Randomized Rounding : Set Cover and Independent Set[1]

- In this lecture, we introduce a power technique in algorithm design in general : ***randomization***. More precisely, *randomized rounding* takes an feasible LP solution, interprets the fractional solution $\mathbf{x}_i$ as the chance or marginal probability that the variable i is set to 1 in the optimum solution, and then designs a *randomized* algorithm which produces a (distribution over) feasible solution. Since the solution produced by the algorithm can (and most often will) be different each time it is called, instead of looking at the cost/value of a solution, one talks about the *expected* cost/value of a solution.

  Definition 1. *For a minimization problem, an α-approximate randomized algorithm returns a feasible solution S of* **expected** *cost* $\mathbf{Exp}[c(S)] \leq \alpha OPT$*. For a maximization problem, an α-approximate randomized algorithm returns a feasible solution S of* **expected** *value* $\mathbf{Exp}[v(S)] \geq OPT/\alpha$.

  As we go along, we will use facts from probability theory, mostly regarding the concentration of random variables around their means.

- ***Canonical Example : Set Cover.*** Recall the set cover problem. We have a set family $\mathcal{S} := (U, (S_1, \ldots, S_m))$ where S_j is a subset of the universe U. Each set S_j has a non-negative cost $c(S_j)$. The objective is to select a family of these subsets of minimum cost whose union is the universe. Following is an LP relaxation for the problem where x_j is supposed to denote whether set j is picked or not.

$$\mathsf{lp}(\mathcal{S}) := \text{minimize} \quad \sum_{j=1}^{m} c(S_j) x_j \qquad \text{(Set Cover LP)}$$

$$\sum_{j: i \in S_j} x_j \geq 1, \qquad \forall i \in U \qquad (1)$$

$$0 \leq x_j \leq 1, \ \ \forall j = 1, \ldots, m \qquad (2)$$

  If the $x_j \in \{0,1\}$, then the above captures the set cover problem exactly. When $x_j \in [0,1]$, one *interpretation* of this solution can be the "chance" that set j is picked in an optimal solution. To be more precise and useful, if we ourselves could design a *randomized* algorithm which always returns a set cover *and* the probability of set j being present in the solution is $= x_j$, then such a distribution is perhaps what the LP is prescribing. And indeed, by linearity of expectation, the expected cost of such a solution is going to be $\leq \mathsf{lp}(\mathcal{S})$. Make sure you see this before proceeding.

  Of course, if we can find a solution whose expected cost is $\leq \mathsf{lp}(\mathcal{S}) \leq \mathtt{opt}$, then we would be exactly solving set cover. So the above is not possible unless P=NP. However the above interpretation is useful, and the x_j's can be used to design an *approximation algorithm*. Here is it without further ado.

[1] *Lecture notes by Deeparnab Chakrabarty. Last modified : 14th Jan, 2022*
These have not gone through scrutiny and may contain errors. If you find any, or have any other comments, please email me at deeparnab@dartmouth.edu. Highly appreciated!

1: **procedure** SET COVER RANDOMIZED ROUNDING($\mathcal{S} = (U, (S_j, c(S_j) : j \in [m]))$):
2: Solve (Set Cover LP) to obtain $x_j \in [0, 1]$ for $1 \leq j \leq m$.
3: Sample each set j **independently** with probability $p_j := \min(1, \ln n \cdot x_j)$.
4: For each element i not covered in Line 3, pick the minimum cost set $S(i) := \min_{S:i \in S} c(S)$ which contains i.

Line 3 is the randomized step and will change from run-to-run. If we denote the indices of the sets picked in Line 3 as R, then note that $\bigcup_{j \in R} S_j$ may or may not be U. In order to *fix* this, in Line 4 one goes over yet uncovered elements and picks the minimum cost set containing that element. Another point of note : in Line 3, the sampling probability is not x_j but something which is "boosted up". This boosting is by hindsight, as hopefully will be clear from the analysis below.

Theorem 1. SET COVER RANDOMIZED ROUNDING is a $(1 + \ln n)$-approximate randomized algorithm.

- *Proof.* By design, due to Line 4, the algorithm returns a feasible solution with probability 1. We need to argue about the *expected* cost of this solution. We begin with an easy observation

Claim 1. *For any element i, we have $c(S(i)) \leq \mathsf{lp}(\mathcal{S})$.*

Proof. Fix an element i and consider the contribution of only the sets containing i to the LP solution. We get $\mathsf{lp} \geq \sum_{j:i \in S_j} c(S_j) x_j \geq c(S(i)) \sum_{j:i \in S_j} x_j \geq c(S(i))$, where the first inequality followed since $S(i)$ is the cheapest set containing i, and the second followed from (1). □

Let alg be the *random* variable indicating the cost of the solution picked by the algorithm. We write $\mathsf{alg} = \mathsf{alg}_1 + \mathsf{alg}_2$ where alg_1 is the random variable indicating the costs of the sets picked in Line 3, and alg_2 is the random variable indicating the costs of the sets picked in Line 4. Note that alg_2 is a random variable as well, although Line 4 has no randomness in it. This is because it depends on the randomness in the step above. Indeed, alg_1 and alg_2 are **not** independent random variables. Nevertheless, the beautiful linearity of expectation[2] result lets us assert

$$\mathbf{Exp}[\mathsf{alg}] = \mathbf{Exp}[\mathsf{alg}_1] + \mathbf{Exp}[\mathsf{alg}_2]$$

We now proceed and bound the two expectations in the RHS. Indeed, the theorem then follows from Claim 2 and Claim 3.

Claim 2. $\mathbf{Exp}[\mathsf{alg}_1] \leq \ln n \cdot \mathsf{lp}$.

Proof. We first write $\mathsf{alg}_1 = \sum_{j=1}^{m} c(S_j) X_j$ where X_j is the *indicator random variable* whether set S_j is picked in Line 3. Once again, linearity of expectation states $\mathbf{Exp}[\mathsf{alg}_1] = \sum_{j=1}^{m} c(S_j) \mathbf{Exp}[X_j]$, and $\mathbf{Exp}[X_j] = p_j \leq \ln n \cdot x_j$, thus completing the proof. □

Claim 3. $\mathbf{Exp}[\mathsf{alg}_2] \leq \mathsf{lp}$.

[2] For *any* two random variables X, Y, we have $\mathbf{Exp}[X + Y] = \mathbf{Exp}[X] + \mathbf{Exp}[Y]$.

Proof. Similar to the above claim, we now write alg_2 also as a sum of random variables thus: $\mathsf{alg}_2 = \sum_{i\in U} c(S(i)) \cdot Y_i$ where Y_i is the indicator random variable whether element i is left uncovered in Line 3. We soon show that $\mathbf{Exp}[Y_i] \le \frac{1}{n}$. This would imply $\mathsf{alg}_2 \le \frac{1}{n}\sum_{i\in U} c(S(i)) \le \mathsf{lp}$ where the last inequality follows from Claim 1.

Fix an element i. We note that $\mathbf{Exp}[Y_i]$ is simply the probability i is not covered in Line 3. Observe that this probability is precisely $\prod_{j:i\in S_j}(1-p_j)$ This is where the independence in Line 3 is used. So we may assume $p_j \neq 1$, and therefore $p_j = \ln n \cdot x_j$ for all such sets. Which implies

$$\mathbf{Exp}[Y_i] = \prod_{j:i\in S_j}(1-\ln n\cdot x_j) \;\le\; \prod_{j:i\in S_j} e^{-\ln n\cdot x_j} \;=\; n^{-\sum_{j:i\in S_j} x_j} \le \frac{1}{n}$$

where the last inequality follows from (1). □

Exercise: ☕☕

Consider the Max-Coverage *problem where one has to pick k sets to maximize the number of elements covered. Describe a LP relaxation for the problem, and a randomized rounding algorithm that obtains an $(1-\frac{1}{e})$-approximation.*

Exercise: ☕☕

Consider the multi-set-multi-cover problem where the input is same as the set cover problem, but now every element i has a demand $d(i)$ as to how many times it needs to be covered. More precisely, you are allowed to choose a set S_j multiple times, but if you choose it k_j times you pay cost $k_j c(S_j)$. For every element, you should have $\sum_{j:i\in S_j} k_j \ge d(i)$. Describe an LP relaxation and an $O(\log n)$ randomized rounding algorithm.

- ***Independent Set.*** We now describe a randomized algorithm for a *maximization* problem, the independent set problem in graphs. In this problem we are given an undirected graph $G = (V, E)$ with non-negative weights w_v on vertices. The objective is to pick an independent set $I \subseteq V$ with as large a weight as possible. Recall, I is independent if no edge (u, v) has both endpoints in I. The approximation factor obtained isn't great, but the main point is to introduce the technique of "alteration". In the problem sets, we may explore a better factor.

- *LP Relaxation.*Here is an LP relaxation for the problem.

$$\mathsf{lp}(G,w) := \text{maximize} \quad \sum_{v\in V} w_v x_v \qquad \text{(IS LP)}$$

$$x_u + x_v \le 1, \qquad \forall (u,v)\in E \qquad (3)$$

$$0 \le x_u \le 1,\ \forall u \in V \qquad (4)$$

- *Randomized Rounding.* We now describe an algorithm which is a $2\sqrt{m}$-approximation, where m is the number of edges. Let $W := \max_{v\in V} w_v$. Note that there is a trivial algorithm whose value is W: return the singleton vertex with maximum weight. This benchmark will be used.

```
1: procedure IS RAND ROUNDING(𝒮 = (U, (S_j, c(S_j) : j ∈ [m]))):
2:     Solve (IS LP) to obtain x_v ∈ [0,1] for v ∈ V with value lp.
3:     if lp ≤ 2√m · W then:
4:         return single vertex of maximum weight W.  ▷ By design, a 2√m-appx.
5:     Sample independently vertex v with probability p_v := x_v/√m to get a set I.  ▷ At this pointI
   may not be independent.
6:     For each edge (u,v) with u and v in I, delete both from I.
7:     ▷ It would've sufficed to delete any one, but as we show this overzealousness doesn't
   hurt. After all "bad" edges are thus fixed, I is indeed independent.
8:     return I.
```

Theorem 2. IS RAND ROUNDING returns a independent set I with $\mathbf{Exp}[w(I)] \leq \frac{\mathsf{lp}}{2\sqrt{m}}$.

- *Proof.* Once again, it is clear that the solution returned is an independent set. Also note that if $\mathsf{lp} \leq 2\sqrt{m} \cdot W$, the max weight singleton vertex has weight $\geq \mathsf{lp}/2\sqrt{m}$. So we may assume otherwise, that is, $W \leq \frac{\mathsf{lp}}{2\sqrt{m}}$.

  Let I_1 be the set of vertices picked after Line 5, and let D be the subset of vertices deleted from I_1 in Line 6. Thus, $I = I_1 \setminus D$. By linearity of expectation, $\mathbf{Exp}[w(I)] = \mathbf{Exp}[w(I_1)] - \mathbf{Exp}[w(D)]$

  Let X_v be the indicator random variable that $v \in I_1$, and for an edge $(u,v) \in E$, let Z_{uv} be the indicator random variable that both u and v are in I_1. Now note that

$$w(I_1) = \sum_{v \in V} w_v X_v \quad \text{and} \quad w(D) \leq \sum_{(u,v) \in E} (w_u + w_v) \cdot Z_{uv}$$

  Note that we have an inequality for $w(D)$, since we may possibly be double counting in the RHS. For example, if there are two edges (u,v) and (u,z) in E, and if u, v, z are all in I_1, then we should count $w_u + w_v + w_z$ in $w(D)$, but the RHS double counts w_u.

  Next, notice that $\mathbf{Exp}[X_v] = \frac{x_v}{\sqrt{m}}$ and $\mathbf{Exp}[Z_{uv}] = \frac{x_u x_v}{m}$. We can now upper bound this expectation as follows

$$\mathbf{Exp}[Z_{uv}] = \frac{x_u x_v}{m} \underbrace{\leq}_{\text{AM-GM}} \frac{x_u^2 + x_v^2}{2m} \underbrace{\leq}_{\text{since } x_u, x_v \leq 1} \frac{x_u + x_v}{2m} \underbrace{\leq}_{\text{by(3)}} \frac{1}{2m}$$

  Substituting all of this above, we get

$$\mathbf{Exp}[w(D)] \leq \frac{1}{2m} \sum_{(u,v) \in E} (w_u + w_v) = \sum_{v \in V} \frac{\deg(v)}{2m} w_v \;\leq\; \max_{v \in V} w_v = W$$

  where the last inequality follows since $\sum_{v \in V} \deg(v) = 2m$. Thus, the LHS in the last inequality is a (weighted) average of all the weights, which is at most the maximum weight. Since $W \leq \frac{\mathsf{lp}}{2\sqrt{m}}$, we get $\mathbf{Exp}[w(D)] \leq \frac{\mathsf{lp}}{2\sqrt{m}}$. And so,

$$\mathbf{Exp}[w(I)] = \mathbf{Exp}[w(I_1)] - \mathbf{Exp}[w(D)] \geq \frac{\mathsf{lp}}{\sqrt{m}} - \frac{\mathsf{lp}}{2\sqrt{m}} = \frac{\mathsf{lp}}{2\sqrt{m}}$$ □

Exercise: ☕☕*Suppose* d *is the* maximum *degree of the graph* $G = (V, E)$. *Modify* IS RAND ROUNDING *and its analysis to describe an algorithm which returns a* $4\mathrm{d}$*-approximation. That is, it returns an independent set* I *with* $\mathbf{Exp}[w(I)] \geq \frac{\mathsf{lp}}{4d}$. *Indeed, if designed correctly, your algorithm should also work when* G *is a* hypergraph.