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:**

  \[ \varphi = (x_1 \lor \overline{x}_2) \land (x_1 \lor x_3 \lor \overline{x}_4 \lor x_6) \land (\overline{x}_2 \lor \overline{x}_3 \lor x_4) \land \overline{x}_5 \]

  - **Clause 1**
  - **Clause 2**
  - **Clause 3**
  - **Clause 4**

- **Literals:** \( \overline{x}_2, \ x_4, \text{etc.} \)

- **Truth assignment:** \( a : \{x_1, \ldots, x_n\} \rightarrow \{\text{TRUE, FALSE}\} \)

- For integers \( k \geq 2 \), a **\( k \)-CNF formula** is a CNF formula in which each clause is of size **at most** \( k \),

- an **\( E_k \)-CNF formula** is a CNF formula in which each clause is of size **exactly** \( k \).
Satisfiability Problems

- **MAX-SAT**: given a CNF formula $\varphi$, find a truth assignment satisfying as many clauses as possible
- **MAX-$k$SAT**: given a $k$-CNF formula $\varphi$, find a truth assignment satisfying as many clauses as possible
- **MAX-EkSAT**: given an Ek-CNF formula $\varphi$, 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 \( a_1, \ldots, a_n \), we have

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

(1)

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

\[
w_1 a_1 + \cdots + w_n a_n \geq a_1^{w_1} \cdots a_n^{w_n}.
\]  

(2)

Equality holds iff all \( a_i \) are equal.
The Cauchy-Schwarz Inequality

Theorem (Cauchy-Schwarz inequality)

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

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

(3)
Theorem (Jensen inequality)

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

$$f\left(\sum_{i=1}^{n} w_ix_i\right) \leq \sum_{i=1}^{n} w_if(x_i).$$

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 \textsc{Max-E3-Sat}

The Algorithm

Assign each variable to \textsc{true}/\textsc{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 $S_\varphi$ be the number of satisfied clauses. Then,

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

($m$ 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,

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

- Both \(E[S_\varphi \mid x_1 = \text{TRUE}]\) and \(E[S_\varphi \mid x_1 = \text{FALSE}]\) can be computed in polynomial time.

- Suppose \(E[S_\varphi \mid x_1 = \text{TRUE}] \geq E[S_\varphi \mid x_1 = \text{FALSE}]\), then

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

- Set \(x_1 = \text{TRUE}\), let \(\varphi'\) be \(\varphi\) 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 \text{TRUE}/\text{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 $S_{\phi}$ be the total weight of satisfied clauses. Then,

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

- 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
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

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

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

- Let \( N = \{ j \mid C_j = \{ \bar{x}_i \} \} \), 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,

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

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

The Relaxation

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

Let $(y^*, z^*)$ be an optimal solution to the LP.
Randomized Rounding with Many Biased Coins

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

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

\[
\mathbb{E}[S_\phi] \geq \sum_{j=1}^{m} w_j \left( 1 - \left[ 1 - \frac{z_j^*}{l_j} \right]^{l_j} \right) \\
\geq \sum_{j=1}^{m} w_j \left( 1 - \left[ 1 - \frac{1}{l_j} \right]^{l_j} \right) z_j^* \\
\geq \min_j \left( 1 - \left[ 1 - \frac{1}{l_j} \right]^{l_j} \right) \sum_{j=1}^{m} w_j z_j^* \\
\geq \left( 1 - \frac{1}{e} \right) \text{OPT}(\phi).
\]

**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.

\[
\mathbb{E}[\max\{S_1^\phi, S_2^\phi\}] \geq \mathbb{E}\left[(S_1^\phi + S_2^\phi)/2\right] \\
\geq \sum_{j=1}^{m} w_j \left(\frac{1}{2} \left(1 - \frac{1}{2^{l_j}}\right) + \frac{1}{2} \left(1 - \left[1 - \frac{1}{l_j}\right]^{l_j}\right) z_j^*\right) \\
\geq \frac{3}{4} \sum_{j=1}^{m} w_j z_j^* \geq \frac{3}{4} \OPT(\phi).
\]

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