Combinatorial optimization and SDP/SOS.
Part I: the maximum cut problem
Dávid Papp, Spring 2019

Given a *complete* graph $G = (V = [n], E)$ with nonnegative edge weights $w\colon E \mapsto \mathbb{R}_+$, a **cut** is a subset of the vertices $S \subseteq V$. The **value** of the cut S is $w(S) \stackrel{\text{def}}{=} \sum_{i\in S, j\in\bar{S}} w_{ij}$. (Here, $\bar{S} = V \setminus S$.)

The *maximum cut problem* is to find a cut of maximum weight in a given edge-weighted complete graph. The maximum cut problem can be "solved" trivially, but enumeration is prohibitively expensive. The maximum cut problem is NP-hard. Every cut gives a lower bound on the value of the maximum cut. Natural questions: can we find good cuts in polynomial time? Can we find a good *upper* bound on the value of the maximum cut in polynomial time?

Approach 1: a greedy (or myopic) algorithm. Assign each vertex $i = 1, \dots, n$, in order, to S or $\bar{S}$ depending on whether

$$\sum_{j\le i-1, j\in S} w_{ij} > \sum_{j\le i-1, j\in\bar{S}} w_{ij}.$$

Exercise. The greedy algorithm in general does not find a maximum cut. However, the value of the cut obtained by it is *at least half of the value of the maximum cut. Hint:* the value of the cut is at least $\frac{1}{2}\sum_{ij\in E} w_{ij}$. Thus, this greedy algorithm is a **$\frac{1}{2}$-approximation algorithm.**

Approach 2: a randomized algorithm. Flip n unbiased coins, then assign each vertex $i = 1, \dots, n$ to S or $\bar{S}$ depending on whether coin i came up heads or tails.

Exercise. The expected value of the cut returned by the above randomized algorithm is at least half of the value of the maximum cut.

Surprisingly, there are no known elementary or combinatorial algorithms with a better approximation ratio. However, using SDP, we can obtain a **0.878-approximation algorithm**, due to Michel Goemans and David Williamson[1], as follows.

We can formulate the problem as a quadratic optimization problem over the n-dimensional hypercube: define the vector $x \in \{-1, 1\}^n$ such that

$$x_i = \begin{cases} +1 & i \in S \\ -1 & i \in \bar{S} \end{cases}.$$

Then the weight of a cut S is

$$w(S) = \sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}\frac{1 - x_i x_j}{2}.$$

[1] `http://www-math.mit.edu/~goemans/PAPERS/maxcut-jacm.pdf`

Thus, the maximum cut problem can be formulated as

$$\max_{x\in\{-1,1\}^n} \sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}\frac{1-x_ix_j}{2}. \tag{1}$$

This is still the same NP-hard problem as before. We can massage it into an SDP with a rank constraint. Introduce a new (matrix) variable for $X \stackrel{\text{def}}{=} xx^{\mathrm{T}}$. Then X is positive semidefinite and has rank one. Moreover, $X_{ii} = x_i^2 = 1$ (this is why we opted for the ± 1 "encoding" in place of the 0/1). We can see (think this through!!) that (1) is *equivalent* to

$$\begin{array}{ll} \text{maximize} & \displaystyle\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}\frac{1-X_{ij}}{2} \\ \text{subject to} & X \in PSD_n \\ & X_{ii} = 1 \quad (i = 1,\ldots,n) \\ & \operatorname{rank}(X) = 1. \end{array}$$

This is still the same NP-hard problem as before. Dropping the rank constraint, we obtain the **SDP relaxation**:

$$\begin{array}{ll} \text{maximize} & \displaystyle\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}\frac{1-X_{ij}}{2} \\ \text{subject to} & X \in PSD_n \\ & X_{ii} = 1 \quad (i = 1,\ldots,n). \end{array} \tag{2}$$

This SDP can be solved in polynomial time (this is slightly less obvious than it is often made out to be), and because it is a relaxation of a maximization problem, it provides an *upper bound* on the value of the maximum cut.

How good is this bound? Can we obtain a high-value cut from it?

Here is a geometric interpretation (or alternative derivation) of the SDP. Instead of representing each vertex with a binary variable, we can represent them with an n-dimensional *unit vector* $v_i \in S_n$, and replace x_ix_j with $v_i^{\mathrm{T}}v_j$, since with the choice of $v_i = (x_i, 0, \ldots, 0)$, each v_i is a unit vector, and $x_ix_j = v_i^{\mathrm{T}}v_j$. Thus, this is again a *relaxation* of (1):

$$\begin{array}{ll} \text{maximize} & \displaystyle\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}\frac{1-v_i^{\mathrm{T}}v_j}{2} \\ \text{subject to} & \|v_i\| = 1 \quad (i = 1,\ldots,n). \end{array} \tag{3}$$

The problems (2) are (3) are equivalent. Given a feasible solution to (3), we can take V as the matrix whose columns are $v_1, \ldots, v_n$. Then, $X = V^{\mathrm{T}}V$ is a feasible solution to (2), with the same objective function value. Conversely, for every feasible solution X of (2), we can find a matrix V satisfying $X = V^{\mathrm{T}}V$ (even in polynomial time, e.g., using Cholesky

decomposition). The columns of this V yield a feasible solution to (3) with the same objective function value.

The second relaxation is more intuitive to extract a cut from. (**Picture!**) Given $v_1, \ldots, v_n$, choose a hyperplane through the origin (but otherwise in general position), and assign the vertex i depending on which side of the hyperplane is v_i on.

Approach 3: SDP with randomized rounding. (This is the Goemans–Williamson algorithm.) Solve the SDP relaxation (2), factorize X into to $V^{\mathrm{T}}V$, then choose a hyperplane through the origin *uniformly at random*, and assign the vertex i depending on which side of the hyperplane is v_i on.

Exercise. How does one "choose a hyperplane through the origin uniformly at random"? Equivalently, how can we draw a uniformly random vector on the unit sphere?

Theorem 1. *The Goemans–Williamson algorithm is a 0.878-approximation algorithm (in expectation).*

Proof. Let $z_{\max}$ be the value of the maximum cut, z_{sdp} be the optimal value of the SDP relaxation, and $z_{\exp}$ be the expected value of the cut obtained using the algorithm. Clearly, $z_{\mathrm{sdp}} \geq z_{\max} \geq z_{\exp}$. (Why?) Moreover , letting $v_1, \ldots, v_n$ be the optimal solution from the SDP relaxation, we can see (**Picture!**) that

$$\begin{aligned} z_{\exp} &= \sum_{ij \in E} w_{ij} \mathrm{Prob}[\text{vertices } i \text{ and } j \text{ are separated}] \\ &= \sum_{ij \in E} w_{ij} \frac{2\measuredangle(v_i, v_j)}{2\pi} \\ &= \frac{1}{\pi} \sum_{ij \in E} w_{ij} \arccos(v_i^{\mathrm{T}} v_j) \end{aligned}$$

Plugging everything together:

$$v_{\mathrm{sdp}} \geq v_{\max} \geq v_{\exp} \geq \frac{1}{\pi} \sum_{ij \in E} w_{ij} \arccos(v_i^{\mathrm{T}} v_j) \geq \alpha v_{\mathrm{sdp}} \geq \alpha v_{\max},$$

if we can find some coefficient $\alpha > 0$ such that $\frac{1-y}{2} \geq \alpha \frac{\arccos(y)}{\pi}$ for every $y \in [-1, 1]$. (Apply this term-by-term, with $v_i^{\mathrm{T}} v_j$ playing the role of y.) As it happens, we can choose $\alpha > 1/2$; specifically, we can have

$$\alpha = 0.878 < \min_{y \in [-1,1]} \frac{2}{\pi} \frac{\arccos(y)}{1-y}. \qquad \square$$