ADVANCED ALGORITHMS (III)

CHIHAO ZHANG

1. MaxCut

Today we consider the problem of MaxCut, which is very similar to Max2SAT we met last week. The problem is formally defined as

MaxCut	
Input:	An undirected graph $G = (V, E)$.
Problem:	Compute a set $S \subseteq V$ that maximizes $\left\lvert E(S, \bar{S})\right\rvert$.

Here, for every set of vertices $S \subseteq V$, the set $E(S, \bar{S})$ is the set of edges with one end in S and the other in $\bar{S}$. These edges form a *cut* since their removal disconnects S and $\bar{S}$. Sometimes we may have nonnegative edge weight w_e on each edge $e \in E$, and the problem becomes to finding a set S so that $\sum_{e \in E(S,\bar{S})} w_e$ is maximized. In this lecture, we only focus on the unweighted case, namely $w_e = 1$ for every $e \in E$. It is not hard to generalize all algorithms we are going to introduce today to the weighted case.

If we consider each vertex $v \in V$ as a variable and each edge as a constraint, then the problem is quite similar to MaxSAT, except that the constraint here is **XOR** instead of **OR**. So we can construct S by tossing an independent fair coin on each vertex: if the coin on a vertex v goes HEAD, we put v into S; otherwise we put v into $\bar{S}$. It is clear that the expectation of this randomized algorithm is a $\frac{1}{2}$-approximation of MaxCut, and we can derandomize the algorithm using *conditional expectation method* in the usual way.

Like the case of MaxSAT, we want to improve the approximation ratio of this simple algorithm by tossing clever coins. Following the strategy we used for MaxSAT, we desgin a linear programming relaxation of the problem.

To desgin a LP relaxation, it is natural to introduce a variable $x_v \in \{0, 1\}$ for every $v \in V$ to indicate whether v is in S. Similarly, we introduce variables $y_{u,v}, y_{v,u} \in \{0, 1\}$ for every $\{u, v\} \in E$ to indicate whether the edge $\{u, v\}$ is in the cut.

The cost function is of course $\sum_{\{u,v\} \in E} y_{u,v}$. However, it is not very straightforward to write *linear* constraints for being a cut. We make use of the following observation: For every $S \subseteq V$, the tuple $(S, \bar{S}, E(S, \bar{S}))$ must be an induced bipartite subgraph of G. Therefore, we try to use linear constraints to characterize those induced bipartite subgraphs. To implement this idea, we need to extend our definition of variables $y_{u,v}$ to every pair of vertices $u, v \in V$ (not only for those edges in E). The integer programming

$$
\begin{aligned}
&\max \quad \sum_{\{u,v\} \in E} y_{u,v} \\
(1) \qquad &\text{s.t.} \quad y_{u,v} \le y_{u,w} + y_{w,v}, \quad \forall u, v, w \in V \qquad \text{(P1)} \\
(2) \qquad &\sum_{\{u,v\} \in C} y_{u,v} \le |C| - 1, \quad \forall \text{odd cycle } C \\
&y_{u,v} = y_{v,u}, \quad \forall u, v \in V \\
&y_{u,v} \in \{0, 1\}, \quad \forall u, v \in V.
\end{aligned}
$$

is equivalent to MaxCut. Here, (1) is used to rule out the case that $y_{u,v} = 1, y_{u,w} = y_{v,w} = 0$ for any $u, v, w \in V$, which is not allowed in a bipartition; (2) is to guarantee that there is no odd cycle. In fact, (2) may involve exponentially many constraints, since we need to consider every odd cycle in G. Now we claim that these constraints

can be replaced by (2'), which involves at most polynomially many constraints.

$$
\begin{aligned}
\max \quad & \sum_{\{u,v\}\in E} y_{u,v} \\
\text{s.t.} \quad & y_{u,v} \le y_{u,w} + y_{w,v}, \quad \forall u,v,w \in V \\
& y_{u,v} + y_{u,w} + y_{w,v} \le 2, \quad \forall u,v,w \in V \\
& y_{u,v} = y_{v,u}, \quad \forall u,v \in V \\
& y_{u,v} \in \{0,1\}, \quad \forall u,v \in V.
\end{aligned}
\tag{P2}
$$

(2')

We leave as an exercise to verify that (P1) and (P2) are equivalent. Hence by relaxing $y_{u,v} \in \{0,1\}$ to $y_{u,v} \in [0,1]$, we obtain the following LP:

$$
\begin{aligned}
\max \quad & \sum_{\{u,v\}\in E} y_{u,v} \\
\text{s.t.} \quad & y_{u,v} \le y_{u,w} + y_{w,v}, \quad \forall u,v,w \in V \\
& y_{u,v} + y_{u,w} + y_{w,v} \le 2, \quad \forall u,v,w \in V \\
& y_{u,v} = y_{v,u}, \quad \forall u,v \in V \\
& y_{u,v} \in [0,1], \quad \forall u,v \in V.
\end{aligned}
\tag{P3}
$$

2. Integrality Gap of MaxCut LP

Can we obtain a better approximation algorithm by rounding (P3)? The answer is probably no, since the LP relaxation has large integrality gap. In this section, we show that the integrality gap of (P3) can be arbitrarily close to 2.

Theorem 1. *For any $\varepsilon > 0$, there exists a graph G satisfying*

$$\frac{\mathrm{LP}(G)}{\mathrm{MaxCut}(G)} \ge 2 - \varepsilon,$$

where $\mathrm{LP}(G)$ is the optimal cost of (P3) on G.

Our proof heavily rely on the Chernoff bound. We choose to use the following simple (but loose) form of multiplicative Chernoff bound.

Proposition 2. *Let X be the sum of n independent Bernoulli trials with success probability p and $\mu = pn$ be the expectation of X. Then*

$$\mathbf{Pr}\left[X \le (1-\delta)\mu\right] \le \exp\left(-\frac{\delta^2\mu}{2}\right), \quad \forall 0 \le \delta \le 1$$

$$\mathbf{Pr}\left[X \ge (1+\delta)\mu\right] \le \exp\left(-\frac{\delta^2\mu}{2+\delta}\right), \quad \forall \delta \ge 0$$

We prove Theorem 1 by the probabilistic method, i.e., instead of directly constructing an instance, we only prove its existence. An Erdős-Rényi random graph $G \sim \mathcal{G}(n,p)$ is a random n vertices graph in which every edge is present with probability p independently.

Assume $G = (V,E) \sim \mathcal{G}(n,p)$ for some parameter p to be set. We first upper bound the probability that $\textsc{MaxCut}(G)$ is *at least* $(1+\delta)\cdot\frac{pn^2}{4}$ for some $\delta \ge 0$. For a fix partition $(S,\bar{S})$ of V, the number of pairs of vertices in $S\times\bar{S}$ is at most $\frac{n^2}{4}$, and each of which has probability p to be in E. We let $X = \mathrm{Bin}\left(\frac{n^2}{4},p\right)$[1], then $\mathbf{E}[X] = \frac{pn^2}{4}$. Therefore, it follows from Proposition 2 that

$$\mathbf{Pr}\left[\left|E(S,\bar{S})\right| \ge (1+\delta)\cdot\frac{pn^2}{4}\right] \le \mathbf{Pr}\left[X \ge (1+\delta)\cdot\mathbf{E}[X]\right] \le \exp\left(-\frac{\delta^2 pn^2}{4(2+\delta)}\right).$$

[1] $Bin(n,p)$ is the binomial distribution.

There are at most 2^n partitions of V, so by the union bound, we have

$$\mathbf{Pr}\left[\textsc{MaxCut}(G) \geq (1+\delta)\cdot\frac{pn^2}{4}\right] \leq 2^n \cdot \exp\left(-\frac{\delta^2 pn^2}{4(2+\delta)}\right).$$

We then turn to bound the probability that a graph $G = (V,E) \sim \mathcal{G}(n,p)$ has small number of edges. Note that $\mathbf{E}[|E|] = p\binom{n}{2}$, so again by Proposition 2 we have for every $\delta' \geq 0$ and sufficiently large n,

$$\mathbf{Pr}\left[|E| \leq (1-\delta')\cdot\frac{pn^2}{2}\right] \approx \mathbf{Pr}\left[|E| \leq (1-\delta')\cdot\mathbf{E}[|E|]\right] \leq \exp\left(-\frac{\delta'^2 pn(n-1)}{4}\right).$$

Given these bounds, it is not hard to conclude with the following lemma:

Lemma 3. *Assume n is sufficiently large. If the parameters δ, δ' and p satisfy $\delta^2 p, \delta'^2 p \geq \frac{C}{n}$ for sufficiently large constant C, then with probability $1-o(1)$, G is a graph satisfying $\textsc{MaxCut}(G) \leq \frac{1+\delta}{1-\delta'}\cdot\frac{|E|}{2}$.*

Keeping in mind that we need $\textsc{MaxCut}(G) \leq (1+o(1))\cdot\frac{|E|}{2}$, so we have to choose parameters so that $\delta, \delta' = o(1)$ and $p = \omega\left(\frac{1}{n}\right)$.

It remains to show that, for suitable p, a graph $G \sim \mathcal{G}(n,p)$ has large $\mathrm{LP}(G)$ with high probability. We are going to construct a solution of (P1). Imagine that we construct the solution by setting $y_{u,v} = y_{v,u} = \frac{k-1}{k}$ for every edge $\{u,v\} \in E$. It is easy to see that, in order for this solution to be feasible, we do not allow the graph containing short odd cycles. To fix this, we can modify $y_{u,v}$ to zero for those edge $\{u,v\}$ on short cycles. This modification decreases the cost of the LP solution $\sum y_{u,v}$. Therefore, as long as we can show that there are not too many such edges, the cost of this feasible LP solution is close to $|E|$. So we turn to bound the number of edges on short cycles in $\mathcal{G}(n,p)$.

Lemma 4. *Let $p = \frac{\log n}{n}$, a graph $G = (V,E) \sim \mathcal{G}(n,p)$ saitsfies $\mathrm{LP}(G) = (1-o(1))|E|$ with probaiblity $1-o(1)$.*

Proof. Let X be the random variable indicating the number of edges on *simple* cycles of length less than k. We first compute the expectation of X. By the linearity of expectation, we have

$$\mathbf{E}[X] \leq \sum_{i=3}^{k} i\cdot\binom{n}{i}\frac{i!}{2i}p^i \leq (pn)^k.$$

We let $k = \frac{\log n}{\log\log n}$ and denote $\alpha = n(\log n)^{\frac{1}{2}}$. Then by Markov inequality,

$$\mathbf{Pr}[X \geq \alpha] = \mathbf{Pr}\left[X \geq (\log n)^{\frac{1}{2}}\cdot(pn)^k\right] \leq \frac{1}{(\log n)^{\frac{1}{2}}} = o(1).$$

In other words, with probaility $1-o(1)$, the number of edges on simple cycles of length less than k is at most $n(\log n)^{\frac{1}{2}}$. By Chernoff bound, we have $\alpha = o(|E|)$ with probability $1-o(1)$. This implies

$$\mathrm{LP}(G) = \sum_{\{u,v\}\in E} y_{u,v} = (1-o(1))\frac{k-1}{k}|E| = (1-o(1))|E|$$

with probability $1-o(1)$. □

Therefore, Theorem 1 follows from Lemma 3 and Lemma 4 by choosing $\delta, \delta' = \left(\frac{C}{\log n}\right)^{\frac{1}{2}}$ and $p = \frac{\log n}{n}$ where C is a sufficiently large constant.

3. Positive Semi-Definite Matrix

We need to review some concepts in linear algebra before introducing our next algorithm for MaxCut.

Definition 5. We call an $n\times n$ symmetric matrix A *positive semi-definite* if $x^T Ax \geq 0$ holds for every vector $x \in \mathbb{R}^n$ and denote it by $A \succeq 0$.

If we replace the "$\geq$" condition $x^T Ax \geq 0$ above by $>$, $\leq$ and $<$ respectively, we obtain "positive definite" ($A \succ 0$), "negative semi-definite" ($A \preceq 0$) and "negative definite" ($A \prec 0$) matrix A respectively.

There are a few equivalent characterizations of the positive semi-definite matrices.

Proposition 6. *For an $n\times n$ symmetric matrix, the followings are equivalent.*

(1) $A \succeq 0$;
(2) A *has* n *non-negative eigenvalues;*
(3) $A = U^T U$ *for some* $n \times n$ *matrix* $U = \begin{bmatrix} \mathbf{u}_1 & \mathbf{u}_2 & \dots & \mathbf{u}_n \end{bmatrix}$.

To verify Proposition 6, we make use of the spectral decomposition theorem.

Theorem 7 (Spectral Decomposition Theorem)**.** *An* $n \times n$ *symmetric matrix has* n *real eigenvalues* $\lambda_1, \dots, \lambda_n$ *with corresponding eigenvectors* $\mathbf{v}_1, \dots, \mathbf{v}_n$ *which are orthonomal. Moreover, it holds that*

$$A = V \Lambda V^T,$$

where $V = \begin{bmatrix} \mathbf{v}_1 & \mathbf{v}_2 & \dots & \mathbf{v}_n \end{bmatrix}$ *and* $\Lambda = \operatorname{diag}(\lambda_1, \dots, \lambda_n)$.

Proof of Proposition 6. "(1) $\Longrightarrow$ (2)": We know from Theorem 7 that all eigenvalues of A are real. Assume for the sake of contradiction that some eigenvalue λ_j is negative and consider its corresponding eigenvector v_j. We have

$$v_j^T A v_j = \lambda_j v_j^T v_j = \lambda_j \|v_j\|_2^2 < 0.$$

This is a contradiction with $A \succeq 0$.

"(2) $\Longrightarrow$ (3)": Since all λ_i are non-negative, we can define a matrix $\sqrt{\Lambda} \triangleq \operatorname{diag}(\sqrt{\lambda_1}, \dots, \sqrt{\lambda_n})$. Then again by Theorem 7, we have

$$A = V \Lambda V^T = V \sqrt{\Lambda} \cdot \sqrt{\Lambda} V^T = \left(\sqrt{\Lambda} V^T\right)^T \cdot \sqrt{\Lambda} V^T.$$

So we can let $U \triangleq \sqrt{\Lambda} V^T$.

"(3) $\Longrightarrow$ (1)": For every $x \in \mathbb{R}^n$, we have

$$x^T A x = x^T U^T U x = \|Ux\|_2^2 \geq 0.$$

□

4. Positive Semi-Definite Programming

The main tool we are going to develop in this section is *positive semi-definite programming* (SDP). We treat PSD as a generalization of LP. For example, consider the LP

$$\begin{aligned} \max \quad & 2x - 3y \\ \text{s.t.} \quad & x + y \leq 2 \\ & 3x - y \leq 1 \\ & x \geq 0 \\ & y \geq 0 \end{aligned}$$

We can rewrite it in an equivalent matrix form

$$\begin{aligned} \max \quad & \begin{bmatrix} 2 & 0 \\ 0 & -3 \end{bmatrix} \bullet \begin{bmatrix} x & 0 \\ 0 & y \end{bmatrix} \\ \text{s.t.} \quad & \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \bullet \begin{bmatrix} x & 0 \\ 0 & y \end{bmatrix} \leq 2 \\ & \begin{bmatrix} 3 & 0 \\ 0 & -1 \end{bmatrix} \bullet \begin{bmatrix} x & 0 \\ 0 & y \end{bmatrix} \leq 1 \\ & \begin{bmatrix} x & 0 \\ 0 & y \end{bmatrix} \succeq 0 \end{aligned}$$

where for two $n \times n$ matrices $A = (a_{i,j})_{1 \leq i,j \leq n}$ and $B = (b_{i,j})_{1 \leq i,j \leq n}$, the Frobenius inner product of $A \bullet B$ is defined to be $A \bullet B \triangleq \sum_{1 \leq i,j \leq n} a_{i,j} \cdot b_{i,j}$. Here, every matrix appeared is *diagonal* and we have an additional *positive semi-definite constraint* $X \succeq 0$.

As shown in above example, every linear program in standard form can be written in the matrix form in which the variables $\{x_i\}_{i \in n}$ of the LP are collected in a matrix $X = \operatorname{diag}(x_1, \dots, x_n)$. The *positive semi-definite programming*

generalizes the diagonal matrix X to arbitrary symmetric matrix:

$$\begin{aligned} \max \quad & C \bullet X \\ \text{s.t.} \quad & A_k \bullet X \le b_k, \quad \forall k \in [m] \\ & X \succeq 0 \end{aligned} \tag{P4}$$

Here $C = (c(i,j))_{1 \le i,j \le n}$, $X = (x(i,j))_{1 \le i,j \le n}$ and $A_k = (a_k(i,j))_{1 \le i,j \le n}$ for $k \in [m]$ are $n \times n$ matrices.

Sometimes it is convenient to apply (3) of Proposition 6 to get rid of the positive semi-definiteness constraint. Since we require X to be positive semi-definite, there exists a matrix $U = \left[\mathbf{u}_1, \ldots, \mathbf{u}_n\right]$ such that $X = U^T U$. We can rewrite (P4) as the following *vector program*.

$$\begin{aligned} \max \quad & \sum_{1 \le i,j} c(i,j) \cdot \mathbf{u}_i^T \mathbf{u}_j \\ \text{s.t.} \quad & \sum_{1 \le i,j \le n} a_k(i,j) \cdot \mathbf{u}_i^T \mathbf{u}_j \le b_k, \quad \forall k \in [m] \\ & \mathbf{u}_i \in \mathbb{R}^n, \quad i \in [n] \end{aligned}$$

Like LP, we can (approximately) solve SDP (and thus its equivalent vector program) in polynomial-time via ellipsoid method provided an efficient separation oracle.

5. Remark

The $2 - \varepsilon$ integrality gap of MaxCut LP is folklore. The proofs in Section 2 are adapted from [PT94] and [VNT93]. See [Fre04] for a gentle introduction of SDP.

References

[Fre04] Robert M Freund. Introduction to semidefinite programming (sdp). *OCW, Massachusetts Institute of Technology*, page 10, 2004. Available at https://ocw.mit.edu/courses/sloan-school-of-management/15-084j-nonlinear-programming-spring-2004/lecture-notes/lec23_semidef_opt.pdf. 5

[PT94] Svatopluk Poljak and Zsolt Tuza. The expected relative error of the polyhedral approximation of the max-cut problem. *Operations Research Letters*, 16(4):191–198, 1994. 5

[VNT93] Nguyen Van Ngoc and Zsolt Tuza. Linear-time approximation algorithms for the max cut problem. *Combinatorics, Probability and Computing*, 2(2):201–210, 1993. 5