Randomized Algorithms

Randomized Rounding

- Brief Introduction to Linear Programming and Its Usage in Combinatorial Optimization
- Randomized Rounding for Cut Problems
- **Randomized Rounding for Covering Problems**
- Randomized Rounding for Satisfiability Problems
- Randomized Rounding and Semi-definite Programming

Approximate Sampling and Counting

- ...

(Randomized) Rounding

- A (minimization) combinatorial problem $\Pi \Leftrightarrow$ an ILP
- Let $\bar{\mathbf{y}}$ be an optimal solution to the ILP
- Relax ILP to get an LP; let $\mathbf{y}^*$ be an optimal solution to the LP
- Then,

$$\textsc{opt}(\Pi) = \mathsf{cost}(\bar{\mathbf{y}}) \geq \mathsf{cost}(\mathbf{y}^*)$$

  (If Π is maximization, reverse the inequality!)
- Carefully "round" $\mathbf{y}^*$ (rational) to get a feasible solution $\mathbf{y}^A$ (integral) to the ILP, such that $\mathbf{y}^A$ is *not too bad*, say $\mathsf{cost}(\mathbf{y}^A) \leq \alpha\, \mathsf{cost}(\mathbf{y}^*)$
- Conclude that $\mathsf{cost}(\mathbf{y}^A) \leq \alpha \cdot \textsc{opt}(\Pi)$
- Thus, we get an α-approximation algorithm for Π
- If $\alpha = 1$, then we have solved Π exactly!

An Integer Linear Program for Set Cover

Definition (Set-Cover Problem)

Inputs: a collection $\mathcal{S} = \{S_1, \dots, S_n\}$ of subsets of $[m] = \{1, \dots, m\}$, where S_j is of weight $w_j \in \mathbb{Z}^+$.
Objective: find a sub-collection $\mathcal{C} = \{S_i \mid i \in J\}$ with least total weight such that $\bigcup_{i \in J} S_i = [m]$.

ILP for Set Cover

$$\begin{array}{rl} \min & w_1 x_1 + \cdots + w_n x_n \\ \text{subject to} & \displaystyle\sum_{j : S_j \ni i} x_j \geq 1, \quad \forall i \in [m], \\ & x_j \in \{0, 1\}, \quad \forall j \in [n]. \end{array} \tag{1}$$

Let $\bar{\mathbf{x}}$ be an optimal solution to this ILP.

Relaxation

The relaxation of the ILP is the following LP:

$$
\begin{array}{rl}
\min & w_1x_1 + \cdots + w_nx_n \\
\text{subject to} & \displaystyle\sum_{j : S_j \ni i} x_j \geq 1, \quad \forall i \in [m], \\
& 0 \leq x_j \leq 1 \quad \forall j \in [n].
\end{array} \tag{2}
$$

Let $\mathbf{x}^*$ be an optimal solution to this LP.

First Attempt at Randomized Rounding

- Want: a feasible solution $\mathbf{x}^A$ which is *not too far* from $\mathbf{x}^*$ on average.
- Make sense to try:

$$\text{Prob}[x_j^A = 1] = x_j^*.$$

- Solution quality:

$$\mathsf{E}[\text{cost}(\mathbf{x}^A)] = \sum_{j=1}^{n} w_j x_j^* = \text{cost}(\mathbf{x}^*) \leq \text{cost}(\bar{\mathbf{x}}) = \text{OPT}.$$

- Feasibility? Consider an arbitrary constraint $x_{j_1} + \cdots + x_{j_k} \geq 1$.
- The probability that this constraint is not satisfied by $\mathbf{x}^A$ is

$$(1 - x_{j_1}^*) \dots (1 - x_{j_k}^*) \leq \left(\frac{k - (x_{j_1}^* + \cdots + x_{j_k}^*)}{k}\right)^k \leq \left(1 - \frac{1}{k}\right)^k \leq \frac{1}{e}.$$

There are m constraints; thus, $\text{Prob}[\mathbf{x}^A \text{ is not feasible}] \leq m/e$.

First attempt doesn't quite work!

Second Attempt at Randomized Rounding

- Should round x_j to 1 with higher probability. Let t be a parameter determined later.

$$\text{Prob}[x_j^A = 0] = (1 - x_j^*)^t$$

(This is equivalent to running the first strategy independently t rounds, and set $x_j^A = 0$ only when $x_j^A = 0$ in all rounds.)

- Solution Quality

$$\text{E}[\text{cost}(\mathbf{x}^A)] \leq t \cdot \text{OPT}.$$

- Feasibility? Prob$[\mathbf{x}^A$ does not satisfy any given constraint$] \leq (1/e)^t$.
- Thus, Prob$[\mathbf{x}^A$ is not feasible$] \leq m(1/e)^t$.

Finishing Up

- Markov inequality gives

$$\text{Prob}[\text{cost}(\mathbf{x}^A) > \rho \cdot \text{OPT}] < \frac{\text{E}[\text{cost}(\mathbf{x}^A)]}{\rho \cdot \text{OPT}} \leq \frac{t \cdot \text{OPT}}{\rho \cdot \text{OPT}} = \frac{t}{\rho}.$$

- Consequently,

$$\text{Prob}[\mathbf{x}^A \text{ is feasible and } \ \text{cost}(\mathbf{x}^A) \leq \rho \cdot \text{OPT}] \geq 1 - m(1/e)^t - \frac{t}{\rho}.$$

  We can pick $t = \theta(\lg m)$ and $\rho = 4t$ so that $1 - m(1/e)^t - \frac{t}{\rho} \geq \frac{1}{2}$.
- To boost the confidence up (say, to $1 - 1/2^m$), run the algorithm m times!
- Basically, we got a $\Theta(\log m)$-approximation algorithm for weighted set cover.
- Asymptotically, we cannot approximate better than that!