

# 1 Randomized Rounding

Suppose we want to solve an optimization problem, such as MAX-SAT or MAX-CLIQUE. All most all optimization problems can always be converted into an instance of *integer linear program* (ILP). This means we have

An instance of an ILP problem is a set of  $m + n + mn$  constants (where  $m, n \in \mathbb{Z}^+$ )

$$c_1, \dots, c_n, d_1, \dots, d_m, a_{11}, \dots, a_{mn} \in \mathbb{Q}.$$

The problem is to choose an assignment of values of  $x_1, \dots, x_n$  that minimize (or maximize) the *objective function*

$$g(x_1, \dots, x_n) = c_1x_1 + c_2x_2 + \dots + c_nx_n$$

subject to the linear constraints

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \dots + a_{1n}x_n &\geq d_1 \\ a_{21}x_1 + a_{22}x_2 + \dots + a_{2n}x_n &\geq d_2 \\ &\dots \\ a_{m1}x_1 + a_{m2}x_2 + \dots + a_{mn}x_n &\geq d_m \end{aligned}$$

and the *additional* constraint that each  $x_i$  must be an integer. If this integer constraint is missing, then the problem is called linear programming (LP) and is polynomial-time solvable, whereas ILP is NP-hard. The first polynomial-time algorithm for linear programming was the ellipsoid algorithm, due to Kachiyan. This was improved on by Karmaker, who invented the interior point method, which is, in practice, competitive with the (worst-case exponential) simplex algorithm, due to Dantzig.

Though ILP is NP-hard, LP is solvable in polynomial time. This suggests following approach to solve an optimization problem: Convert the optimization problem to an ILP instance. Relax the integer constraint to obtain a LP instance. Solve the LP instance to obtain a possibly non-integer solution. Now, “Round” the answers to obtain a integer solution to the ILP instance. Convert this solution to obtain a solution to the optimization problem instance, and analyze how close t he obtained answer is to optimal.

**Definition 1.1.** Given a LP or ILP instance, a *feasible solution* is an assignment to the variables that satisfies all the constraints.

Observe that a feasible solution to the ILP is also a feasible solution to the LP instance obtained by relaxing ILP. Denote the optimal value of  $g$  by  $\text{OPT}_{\text{LP}}$ .

**Observation 1.2.**  $\text{OPT}_{\text{LP}} \leq \text{OPT}_{\text{ILP}}$  for minimization problems, and  $\text{OPT}_{\text{LP}} \geq \text{OPT}_{\text{ILP}}$  for maximization problems. Here LP is obtained by relaxing ILP instance.

**Definition 1.3.** The ratio  $\frac{\text{OPT}_{\text{ILP}}}{\text{OPT}_{\text{LP}}}$  is called the *integrality gap*.

## 1.1 Set-Cover

An instance of the problem Set-Cover is a collection of finite sets  $S_1, S_2, \dots, S_n$ . Define  $U = \bigcup_{i=1}^n S_i$  and label the elements as  $U = \{1, \dots, m\}$ . The goal is to find the smallest subcollection of  $S_1, S_2, \dots, S_n$  that covers  $U$ .

This problem is NP-hard. Now we describe approximation algorithms for set cover that use the above discussed approach.

Let  $f \in \mathbb{N}$  denote the maximum number of sets in which any element of  $U$  can appear. For vertex cover,  $f = 2$  (and this is also the minimum number of sets in which any element can appear; each element of  $U$  is an edge, and each set is a node.)

We construct an ILP as follows. Minimize

$$g(y_1, \dots, y_n) = \sum_{i=1}^n y_i,$$

where each  $y_i = 1$  if  $S_i$  is in the subcollection, and  $y_i = 0$  otherwise. Let  $T_i$  denote the collection of sets in which “ $i$ ” appears (where  $i$  is an element of  $U$ ). Then to express the fact that we want each element of  $U$  to be covered by the subcollection, add the linear constraint

$$(\forall i \in U) \sum_{j \in T_i} y_j \geq 1 \text{ and } y_i \in \{0, 1\}$$

Let  $\text{OPT}_{\text{ILP}}$  denote the optimal value of  $g$ .

Now solve the relaxed LP problem in which we relax the constraint  $y_i \in \{0, 1\}$  to

$$0 \leq y_i \leq 1$$

Let

$$y^* = \langle y_1^*, \dots, y_n^* \rangle$$

be an optimal solution for the LP instance.

Form the set cover as follows.  $S_i$  belongs to the cover if and only if  $y_i^* \geq \frac{1}{f}$ .

We need to answer two questions:

1. Is this really a set cover?
2. How close is this to the optimal solution?

**Claim 1.** *The algorithm produces a set cover.*

*Proof.* Consider an element  $e \in U$ . It can appear in at most  $f$  sets. We call them

$$S_{e_1}, S_{e_2}, \dots, S_{e_l}$$

where  $l \leq f$ .

Corresponding to the element  $e$ , we have the LP following constraint

$$y_{e_1} + y_{e_2} + \dots + y_{e_l} \geq 1.$$

Since  $l \leq f$ , at least one of

$$y_{e_j}^* \geq \frac{1}{f}.$$

So  $e$  is covered. □

**Claim 2.** *The algorithm achieves an approximation ratio of  $f$ .*

*Proof.* Let

$$y^* = \langle y_1^*, \dots, y_n^* \rangle$$

be the optimal solution for the LP. Suppose

$$\hat{y} = \langle \hat{y}_1, \dots, \hat{y}_n \rangle$$

is the solution of the ILP instance that is obtained by rounding.

$$\hat{y}_i \leq f y_i^*$$

because  $\hat{y}_i$  is made to be 1 only if  $y_i^* \geq \frac{1}{f}$ , meaning that we are multiplying  $y_i^*$  by at most  $f$ . This implies

$$\sum_{i=1}^n \hat{y}_i \leq f \sum_{i=1}^n y_i^* \leq f \cdot \text{OPT}_{\text{ILP}}.$$

□

**Corollary 1.4.** *This gives a 2-approximation algorithm for vertex cover.*

But if  $f$  is large, then the approximation ratio is bad.

## 1.2 Improved Algorithm for Set-Cover

Place the set  $S_i$  into the cover with probability  $y_i^*$ . This is called *randomized rounding*. We must answer the previous two questions again.

Let  $X_i$  denote the random variable

$$X_i = \begin{cases} 1, & \text{if } S_i \text{ is in the cover;} \\ 0, & \text{otherwise.} \end{cases}$$

Note that

$$\Pr[X_i = 1] = y_i^*$$

meaning

$$\mathbb{E}[X_i] = y_i^*.$$

Let  $X = \sum X_i$ . This denotes the number of sets produced by the algorithm. Then by linearity of expectation

$$\mathbb{E}[X] = \sum y_i^* = \text{OPT}_{\text{LP}} \leq \text{OPT}_{\text{ILP}}.$$

But if this is a strict inequality, then this cannot be a cover. If it is an equality, then this is an exact solution, but this problem is NP-hard. So know that sometimes we will get something too small to be a cover.

Look at the probability that an element  $e$  is not covered by the above algorithm. Say  $e$  appears in  $l$  sets

$$S_{e_1}, S_{e_2}, \dots, S_{e_l}$$

$S_{e_j}$  belongs to the cover with probability  $y_{e_j}^*$ . So

$$\Pr[e \text{ is not covered}] \leq \prod_{j=1}^l (1 - y_{e_j}^*).$$

Recall that there is a constraint

$$\sum_{j=1}^l y_{e_j} \geq 1.$$

So

$$\begin{aligned} \sum_{j=1}^l y_{e_j}^* \geq 1 &\implies \sum_{j=1}^l (1 - y_{e_j}^*) \leq l - 1 \\ &\implies \sum_{j=1}^l (1 - y_{e_j}^*) \leq l - 1. \end{aligned}$$

In general, given any  $a_1, \dots, a_l, t \in \mathbb{R}^+$  such that  $\sum_{i=1}^l a_i \leq t$ , then  $\prod_{i=1}^l a_i$  is maximal when each of the values is exactly the same (i.e., when  $a_i = \frac{t}{l}$ ). So

$$\prod_{j=1}^l (1 - y_{e_j}^*) \leq \left(\frac{l-1}{l}\right)^l \leq \frac{1}{e}.$$

Knowing this, we repeat the algorithm  $4 \log n$  times and take the union of all the covers found. Then

$$(\forall i \in U) \Pr[i \text{ is not covered}] \leq \frac{1}{e^{4 \log n}} \leq \frac{1}{2^{4 \log n}} = \frac{1}{n^4}.$$

By the union bound,

$$\Pr[(\exists i \in U) i \text{ is not covered}] \leq \frac{n}{n^4} = \frac{1}{n^3}. \quad (1.1)$$

What is the cost of the cover?

$$\mathbb{E}[\text{number of sets in the cover}] \leq 4 \log n \cdot \text{OPT}_{\text{ILP}}.$$

So by Markov's inequality

$$\Pr[\text{number of sets in the cover} \geq 16 \log n \cdot \text{OPT}_{\text{ILP}}] \leq \frac{1}{4}. \quad (1.2)$$

Then by (1.1) and (1.2),

$$\Pr[\text{algorithm produces a cover and size of cover} \leq 16 \log n \cdot \text{OPT}_{\text{ILP}}] \geq \frac{1}{2}.$$