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

- ...

CNF Formulas

- Conjunctive Normal Form (CNF) formulas:

$$\varphi = \underbrace{(x_1 \vee \bar{x}_2)}_{\text{Clause } 1} \wedge \underbrace{(x_1 \vee x_3 \vee \bar{x}_4 \vee x_6)}_{\text{Clause } 2} \wedge \underbrace{(\bar{x}_2 \vee \bar{x}_3 \vee x_4)}_{\text{Clause } 3} \wedge \underbrace{(\bar{x}_5)}_{\text{Clause } 4}$$

- Literals: $\bar{x}_2$, x_4, etc.
- Truth assignment: $a : \{x_1, \dots, x_n\} \to \{\text{TRUE}, \text{ FALSE}\}$
- For integers $k \geq 2$, a k-CNF formula is a CNF formula in which each clause is of size *at most* k,
- an Ek-CNF formula is a CNF formula in which each clause is of size *exactly* k.

Satisfiability Problems

- MAX-SAT: given a CNF formula φ, find a truth assignment satisfying as many clauses as possible
- MAX-kSAT: given a k-CNF formula φ, find a truth assignment satisfying as many clauses as possible
- MAX-EkSAT: given an Ek-CNF formula φ, find a truth assignment satisfying as many clauses as possible
- Weighted-Xsat: X $\in \{\emptyset, k$ E$k\}$ – clause j has *weight* w_j, find a truth assignment satisfying clauses with largest total weight

These are very fundamental problems in optimization, with many applications (in security, software verification, etc.)

The Arithmetic-Geometric Means Inequality

Theorem (Arithmetic-geometric means inequality)

For any non-negative numbers $a_1, \dots, a_n$, we have

$$\frac{a_1 + \cdots + a_n}{n} \geq (a_1 \cdots a_n)^{1/n}. \tag{1}$$

There is also the stronger weighted version. Let $w_1, \dots, w_n$ be positive real numbers where $w_1 + \cdots + w_n = 1$, then

$$w_1 a_1 + \cdots + w_n a_n \geq a_1^{w_1} \cdots a_n^{w_n}. \tag{2}$$

Equality holds iff all a_i are equal.

The Cauchy-Schwarz Inequality

Theorem (Cauchy-Schwarz inequality)

Let $a_1, \dots, a_n$ *and* $b_1, \dots, b_n$ *be non-negative real numbers. Then,*

$$\left(\sum_{i=1}^n a_i b_i\right)^2 \leq \left(\sum_{i=1}^n a_i^2\right)\left(\sum_{i=1}^n b_i^2\right). \tag{3}$$

Jensen Inequality

Theorem (Jensen inequality)

Let $f(x)$ be a convex function on an interval (a, b). Let $x_1, \dots, x_n$ be points in (a, b), and $w_1, \dots, w_n$ be non-negative weights such that $w_1 + \cdots + w_n = 1$. Then,

$$f\left(\sum_{i=1}^{n} w_i x_i\right) \leq \sum_{i=1}^{n} w_i f(x_i). \tag{4}$$

If f is strictly convex and if all weights are positive, then equality holds iff all x_i are equal. When f is concave, the inequality is reversed.

Convex test: non-negative second derivative.

Concave test: non-positive second derivative.

The "Naive" Randomized Algorithm for MAX-E3-SAT

The Algorithm

Assign each variable to TRUE/FALSE with probability $1/2$

- Let X_C be the random variable indicating if clause C is satisfied
- Then, $\text{Prob}[X_C = 1] = 7/8$
- Let S_φ be the number of satisfied clauses. Then,

$$\mathsf{E}[S_\varphi] = \mathsf{E}\left[\sum_C X_C\right] = \sum_C \mathsf{E}[X_C] = 7m/8 \geq \frac{\text{OPT}}{8/7}$$

(m is the number of clauses)

- So this is a randomized approximation algorithm with ratio $8/7$

Derandomization Using Conditional Expectation

- Derandomization is to turn a randomized algorithm into a deterministic algorithm
- By conditional expectation

$$\mathsf{E}[S_\varphi] = \frac{1}{2}\mathsf{E}[S_\varphi \mid x_1 = \text{ TRUE}] + \frac{1}{2}\mathsf{E}[S_\varphi \mid x_1 = \text{ FALSE}]$$

- Both $\mathsf{E}[S_\varphi \mid x_1 = \text{ TRUE}]$ and $\mathsf{E}[S_\varphi \mid x_1 = \text{ FALSE}]$ can be computed in polynomial time
- Suppose $\mathsf{E}[S_\varphi \mid x_1 = \text{ TRUE}] \geq \mathsf{E}[S_\varphi \mid x_1 = \text{ FALSE}]$, then

$$\mathsf{E}[S_\varphi \mid x_1 = \text{ TRUE}] \geq \mathsf{E}[S_\varphi] \geq 7m/8$$

- Set $x_1 = \text{TRUE}$, let φ' be φ with c clauses containing x_1 removed, and all instances of $x_1, \bar{x}_1$ removed.
- Recursively find value for x_2

The "Naive" Randomized Algorithm for MAX-SAT

The Algorithm

Assign each variable to TRUE/FALSE with probability $1/2$

- Let X_j be the random variable indicating if clause C_j is satisfied
- If C_j has l_j literals, then $\text{Prob}[X_j = 1] = 1 - 1/2^{l_j}$
- Let S_φ be the total weight of satisfied clauses. Then,

$$\mathsf{E}[S_\phi] = \sum_{j=1}^{m} w_j(1 - (1/2)^{l_j}) \geq \frac{1}{2}\sum_{j=1}^{m} w_j \geq \frac{1}{2}\text{OPT}(\phi).$$

- So this is a randomized approximation algorithm with ratio 2, quite a bit worse than $8/7$.
- The algorithm can be derandomized with conditional expectation

Randomized Algorithm for MAX-SAT with One Biased Coin

The One-Biased-Coin Algorithm

Assign each variable to TRUE/FALSE with probability q (to be determined).

- Let n_j and p_j be the number of negated variables and non-negated variables in clause C_j, then

$$\mathsf{E}[S_\phi] = \sum_{j=1}^{m} w_j(1 - q^{n_j}(1-q)^{p_j}).$$

- In the naive algorithm, a clause with $l_j = 1$ is troublesome. We will try to deal with small clauses.

The Analysis

- If (x_i) is a clause but $(\bar{x}_i)$ is not: change variable $y_i = x_i$
- If $(\bar{x}_i)$ is a clause but (x_i) is not: change variable $y_i = \bar{x}_i$
- If (x_i) appears many times as clauses, replace them with one clause (x_i) whose weight is the sum
- If $(\bar{x}_i)$ appears many times as clauses, replace them with one clause $(\bar{x}_i)$ whose weight is the sum
- After this is done:
  - each singleton clause (x_i) appears at most once
  - each singleton clause $(\bar{x}_i)$ appears at most once
  - if $(\bar{x}_i)$ is a singleton, then so is (x_i).

The Analysis

- Let $N = \{j \mid C_j = \{\bar{x}_i\}, \text{ for some } i\}$. Then,

$$\text{OPT}(\phi) \leq \sum_{j=1}^{m} w_j - \sum_{j \in N} w_j.$$

- If $j \in N$, $(1 - q^{n_j}(1-q)^{p_j}) = (1-q)$.
- If $j \notin N$, then either $p_j \geq 1$ or $n_j \geq 2$, and thus

$$(1 - q^{n_j}(1-q)^{p_j}) \geq 1 - \max\{1-q, q^2\}.$$

Choose q such that $1 - q = q^2$, i.e. $q \approx 0.618$, we have for $j \notin N$

$$(1 - q^{n_j}(1-q)^{p_j}) \geq 1 - (1-q) = q.$$

- Finally,

$$\mathsf{E}[S_\phi] = \sum_{j \notin N} w_j(1 - q^{n_j}(1-q)^{p_j}) + \sum_{j \in N} w_j(1-q) \geq q \cdot \text{OPT}(\phi).$$

Conclusions

- We have a $1/q \approx 1/0.618 \approx 1.62$-approximation algorithm
- This can be derandomized too.
- To make use of the structure of the formula φ, perhaps it makes sense to use n biased coins:

$$\text{Prob}[x_i = \text{ TRUE}] = q_i.$$

- But, how to choose the q_i?

Randomized Rounding for MAX-SAT

The Integer Program

Think: (a) $y_i = 1$ iff $x_i = \text{TRUE}$; (b) $z_j = 1$ iff C_j is satisfied.

$$
\begin{array}{rll}
\max & w_1 z_1 + \cdots + w_m z_n & \\
\text{subject to} & \displaystyle\sum_{i : x_i \in C_j} y_i + \sum_{i : \bar{x}_i \in C_j} (1 - y_i) \geq z_j, & \forall j \in [m], \\
& y_i, z_j \in \{0, 1\}, & \forall i \in [n], j \in [m]
\end{array}
$$

The Relaxation

$$
\begin{array}{rll}
\max & w_1 z_1 + \cdots + w_n z_n & \\
\text{subject to} & \displaystyle\sum_{i : x_i \in C_j} y_i + \sum_{i : \bar{x}_i \in C_j} (1 - y_i) \geq z_j, & \forall j \in [m], \\
& 0 \leq y_i \leq 1 & \forall i \in [n], \\
& 0 \leq z_j \leq 1 & \forall j \in [m].
\end{array}
$$

Let $(\mathbf{y}^*, \mathbf{z}^*)$ be an optimal solution to the LP.

Randomized Rounding with Many Biased Coins

Set $x_i = \text{TRUE}$ with probability y_i^*.

$$
\begin{aligned}
\mathsf{E}[S_\phi] &= \sum_{j=1}^m w_j \left(1 - \prod_{i:x_i\in C_j}(1-y_i^*)\prod_{i:\bar{x}_i\in C_j} y_i^*\right) \\
&\geq \sum_{j=1}^m w_j \left(1 - \left[\frac{\sum_{i:x_i\in C_j}(1-y_i^*) + \sum_{i:\bar{x}_i\in C_j} y_i^*}{l_j}\right]^{l_j}\right) \\
&= \sum_{j=1}^m w_j \left(1 - \left[\frac{l_j - \left(\sum_{i:x_i\in C_j} y_i^* + \sum_{i:\bar{x}_i\in C_j}(1-y_i^*)\right)}{l_j}\right]^{l_j}\right)
\end{aligned}
$$

Randomized Rounding with Many Biased Coins

The function $f(z) = (1 - (1 - z/l_j)^{l_j}$ is concave when $z \in [0, 1]$. Thus,

$$
\begin{aligned}
\mathsf{E}[S_\phi] &\geq \sum_{j=1}^{m} w_j \left(1 - \left[1 - \frac{z_j^*}{l_j}\right]^{l_j}\right) \\
&\geq \sum_{j=1}^{m} w_j \left(1 - \left[1 - \frac{1}{l_j}\right]^{l_j}\right) z_j^* \\
&\geq \min_j \left(1 - \left[1 - \frac{1}{l_j}\right]^{l_j}\right) \sum_{j=1}^{m} w_j z_j^* \\
&\geq \left(1 - \frac{1}{e}\right) \mathrm{OPT}(\phi).
\end{aligned}
$$

Theorem

The LP-based randomized rounding algorithm above has approximation ratio $e/(e-1) \approx 1.58$.

The "Best-of-Two" Algorithm

- The LP-based algorithm works well if all l_j are small. For example, if $l_j \leq 2$ then
$$\left(1 - \left[1 - \frac{1}{l_j}\right]^{l_j}\right) \geq \frac{3}{4}$$
which gives a $\frac{4}{3}$-approximation.
- Similarly, the naive algorithm works well if all l_j are large.
- Combination: run both and output the better solution.

$$\begin{aligned}
\mathsf{E}[\max\{S_\phi^1, S_\phi^2\}] &\geq \mathsf{E}[(S_\phi^1 + S_\phi^2)/2] \\
&\geq \sum_{j=1}^{m} w_j \left(\frac{1}{2}\left(1 - \frac{1}{2^{l_j}}\right) + \frac{1}{2}\left(1 - \left[1 - \frac{1}{l_j}\right]^{l_j}\right) z_j^*\right) \\
&\geq \frac{3}{4}\sum_{j=1}^{m} w_j z_j^* \geq \frac{3}{4}\mathrm{OPT}(\phi).
\end{aligned}$$

So, we have a $\frac{4}{3}$-approximation algorithm!