$S^{n-1} = \{v \in \mathbb{R}^n : \|v\|_2 = 1\}$$

is the unit sphere. We replace the product $x_i x_j$ with the inner product $\langle v_i, v_j \rangle$, obtaining

$$\mathrm{SDP} = \max_{v_1,\ldots,v_n \in S^{n-1}} \sum_{\{i,j\} \in E} \frac{1 - \langle v_i, v_j \rangle}{2}.$$



Signs ± 1: two antipodal points;
the relaxation allows all unit vectors.

There are three questions to address: why this is a relaxation, whether we can solve it, and how to round its solution back to a cut.

Why is this a relaxation? Given any signs $x_1, \ldots, x_n$, fix the unit vector $v_0 = (1, 0, \ldots, 0)$ and set

$$v_i = x_i v_0.$$

These vectors are either v_0 or $-v_0$, and

$$\langle v_i, v_j \rangle = x_i x_j \langle v_0, v_0 \rangle = x_i x_j.$$

Thus, every solution of the original quadratic program gives a vector solution with exactly the same objective value. The original two choices form a zero-dimensional sphere, embedded in S^{n-1}. Consequently,

$$\mathrm{SDP} \geq \mathrm{OPT}\,.$$

Why can we solve it? Introduce the Gram matrix X, whose entries are

$$X_{ij} = \langle v_i, v_j \rangle.$$

By the Gram matrix characterization, such a matrix is positive semidefinite. Moreover,

$$X_{ii} = \langle v_i, v_i \rangle = \|v_i\|_2^2,$$

so the unit-vector constraints become $X_{ii} = 1$. The vector optimization is therefore equivalent to the semidefinite program

$$\begin{aligned} \text{SDP} = \max_X \quad & \sum_{\{i,j\} \in E} \frac{1 - X_{ij}}{2} \\ \text{subject to} \quad & X \succeq 0, \\ & X_{ii} = 1 \qquad \text{for all } i. \end{aligned}$$

Conversely, a feasible PSD matrix has a Gram representation, and its diagonal constraints ensure that the corresponding vectors have length one. We can therefore solve this relaxation to the desired accuracy using semidefinite programming.

Rounding: cut the sphere by a random hyperplane

The SDP objective favors small inner products between vectors whose vertices are connected by an edge. For unit vectors, this means placing them far apart, ideally in opposite directions.

We now want to turn these vectors into signs while retaining a large objective value. Choose a vector h uniformly from S^{n-1} and set

$$x_i = \operatorname{sgn}\langle v_i, h \rangle.$$

Thus, $x_i = +1$ when $\langle v_i, h \rangle > 0$ and $x_i = -1$ when $\langle v_i, h \rangle < 0$. The zero case has probability zero and may be assigned either sign.

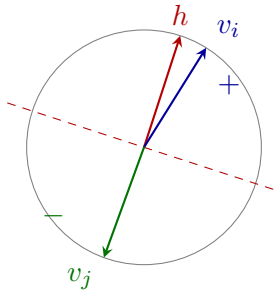
The hyperplane perpendicular to h,

$$h^\perp = \{z \in \mathbb{R}^n : \langle z, h \rangle = 0\},$$

divides the sphere into two sides. We define

$$S = \{i : x_i = -1\}, \qquad \bar{S} = \{i : x_i = +1\}.$$

Exchanging these two names does not change the cut.



By linearity of expectation, the expected size of the rounded cut is

$$\mathbb{E}_h\left[\sum_{\{i,j\}\in E}\frac{1-x_ix_j}{2}\right]=\sum_{\{i,j\}\in E}\mathbb{E}_h\left[\frac{1-x_ix_j}{2}\right].$$

Each summand is the probability that the corresponding edge is cut, because $(1-x_ix_j)/2$ is one when the endpoints receive opposite signs and zero otherwise. We can therefore analyze one edge at a time.

Fix an edge $\{i,j\}$ and let

$$\theta_{ij}=\angle(v_i,v_j).$$

The relevant geometry lies in the plane spanned by v_i and v_j. The intersection of this plane with the rounding hyperplane becomes a random line through the center of a circle.

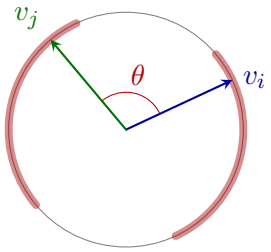
The two vectors receive opposite signs precisely when the line separates them. The separating normal directions occupy two arcs of total angle $2\theta_{ij}$ out of a full angle 2π. Hence,

$$\mathbb{P}[\{i,j\}\text{ is cut}]=\frac{2\theta_{ij}}{2\pi}=\frac{\theta_{ij}}{\pi}.$$

Since v_i and v_j are unit vectors, $\langle v_i,v_j\rangle=\cos\theta_{ij}$. Their contribution to the SDP objective is therefore

$$\frac{1-\langle v_i,v_j\rangle}{2}=\frac{1-\cos\theta_{ij}}{2}=\sin^2\left(\frac{\theta_{ij}}{2}\right).$$

We must compare this SDP contribution with the probability θ_{ij}/π that rounding cuts the edge.



Shaded: normal directions h that separate v_i, v_j;
two arcs of total angle 2θ.

The Goemans–Williamson constant

Write

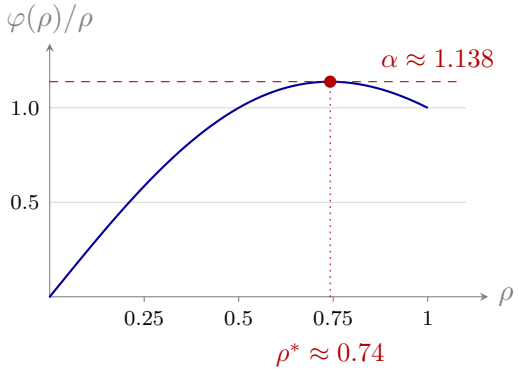
$$\rho=\frac{\theta}{\pi}\in(0,1],\qquad \varphi(\rho)=\sin^2\left(\frac{\rho\pi}{2}\right).$$

For an edge with angle θ, rounding cuts the edge with probability ρ, while the SDP assigns it the contribution $\varphi(\rho)$.

To compare these quantities uniformly over all angles, define,

$$\alpha=\min_{\rho\in(0,1]}\frac{\varphi(\rho)}{\rho}=\min_{\rho\in(0,1]}\frac{\sin^2(\rho\pi/2)}{\rho}\approx 1.1382.$$

Here α is the factor by which the SDP contribution can exceed the rounding probability; the fraction retained by rounding is α^{-1}.



By the definition of α,

$$\frac{\varphi(\rho)}{\rho} \le \alpha \qquad \Longrightarrow \qquad \rho \ge \alpha^{-1}\varphi(\rho).$$

Thus, for every edge,

$$\mathbb{P}[\{i,j\} \text{ is cut}] \ge \alpha^{-1}\frac{1-\langle v_i, v_j\rangle}{2}.$$

When $\theta_{ij} = 0$, both sides are zero, so the inequality also holds in that case.

Summing over all edges gives the rounding guarantee:

$$\begin{aligned}\mathbb{E}[|E(S,\bar{S})|] &= \sum_{\{i,j\}\in E} \mathbb{P}[\{i,j\} \text{ is cut}] \\ &\ge \alpha^{-1} \sum_{\{i,j\}\in E} \frac{1-\langle v_i, v_j\rangle}{2}.\end{aligned}$$

For an optimal solution of the relaxation, this yields

$$\mathbb{E}[|E(S,\bar{S})|] \ge \alpha^{-1}\,\mathrm{SDP} \ge \alpha^{-1}\,\mathrm{OPT} \approx 0.87856\ \mathrm{OPT}\,.$$

This improves the factor-2 guarantee of a random cut to a factor of approximately 1.1382.

Theorem 4 (Goemans–Williamson, 1995)**.** *Solving the SDP and rounding with a random hyperplane gives a cut of expected size at least*

$$\alpha^{-1}\,\mathrm{OPT} \approx 0.878\ \mathrm{OPT}\,.$$

This approximation guarantee is optimal among polynomial-time algorithms, assuming the Unique Games Conjecture.

Near-Bipartite Graphs

We assume we have a graph with max-cut of size $(1-\epsilon)|E|$, that is, the cut cuts almost all of the edges except ϵ fraction of them. We want to find a cut of size $(1-f(\epsilon))|E|$, if we know OPT $\ge (1-\varepsilon)\,|E|$, such that $f(\epsilon) \to 0$ as $\epsilon \to 0$. We can find a cut of size $(1-\sqrt{\epsilon})|E|$.

To obtain this, we follow the same procedure as before, using SDP, random hyperplane and analysing the rounding procedure. Except, we rephrase our problem to minimising the number of uncut edges and showing the number of uncut edges upperbounded by $\sqrt{\epsilon}|E|$.

Therefore, we have,

$$min \sum_{\{i,j\}\in E} \frac{1+\langle v_i, v_j\rangle}{2} \leq \epsilon\,|E|$$

Let us define

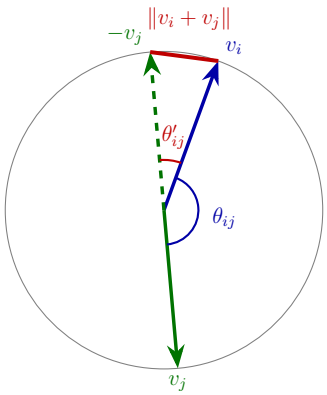
$$\gamma_{ij} = 1 + \langle v_i, v_j\rangle \geq 0.$$

This gives us

$$\frac{1}{|E|}\sum_{\{i,j\}\in E} \gamma_{ij} \leq 2\varepsilon. \tag{1}$$

The final inequality says that the average deviation from being antipodal across all edges is at most 2ε. When γ_{ij} is small, the endpoints of edges are almost opposite to each other and the probability of not cutting them is small.

Let $\theta'_{ij} = \angle(v_i, -v_j)$ and $\rho'_{ij} = \Pr(i.j \text{ is not cut}) = \frac{\theta'_{ij}}{\pi}$.



For small θ'_{ij}, $\rho'_{ij} \leq \|v_i - (-v_j)\|$.

$$\|v_i + v_j\| = \sqrt{2 + 2\langle v_i, v_j\rangle} = \sqrt{2\gamma_{ij}}$$
$$\frac{1}{|E|}\sum_{\{i,j\}\in E} \rho'_{ij} \leq \frac{1}{|E|}\sum_{\{i,j\}\in E} \frac{\|v_i + v_j\|}{2} \leq \frac{1}{|E|}\sum_{\{i,j\}\in E} \sqrt{\frac{\gamma_{ij}}{2}}$$

Using Jensen Inequality, we have,

$$\frac{1}{|E|}\sum_{\{i,j\}\in E} \rho'_{ij} \leq \sqrt{\frac{1}{|E|}\sum_{\{i,j\}\in E} \frac{\gamma_{ij}}{2}} \leq \sqrt{\epsilon}.$$

Theorem 5. *If* OPT $\geq (1-\varepsilon)|E|$*, the same SDP and rounding return a cut of expected size at least* $(1-\sqrt{\varepsilon})|E|$.

Planted Clique and the Lovász Theta Function

The planted-clique problem

Our second application of semidefinite programming is the planted-clique problem. Recall that in the Erdős–Rényi model $G(n, 1/2)$, each unordered pair of distinct vertices is independently included as an edge with probability $1/2$.

To plant a clique, we first sample a graph $H \sim G(n, 1/2)$ and choose a set $S \subseteq [n]$ of size k, independently of H. We then add every missing edge within S. Thus, every pair of distinct vertices in S is connected, and S forms a clique of size k.

We are given a graph and want to distinguish between two possibilities:

1. **The null case:** the graph is sampled from $G(n, 1/2)$.
2. **The planted case:** the graph is sampled from $G(n, 1/2)$ and a clique of size k is planted.

Our goal is to determine which case we are in; we do not require the algorithm to recover the planted vertex set.

A spectral algorithm

A natural approach is to look at the largest eigenvalue of the signed adjacency matrix A_G, defined by

$$(A_G)_{ij} = \begin{cases} 1, & \text{if } \{i,j\} \in E(G), \\ -1, & \text{if } i \neq j \text{ and } \{i,j\} \notin E(G), \\ 0, & \text{if } i = j. \end{cases}$$

Unlike the usual adjacency matrix, this matrix assigns -1 to nonedges.

Suppose G contains a clique S of size k, and let $\mathbf{1}_S$ denote its indicator vector. All off-diagonal entries of A_G indexed by vertices in S equal 1, so

$$\mathbf{1}_S^\top A_G \mathbf{1}_S = k(k-1).$$

Since $\|\mathbf{1}_S\|_2^2 = k$,

$$\lambda_{\max}(A_G) \geq \frac{\mathbf{1}_S^\top A_G \mathbf{1}_S}{\|\mathbf{1}_S\|_2^2} = k - 1.$$

Thus, a large clique forces a large eigenvalue.

On the other hand, for a random graph $H \sim G(n, 1/2)$, the standard random-matrix bound gives

$$\|A_H\|_{\text{op}} \lesssim \sqrt{n}$$

with high probability.

A semirandom adversary

We now allow an adversary to modify the graph after it has been sampled:

- in the planted case, the adversary may add edges;
- in the null case, the adversary may delete edges.

The adversary may know both the sampled graph and our algorithm, and need not be computationally bounded.

These changes appear helpful: adding edges preserves the planted clique, while deleting edges cannot create a new clique. However, they can destroy the guarantee of the spectral algorithm.

For example, in the null case, the adversary can divide the vertices into two subsets of equal size and delete all edges between them. The signed adjacency matrix then has two off-diagonal blocks consisting entirely of -1. In this way, the adversary can make

$$\lambda_{\max}(A_G) = \Omega(n)$$

with high probability, even though the original graph was sampled from the null model.

We therefore need a statistic whose behavior under edge additions and deletions is controlled.

A monotone spectral certificate

We seek a graph statistic $\theta'(G)$ with three properties:

1. A clique of size k forces
$$\theta'(G) \geq k - 1.$$
2. Adding edges cannot decrease its value. For graphs on the same vertex set,
$$E(G_1) \subseteq E(G_2) \quad \Longrightarrow \quad \theta'(G_1) \leq \theta'(G_2).$$
3. On a random graph $H \sim G(n, 1/2)$, it satisfies
$$\theta'(H) = O(\sqrt{n})$$
with high probability.

The second property is called *monotonicity*. It ensures that the adversary's allowed modifications move the statistic in the desired direction.

We can strengthen the third property by asking that, for every graph G,

$$(3') \qquad \theta'(G) \leq \lambda_{\max}(A_G).$$

This implies property (3), since the largest eigenvalue of the signed adjacency matrix of $G(n, 1/2)$ is $O(\sqrt{n})$ with high probability.

To motivate the certificate, we could first ask for the smallest spectral upper bound obtained from a graph G' containing G:

$$\min_{G' \supseteq G} \lambda_{\max}(A_{G'}).$$

Every such graph preserves the cliques of G, adding edges to G restricts the set of possible supergraphs, and choosing $G' = G$ gives property (3'). In these signed adjacency matrices, entries

on edges of G are fixed at 1, while entries on nonedges can be either -1 or 1; allowing the latter to take arbitrary real values leads naturally to the following optimization problem.

Define

$$\begin{aligned} \theta'(G) = \min_{B} \quad & \lambda_{\max}(B) \\ \text{subject to} \quad & B = B^{\top}, \\ & B_{ii} = 0 \qquad \text{for all } i, \\ & B_{ij} = 1 \qquad \text{for all } \{i,j\} \in E(G). \end{aligned}$$

The entries corresponding to nonedges are unrestricted.

The signed adjacency matrix fixes these entries at -1. Here, we allow the optimization to choose them in whichever way minimizes the largest eigenvalue. The entries imposed by actual edges remain fixed. We now check that this quantity has the three properties we need.

Why does a clique force a large value? Suppose S is a clique of size k, and let B be any feasible matrix. Every off-diagonal entry within S must equal 1, while the diagonal entries are zero. Therefore,

$$\mathbf{1}_S^{\top} B \mathbf{1}_S = k(k-1).$$

This again gives

$$\lambda_{\max}(B) \geq \frac{\mathbf{1}_S^{\top} B \mathbf{1}_S}{\|\mathbf{1}_S\|_2^2} = k - 1.$$

This holds for every feasible B, regardless of how the nonedge entries are chosen. Taking the minimum therefore gives

$$\theta'(G) \geq k - 1.$$

In particular, if $\omega(G)$ denotes the size of a largest clique in G, then

$$\omega(G) \leq \theta'(G) + 1.$$

Why is the certificate monotone? Adding an edge introduces an additional constraint of the form $B_{ij} = 1$. Thus, adding edges makes the feasible set smaller.

Writing $\mathcal{F}(G)$ for the feasible set associated with G, we have

$$E(G_1) \subseteq E(G_2) \quad \Longrightarrow \quad \mathcal{F}(G_2) \subseteq \mathcal{F}(G_1).$$

Minimizing over a smaller set cannot decrease the minimum, so

$$\theta'(G_1) = \min_{B \in \mathcal{F}(G_1)} \lambda_{\max}(B) \leq \min_{B \in \mathcal{F}(G_2)} \lambda_{\max}(B) = \theta'(G_2).$$

Therefore, adding edges cannot decrease the certificate, and deleting edges cannot increase it.

Why is it small on a random graph? For any graph G, the signed adjacency matrix A_G is a feasible choice of B: it is symmetric, has zero diagonal, and equals 1 on every edge. Consequently,

$$\theta'(G) \leq \lambda_{\max}(A_G) \leq \|A_G\|_{\mathrm{op}}.$$

For $H \sim G(n, 1/2)$, the random-matrix bound gives

$$\theta'(H) \le \|A_H\|_{\mathrm{op}} \lesssim \sqrt{n}$$

with high probability. This establishes the third property.

Why is this an SDP?

The objective $\lambda_{\max}(B)$ is not linear in the entries of B. To reformulate the problem, we introduce a variable γ that bounds the largest eigenvalue.

For a symmetric matrix B,

$$\lambda_{\max}(B) \le \gamma \quad \Longleftrightarrow \quad \gamma I - B \succeq 0.$$

Indeed, the eigenvalues of $\gamma I - B$ are $\gamma - \lambda_i(B)$. They are all nonnegative exactly when every eigenvalue of B is at most γ.

Thus, we can write

$$\begin{aligned} \theta'(G) = \min_{\gamma, B} \quad & \gamma \\ \text{subject to} \quad & \gamma I - B \succeq 0, \\ & B_{ii} = 0 \qquad \text{for all } i, \\ & B_{ij} = 1 \qquad \text{for all } \{i,j\} \in E(G), \\ & B = B^\top. \end{aligned}$$

Now set $X = \gamma I - B$. The zero diagonal of B gives $X_{ii} = \gamma$, while $B_{ij} = 1$ on edges gives $X_{ij} = -1$. We obtain the equivalent formulation

$$\begin{aligned} \theta'(G) = \min_{\gamma, X} \quad & \gamma \\ \text{subject to} \quad & X \succeq 0, \\ & X_{ii} = \gamma \qquad \text{for all } i, \\ & X_{ij} = -1 \quad \text{for all } \{i,j\} \in E(G). \end{aligned}$$

The objective and equality constraints are linear, and the remaining constraint is positive semidefiniteness. Hence, this is a semidefinite program.

Connection to the Lovász theta function

The certificate is closely related to the Lovász theta function. One standard characterization of this function, for a graph H, is

$$\begin{aligned} \vartheta(H) = \min_{M} \quad & \lambda_{\max}(M) \\ \text{subject to} \quad & M = M^\top, \\ & M_{ii} = 1 \qquad \text{for all } i, \\ & M_{ij} = 1 \qquad \text{if } i \neq j \text{ and } \{i,j\} \notin E(H). \end{aligned}$$

The entries corresponding to edges of H are unrestricted.

Take $H = \overline{G}$, the complement of G. For distinct vertices, being a nonedge of $\overline{G}$ is equivalent to being an edge of G. Thus, the constrained off-diagonal entries are exactly those used in the definition of $\theta'(G)$.

The only remaining difference is the diagonal. Setting $M = B + I$ changes the diagonal entries from zero to one and shifts every eigenvalue by one:

$$\lambda_{\max}(M) = \lambda_{\max}(B) + 1.$$

Therefore,

$$\theta'(G) = \vartheta(\overline{G}) - 1.$$

In this notation, the clique bound becomes

$$\omega(G) \le \vartheta(\overline{G}).$$

Distinguishing the semirandom cases

We can now combine the three properties of the certificate.

In the null case, the observed graph G is obtained by deleting edges from $H \sim G(n, 1/2)$. By monotonicity,

$$\theta'(G) \le \theta'(H) \lesssim \sqrt{n}$$

with high probability. On the event that $\theta'(H) \lesssim \sqrt{n}$, this bound holds for every choice of edges the adversary might delete.

In the planted case, the adversary only adds edges, so the observed graph still contains the planted clique of size k. Hence,

$$\theta'(G) \ge k - 1.$$

If $k - 1 > \sqrt{n}$, choose a threshold t satisfying

$$\sqrt{n} < t < k - 1.$$

Assuming exact evaluation of the certificate, we declare the graph to be from the planted case when $\theta'(G) > t$, and from the null case otherwise. The preceding bounds show that this distinguishes the two cases with high probability.

Thus, the certificate retains the $k \gtrsim \sqrt{n}$ detection guarantee in the presence of the semirandom adversary. Its monotonicity ensures that deleting edges in the null case cannot raise its value, while adding edges in the planted case preserves the lower bound imposed by the clique.

Summary

- **SDP:** a linear objective and linear constraints over the PSD cone;

$$X \succeq 0 \quad \iff \quad X \text{ is a Gram matrix.}$$

- **Max-Cut:** vectors on a sphere and random hyperplane rounding give an expected cut size of at least 0.878 OPT, and $(1 - \sqrt{\varepsilon})|E|$ when $\text{OPT} \ge (1 - \varepsilon)|E|$.
- **Planted clique:** $\theta'(G) = \vartheta(\overline{G}) - 1$ is a monotone certificate, robust to a semirandom adversary.

AI Use Disclosure

We used LLMs to gain additional insight into our own lecture notes and the slides and to generate TikZ figures.