CSCI-B609: A Theorist's Toolkit, Fall 2016 10 Nov 2016

Lecture 21: Maximum Cut and SDP Relaxations

Lecturer: Yuan Zhou *Scribe: Erfan Sadeqi Azer*

In this lecture, we introduce the *MaxCut* problem and several formulations to solve it.

1 Max Cut

Definition 1. *Let $G = (V, E)$ be a simple graph. The goal of* Max Cut *problem is to find:*

$$\max_{S \subseteq V} |edges(S, \bar{S})|.$$

Since this problem is NP-hard, we are interested in the best approximation algorithm. Also from the previous lecture, we already have a 2-approximation algorithm for this problem. So, the goal of this lecture is to get any approximation ratio better than 2.

2 Formulation

The following formulation is a good way to start:

$$\text{Maximize} \sum_{(u,v)\in E} y_{uv}.$$

S.t.,

$$x_u \in \{0, 1\}, \qquad \forall u \in V, \tag{1}$$

$$y_{uv} = x_u(1 - x_v) + x_v(1 - x_u), \forall u, v \in V. \tag{2}$$

Note that (2) is not a linear constrain. Hence the formulation is a Quadratic integer programming formulation of the problem. The following explains an idea for achieving a Linear relaxation of the above formulation.

The variables $x_u \in [0, 1]$ are relaxed to accept the continuous values. For y_{uv} variables, a different method is used. We look at the joint probability between x_u and x_v. To see this more carefully, fix a pair of nodes, say u, v. Define $a, b, c, d \geq 0$ and $a + b + c + d = 1$. The interpretation of these new four variables are as follows:

$a := P(x_u = 0, x_v = 0), b := P(x_u = 1, x_v = 0), c := P(x_u = 0, x_v = 1), d := P(x_u = 1, x_v = 1).$

These will imply that $x_u = b + d$, $x_v = c + d$, and $y_{uv} = b + c$.

Since the formulation is maximizing the summation of y values, we can include only the upper bound constrains as follows: $y_{uv} \leq x_u + x_v$, and $y_{uv} \leq 2 - (x_u + x_v)$.

Thus a complete LP relaxation will look like:

$$\text{Maximize} \sum_{(u,v)\in E} y_{uv}.$$

S.t.,

$$x_u \in [0,1], \qquad \forall u, \tag{3}$$
$$y_{uv} \leq x_u + x_v, \quad \forall u, v \in V, \tag{4}$$
$$y_{uv} \leq 2 - (x_u + x_v), \quad \forall u, v \in V. \tag{5}$$

Unfortunately, there is a trivial solution for this relaxation. $x_u = \frac{1}{2}, y_{uv} = 1, \forall u, v \in V$ will give give an objective value of $|E|$.

In the rest of this lecture, we will attempt to design *Stronger* constrains to help avoid these type of solutions. The idea is to extend the tuple variables from only pairs to k-tuples for arbitrary k. For example for $k = 3$, we can introduce the variables like $z^{a,b,c}_{u,v,w}$ for all $u, v, w \in V$ and $a, b, c \in \{0, 1\}$. In general, we will have $\binom{n}{k} \cdot 2^k$ new variables and $\binom{n}{k} \cdot 2^{O(k)}$ constrains.

These are called relaxation hierarchy and are due to Sherali-Adams. Note that if $k = n$, we will get exact integer solution with exponentially many variables and constrains.

Even though in the case of several other problems, Sherali-Adams relaxation hierarchies are reliable tools, the following theorem expresses the claim that it is not a good way to go for maxcut problem aiming at achieving better approximation ratio than 2.

Theorem 2 (CMM)**.** *Integrality gap of $\Omega(n)$-level relaxation for MaxCut is $2 - \epsilon$ for any $\epsilon > 0$.*

Next section will introduce *Semi-definite programming* as the right tool for this problem.

3 SDP formulation

Let us start with the following formulation and call it *QIP*:

$$\text{Maximize} \sum \frac{1 - y_{uv}}{2}, \tag{6}$$
$$x_u^2 = 1, \quad \forall u \in V, \tag{7}$$
$$y_{uv} = x_u \cdot x_v. \tag{8}$$
$$\tag{9}$$

On the other hand, these variables can be arranged in matrix-vector structure as follows:

$$Y = \begin{bmatrix} y_{11} & y_{12} & y_{13} & \dots & y_{1n} \\ y_{21} & y_{22} & y_{23} & \dots & y_{2n} \\ \dots & \dots & \dots & \dots & \dots \\ y_{n1} & y_{n2} & y_{n3} & \dots & y_{nn} \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \end{bmatrix} \cdot \begin{bmatrix} x_1 & x_2 & x_3 & \dots & x_n \end{bmatrix}.$$

Matrix notation helps us effectively summarize the constrains. Formally,

Lemma 3. *Y is symmetric P.S.D. matrix with rank 1 if and only if*

1. $y_{uu} = 1, \forall u \in V$.
2. $Y \succeq 0$.
3. *Y is symmetric.*
4. $rank(Y) = 1$.

It gives the following SDP formulation:

$$\text{Maximize } \sum \frac{1 - y_{uv}}{2},$$

S.t.,

$$y_{uu} = 1, \forall u \in V \tag{10}$$
$$y_{uv} = y_{vu} \tag{11}$$
$$Y \succeq 0 \tag{12}$$

Lemma 4. *The objective value in SDP formulation is at least as big as the objective value in QIP formulation.*

The next section shows how one can solve such SDP formulated problems.

4 Solving SDPs

A typical SDP formulation consists of a linear objective function, some simple linear constrains and a *moment constrain*. A moment constrain the constrain where a matrix is required to be symmetric and P.S.D.

Not that one moment constrain represents an infinite number of linear constrains on the original variables. Thus, the original ellipsoid algorithm will not be able to solve this problem. However, the idea is to design the SDP solver comes from the ellipsoid algorithm.

We will first review the ellipsoid algorithm:

Data: An LP formulation, δ
Result: Optimal point
Start with E_0 containing bounding box B;
while $vol(E_{i-1}) < \delta$ **do**
  Let q be the center of E_{i-1};
  If q is feasible return q;
  Find $a^T x \leq 0$ violated by q;
  Let E_i be the smallest ellipsoid containing $E_{i-1} \cap \{x | a^T x \leq 0\}$;
  $i = i + 1$;
end

Algorithm 1: Ellipsoid algorithm.

Definition 5. *A* separation oracle for moment *receives a Y as input and outputs true if Y satisfies all the constraints in Moment otherwise returns a $z \in \mathbb{R}^n, zz^T Y < 0$.*

In the following, we review the Cholesky decomposition and show how it can function as a separation oracle for the moment constrain.

In Cholesky decomposition, given a matrix M, the goal is to find the matrices L and D such that $M = L^T DL$, L is an invertible matrix, and D is a non-negative diagonal matrix. Also, Let E_{ij} denote a matrix that all diagonals are 1 except ith and jth ones. All other elements are zero except the two symmetric elements at coordinates i, j and j, i. Those elements are 1.

Note that this notation makes it easy to denote exchanged rows or columns in a matrix. Formally, $M \cdot E_{ij}$ is the matrix M but ith and jth columns are exchanged. Similarly, ith and jth rows are exchanged in $E_{ij}^T \cdot M$.

We need one more notation here: $F_{ij,\lambda}$ is the matrix with all diagonals one, and all off-diagonals zero except the one at ith row and jth column which is λ. With this notation, we can have an expression where a λ multiple of a row(column) is added to another row(column).

The following pseudo-code expresses the steps for Cholesky decomposition:

Data: A square matrix Y
Result: The matrices L and D
Start with $D = Y$, and $L = I$;
Always maintain the loop invariant $Y = L^T DL$;
for $i = 1, 2, \ldots, n$ **do**
  Find $j \geq i$ s.t. $D_{jj} \neq 0$;
  if *There exists such* j **then**
    Let $Y = L^T E_{ij}^{-1} E_{ij} D E_{ij} E_{ij}^{-1} L$;
    $E_{ij}^{-1} L$ is the new L and $E_{ij} D E_{ij}$ is the new D. ;
    If $D_{ii} < 0$ return $z = L^{-1} e_i$;
    (Note that this z is certificate for $Y \preceq 0$, i.e., $z^T Y z < 0$);
    for $j = i + 1, \ldots, n$ **do**
      Let $\lambda = -\frac{D_{ij}}{D_{ii}}$. Let $Y = (L^T F_{ji,\lambda}^{-1})(F_{ji,\lambda} D F_{ij,\lambda})(F_{ij,\lambda}^{-1} L)$;
      $L^T F_{ji,\lambda}^{-1}$ is the new L and $F_{ji,\lambda} D F_{ij,\lambda}$ is the new D.
    end
  end
  else
    Find $k \geq i$ s.t., $D_{jk} \neq 0$;
    If such k does not exist return Y is PSD;
    Else return $z = L^{-1}(e_j(-D_{jk})e_j)$
  end
end

Algorithm 2: The algorithm for Cholesky decomposition.

Note that is algorithm runs in $O(n^3)$ time complexity.