

## Lecture 15: Approximation Algorithms III

### Maximum Cut Revisited

Recall the MAX-CUT problem which asks to partition the vertices of a graph in a way that maximizes the number of edges in between the two sets. Here, we consider a generalized version of the problem. Given graph  $G(V, E)$ , we define edge weights  $w_{ij} = w_{ji}$  such that  $w_{ij} > 0$  whenever  $(i, j) \in E$  and  $w_{ij} = 0$  otherwise. Given a set  $S \subseteq V$ , the cut value obtained by partitioning the vertices into  $S$  and its complement  $\bar{S}$  is given by

$$\text{cut}(S, \bar{S}) = \sum_{i \in S, j \in \bar{S}} w_{ij} .$$

Note that we can characterize the solution to max-cut with the following quadratic integer program QIP in variables  $y_i \in \{-1, 1\}$ .

$$\begin{aligned} & \text{maximize} && \sum_{i < j} w_{ij} \frac{1 - y_i y_j}{2} \\ & \text{subject to} && y_i \in \{-1, 1\} \quad \forall i \in V \end{aligned}$$

Variables  $y_i$  indicate whether vertex  $i$  is in  $S$  or in the complement, and the QIP sums the weights of only those edges which cross the cut  $(S, \bar{S})$ . Since integer programs are NP-complete we can try to relax the constraints and obtain a continuous optimization problem. Consider the quadratic program QP obtained by replacing each integer variable  $y_i$  with an  $n = |V|$  dimensional vector  $v_i$  lying on a unit sphere.

$$\begin{aligned} & \text{maximize} && \sum_{i < j} w_{ij} \frac{1 - v_i^T v_j}{2} \\ & \text{subject to} && v_i^T v_i = 1 \quad \forall i \in V \\ & && v_i \in \mathbb{R}^n \quad \forall i \in V \end{aligned}$$

A physical interpretation for the above program is a mass spring system where the vertices are point masses embedded on an  $n$  dimensional unit sphere and the edges are springs. In physics, a spring typically obeys Hook's law which states that under a displacement vector  $x$ , the spring will exert a force in the opposite direction given by  $F = -kx$  where  $k$  is called a spring constant. Since this force depends only upon displacements we can calculate the potential energy of a spring (work done to compress it) as  $U = -\int F dx = \frac{1}{2}k||x||_2^2$ .

If the each vector  $v_i$  describes the coordinates of vertex  $i$  on the  $n$  dimensional sphere, then the spring (edge) between vertices  $i$  and  $j$  in this configuration is contracted to a length  $\|v_j - v_i\|_2$ . To achieve an optimal configuration we need to minimize its potential energy, which is equivalent to maximizing the contracted length of each spring squared.

$$\|v_j - v_i\|_2^2 = (v_j - v_i)^T(v_j - v_i) = 2(1 - v_i^T v_j)$$

This is also the quantity being maximized in the quadratic program QP. Furthermore, the edge weights  $w_{ij}$  are proportional to the spring constants with higher values corresponding to the stiffer springs.

In the optimal configuration, the stiffest springs push their endpoints further apart. Thus edges with high weights will be crossing from one half of the sphere to the other. Note that a cutting plane through the origin partitions the vertices into two disjoint sets and the edges that are part of the cut will intersect this plane. Using this intuition we can now state an approximation algorithm for MAX-CUT.

---

**Algorithm 1** Randomized MAX-CUT via Quadratic Programming

---

Solve the quadratic program QP for variables  $v_i \in \mathbb{R}^n$ .

Sample  $N(0, 1)$  i.i.d. random variables  $r_1, r_2, \dots, r_n$ .

Normalize  $r = (r_1, r_2, \dots, r_n)$  such that  $r^T r = 1$ .

$S \leftarrow \{i \in V \mid r^T v_i > 0\}$

**return**  $(S, \bar{S})$

---

In the algorithm,  $r$  is the normal of the cutting plane and  $r^T v > 0$  defines the set of points  $v$  on one side of this plane. To ensure that all cutting planes selected at random are equally likely, we verify that  $r$  is chosen uniformly at random from the surface of the sphere.

Given a point on the sphere  $x = (x_1, x_2, \dots, x_n)$ , the likelihood that  $r = x$ , computed as

$$\prod_i \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{x_i^2}{2}\right) = (2\pi)^{-\frac{n}{2}} \exp\left(-\frac{x^T x}{2}\right) = (2\pi)^{-\frac{n}{2}} \exp\left(-\frac{1}{2}\right)$$

is constant. Hence  $r$  is uniformly distributed on the surface of the sphere.

While we still need to know how to solve the quadratic program efficiently, for now we focus on how good an approximation Algorithm 1 is to MAX-CUT.

**Claim 1** *Algorithm 1 is a 0.878 approximation to MAX-CUT.*

**Proof:** Let  $\theta_{ij}$  be the angle between the solution vectors  $v_i$  and  $v_j$ . The cutting plane cuts an interval of  $2\pi$  at 2 points. The probability that one of them lands in an interval of length  $\theta_{ij}$  is  $\theta_{ij}/\pi$ . Thus, the probability that the cutting plane intersects edge  $(i, j)$  is  $\theta_{ij}/\pi$ .

If  $W$  is the cut value obtained from the algorithm, we can compute its expected value as,

$$E(W) = \sum_{i < j} w_{ij} \frac{\theta_{ij}}{\pi}$$

We want to use this to bound the value of the value of the optimal solution  $W_{OPT}$ . Let  $W_{QP}$  be the value of the solution to the quadratic program. Since a solution to the integer program is optimal and the relaxed solution will always be at least as large, we have that  $W_{QP} \geq W_{OPT}$ .

Now, noting that  $v_i^T v_j = \cos(\theta_{ij})$  we have,

$$\begin{aligned} E(W) &= \sum_{i < j} w_{ij} \frac{\theta_{ij}}{\pi} \frac{1 - \cos(\theta_{ij})}{1 - \cos(\theta_{ij})} \\ &= \sum_{i < j} w_{ij} \frac{1 - \cos(\theta_{ij})}{2} \frac{2\theta_{ij}}{\pi(1 - \cos(\theta_{ij}))} \\ &\geq \min_{0 \leq \theta \leq \pi} \frac{2\theta}{\pi(1 - \cos(\theta))} \sum_{i < j} w_{ij} \frac{1 - v_i^T v_j}{2} \\ &\geq 0.878 W_{QP} \\ &\geq 0.878 W_{OPT} \end{aligned}$$

■

We finally address the problem of solving the quadratic program by exploiting the underlying structure and formulating a more tractable problem. In this case we can write our QP as a semidefinite program SDP for which efficient polynomial time approximation algorithms are available.

A symmetric positive semidefinite matrix is one that satisfies  $A = A^T$  and for all vectors  $x$  we have  $x^T A x \geq 0$ . We write  $A \succeq 0$  for a matrix with this property. A symmetric positive semidefinite matrix can be decomposed as  $A = B^T B$  using the Cholesky factorization.

Let the matrix  $B$  have columns  $v_i$ , our QP variables. The matrix  $A = B^T B$  is then symmetric positive semidefinite and our QP can be written as a semidefinite program SDP.

$$\begin{aligned} &\text{maximize} && \sum_{i < j} w_{ij} \frac{1 - A_{ij}}{2} \\ &\text{subject to} && A \succeq 0 \\ &&& A_{ii} = 1, A_{ij} = A_{ji} \end{aligned}$$

After solving for variables  $A_{ij}$  we can use the Cholesky decomposition to obtain the  $v_i$ 's.

## Maximum Satisfiability

In the MAX-SAT problem, we are trying to find a logical assignment to a weighted Boolean formula that maximizes the total weight of clauses satisfied.

Recall that a clause  $c \in C$  is any number of true or false literals joined by *OR* (disjunction) operators. A *Boolean formula*, in conjunctive normal form (CNF), is a collection of clauses joined by *AND* (conjunction) operators. For example,

$$f = (x_1 \vee \bar{x}_2) \wedge (x_3 \vee \bar{x}_1) \wedge x_4$$

is a formula on 3 clauses and 4 literals.

Let  $n$  be the number of literals  $x_1, x_2, \dots, x_n$ . Let  $i_c$  indicate the “importance” of clause  $c$  and  $size(c)$  be the number of literals in the clause. Finally,  $W_c$  is the random variable indicating whether  $c$  is satisfied. Then

$$W = \sum_{c \in C} i_c W_c$$

$$E(W) = \sum_{c \in C} i_c P(c \text{ is satisfied})$$

---

**Algorithm 2** Simple randomized algorithm for MAX-SAT

---

Set every variable to  $T$  or  $F$  with probability  $1/2$  independently at random.

---

**Observation 1** *Algorithm 2,*

$$E(W_c) = \left(1 - \frac{1}{2^{size(c)}}\right)$$

Surprisingly, using linearity of expectation we can show that this simple randomized assignment gives a good approximation.

**Lemma 1** *Algorithm 2 is a 2-approximation.*

**Proof:**

$$\begin{aligned} E(W) &= \sum_{c \in C} \left(1 - \frac{1}{2^{size(c)}}\right) i_c \\ &\geq \frac{1}{2} \sum_{c \in C} i_c \\ &\geq \frac{1}{2} OPT \end{aligned}$$

Alternatively, we can formulate MAX-SAT as an integer program:

$$\begin{aligned} & \max_{z_c} \sum_{c \in C} i_c z_c \quad s.t. \\ \forall c, & \sum_{x_i \in +_c} y_i + \sum_{x_i \in -_c} (1 - y_i) \geq z_c \\ & \forall i, y_i, z_c \in \{0, 1\} \end{aligned}$$

We then relax this to a linear program by replacing the last constraints with

$$\begin{aligned} 0 & \leq y_i \leq 1 \\ 0 & \leq z_c \leq 1 \end{aligned}$$

We can then use randomized rounding to get approximation algorithm 3.

---

**Algorithm 3** Randomized LP rounding for MAX-SAT

---

Solve the linear program.

Let  $(y^*, z^*)$  be the optimum solution of the LP

Set  $x_i$  to  $T$  with probability  $y_i^*$  independently.

Output  $x_1, \dots, x_n$ .

---

**Lemma 2** *The probability that algorithm 3 satisfies clause  $c$  is at least  $\beta_k z_c^*$ , where  $k$  is the size of clause  $c$  and*

$$\begin{aligned} \beta_k &= 1 - \left(1 - \frac{1}{k}\right)^k \\ &> 1 - \frac{1}{e} \end{aligned}$$

**Proof:** WLOG, assume  $c = (x_1 \vee x_2 \vee \dots \vee x_k)$ .

$$\begin{aligned} P(c \text{ is satisfied}) &= 1 - \prod_{i \in c} (1 - y_i^*) \\ &\geq 1 - \left(\frac{\sum (1 - y_i^*)}{k}\right)^k \\ &\geq 1 - \left(1 - \frac{z_c^*}{k}\right)^k \\ &\geq z_c^* \left(1 - \left(1 - \frac{1}{k}\right)^k\right), \end{aligned}$$

where the last step is due to the concavity of  $(1 - \frac{1}{k})^k$  and the definition of concavity. ■

Notice that algorithm 3 performs better on *shorter* clauses, while algorithm 2 performs better on *longer* clauses. Let  $\alpha_k$  be the probability that a clause is satisfied under alg. 2 and  $\beta_k$  the probability for alg. 3. Then

k	1	2	3	...
$\alpha_k$	1/2	3/4	7/8	...
$\beta_k$	1	3/4	$1 - 8/27$	...

So the *average* of  $\alpha_k$  and  $\beta_k$  is always at least 3/4. This means that we can combine our two algorithms in a simple fashion to get a 3/4 approximation (algorithm 4).

---

**Algorithm 4** Hybrid randomized 3/4-approximation algorithm for MAX-SAT

---

```

Toss a fair coin.
if HEADS then
    run algorithm 2
else
    run algorithm 3
end if

```

---