Chapter 6: Randomized Rounding of Semidefinite Programs

(cp. Williamson & Shmoys, Chapter 6)

Semidefinite Matrices

Definition 6.1.

A matrix $X \in \mathbb{R}^{n \times n}$ is positive semidefinite if $y^T \cdot X \cdot y \geq 0$ for all $y \in \mathbb{R}^n$.
In this case we write $X \succeq 0$.

Theorem 6.2.

For symmetric matrix $X \in \mathbb{R}^{n \times n}$ the following statements are equivalent:

i. X is positive semidefinite;
ii. all eigenvalues of X are non-negative;
iii. $X = V^T \cdot V$ for some $V \in \mathbb{R}^{m \times n}$ where $m \leq n$;
iv. $X = \sum_{i=1}^{n} \lambda_i \,(w_i \cdot w_i^T)$ for some $\lambda_i \geq 0$ and $w_i \in \mathbb{R}^n$ such that $w_i^T \cdot w_i = 1$ and $w_i^T \cdot w_j = 0$ for $i \neq j$. □

Semidefinite Programs (SDPs)

Definition 6.3.
A semidefinite program is a linear program with the additional constraint that a square symmetric matrix of variables must be positive semidefinite.

Example.

$$\begin{aligned} \min / \max \quad & \sum_{i,j} c_{ij} x_{ij} & \\ \text{s.t.} \quad & \sum_{ij} a_{ijk} x_{ij} = b_k & \text{for all } k, \\ & x_{ij} = x_{ji} & \text{for all } i, j, \\ & X = (x_{ij}) \succeq 0 & \end{aligned}$$

Remark. The set of feasible solutions of a semidefinite program is convex.

Vector Programs

A semidefinite program can be stated equivalently as a vector program and vice versa (see Theorem 6.2 iii):

$$\begin{aligned} \min / \max \quad & \sum_{i,j} c_{ij} \left(v_i^T \cdot v_j \right) & \\ \text{s.t.} \quad & \sum_{ij} a_{ijk} \left(v_i^T \cdot v_j \right) = b_k & \text{for all } k, \\ & v_i \in \mathbb{R}^n & \text{for all } i = 1, \ldots, n. \end{aligned}$$

Remark.

- Under mild technical conditions, semidefinite programs can be solved within additive error ε in time polynomial in input size and $\log(1/\varepsilon)$.
- For simplicity, we assume in the following that we can efficiently obtain an optimal solution.

SDP Relaxation of MAX CUT

Integer quadratic programming formulation of MAX CUT

$$\begin{aligned} \max \quad & \tfrac{1}{2} \sum_{ij \in E} w_{ij} \, (1 - y_i y_j) \\ \text{s.t.} \quad & y_i \in \{-1, 1\} && \text{for all } i \in V. \end{aligned}$$

Semidefinite programming relaxation of MAX CUT

$$\begin{aligned} \max \quad & \tfrac{1}{2} \sum_{ij \in E} w_{ij} \, (1 - v_i^T \cdot v_j) \\ \text{s.t.} \quad & v_i^T \cdot v_i = 1 && \text{for all } i \in V, \\ & v_i \in \mathbb{R}^n && \text{for all } i \in V. \end{aligned}$$

Randomized Rounding of Vector Program

1. compute (near-)optimal solution (v^*) to SDP relaxation;
2. pick a random vector $r = (r_1, \ldots, r_n)^T$ by drawing each component from $\mathcal{N}(0, 1)$, the normal distribution with mean 0 and variance 1;
3. for $i = 1, \ldots, n$: if $r^T \cdot v_i^* \geq 0$ then put i in S;

Remarks.

- The hyperplane orthogonal to r partitions the n-dimensional unit sphere into two halves, corresponding to S and $V \setminus S$.
- The normalization $r/\|r\|$ of r is uniformly distributed over the n-dimensional unit sphere.
- The projections of r onto two unit vectors e_1, e_2 are independent and normally distributed if and only if e_1 and e_2 are orthogonal.

Corollary 6.4.

Let r' the projection of r onto a 2-dimensional plane. The normalization $r'/\|r'\|$ of r', is uniformly distributed on a unit circle in the plane. □

Analysis of the SDP-based Algorithm

Lemma 6.5.

The probability that edge $ij \in E$ is in the cut is $\frac{1}{\pi}\arccos(v_i^T \cdot v_j)$.

Proof:. . . □

Lemma 6.6.

For $x \in [-1,1]$ it holds that $\frac{1}{\pi}\arccos(x) \geq 0.878 \cdot \frac{1}{2}(1-x)$. □

Theorem 6.7.

SDP-based randomized rounding is a 0.878-approximation algorithm for MAX CUT.

Proof:. . . □

Remark. The algorithm can be derandomized by using a sophisticated application of the method of conditional expectations.

Inapproximability Results for MAX CUT

We state the following results without proof.

Theorem 6.8.

If there is an α-approximation algorithm for MAX CUT with $\alpha > 16/17 \approx 0.941$, then $P = NP$. □

Theorem 6.9.

Given the *Unique Games Conjecture* there is no α-approximation algorithm for MAX CUT with constant

$$\alpha > \min_{-1\leq x\leq 1} \frac{\frac{1}{\pi}\arccos(x)}{\frac{1}{2}(1-x)} \approx 0.878$$

unless $P = NP$. □

Coloring 3-Colorable Graphs

Observation 6.10.
A graph of maximum degree Δ can be $(\Delta + 1)$-colored in polynomial time.

Simple algorithm for coloring 3-colorable graphs:

1. while there is a node v of degree $\geq \sqrt{n}$
2. color nodes in $\{v\} \cup N(v)$ with 3 new colors and delete them;
3. color the remaining nodes with $\sqrt{n}$ new colors;

Theorem 6.11.
The algorithm above colors any 3-colorable graph with at most $4\sqrt{n}$ colors.

Proof:... □

Semidefinite Programming Relaxation

Idea:
Associate nodes with unit vectors such that adjacent nodes are far apart:

$$\begin{array}{lll} \min & \lambda & \\ \text{s.t.} & v_i^T \cdot v_j \leq \lambda & \text{for all } ij \in E, \\ & v_i^T \cdot v_i = 1 & \text{for all } i \in V, \\ & v_i \in \mathbb{R}^n & \text{for all } i \in V. \end{array}$$

Lemma 6.12.
If $G = (V, E)$ is 3-colorable, there is feasible SDP solution with $\lambda \leq -\frac{1}{2}$.

Proof:... □

Randomized Algorithm for Semicoloring

Definition 6.13.

A semicoloring is a coloring of nodes such that there are at most $n/4$ monochromatic edges (i. e., feasible coloring for subgraph on $n/2$ nodes.)

Randomized SDP-based algorithm

1. compute optimal solution v^* to the semidefinite program;
2. choose $t := 2 + \log_3 \Delta$ random vectors $r_1, \ldots, r_t$;
   // (draw each component of r_i from $\mathcal{N}(0, 1)$)
3. color nodes in V with at most 2^t colors such that $i, j \in V$ get the same color if and only if $\operatorname{sgn}(v_i^T \cdot r_k) = \operatorname{sgn}(v_j^T \cdot r_k)$ for $k = 1, \ldots, t$;

Theorem 6.14.

The algorithm colors the nodes with at most $4\Delta^{\log_3 2}$ colors. This is a semicoloring with probability at least $1/2$.

Proof:. . . □ 99

Iterative Coloring Algorithm

Let $g(n) \in \tilde{O}(f(n))$ if $g(n) \in O\big(f(n) \log^c n\big)$ for some constant $c \geq 0$.

Applying the above algorithm iteratively ($O(\log n)$ times) yields $\tilde{O}(n^{\log_3 2})$-coloring. Since $\log_3 2 \approx 0.631$, this is worse than $\sqrt{n}$.

Improved Algorithm

1. while there is a node v of degree $\geq \sigma$
2. color nodes in $\{v\} \cup N(v)$ with 3 new colors and delete them;
3. use above algorithm to $O(\sigma^{\log_3 2} \log n)$-color remaining nodes;

Theorem 6.15.

For $\sigma = n^{\log_6 3}$, the algorithm colors a 3-colorable graph with $\tilde{O}(n^{0.387})$ colors.

Proof:. . . □