Randomization

Algorithmic design patterns.
- Greed.
- Divide-and-conquer.
- Dynamic programming.
- Network flow.
- Randomization.

Randomization. Allow fair coin flip in unit time.

Why randomize? Can lead to simplest, fastest, or only known algorithm for a particular problem.

Ex. Symmetry breaking protocols, graph algorithms, quicksort, hashing, load balancing, Monte Carlo integration, cryptography.
Monte Carlo vs. Las Vegas Algorithms

**Monte Carlo algorithm.** Guaranteed to run in poly-time, likely to find correct answer.
**Ex:** Contraction algorithm for global min cut.

**Las Vegas algorithm.** Guaranteed to find correct answer, likely to run in poly-time.
**Ex:** Randomized quicksort, Johnson's MAX-3SAT algorithm.

Remark. Can always convert a Las Vegas algorithm into Monte Carlo, but no known method to convert the other way.
13.2 Global Minimum Cut
Global Minimum Cut

Global min cut. Given a connected, undirected graph $G = (V, E)$ find a cut $(A, B)$ of minimum cardinality.

Applications. Partitioning items in a database, identify clusters of related documents, network reliability, network design, circuit design, TSP solvers.

Network flow solution.

- Replace every edge $(u, v)$ with two antiparallel edges $(u, v)$ and $(v, u)$.
- Pick some vertex $s$ and compute min $s$-$v$ cut separating $s$ from each other vertex $v \in V$.

False intuition. Global min-cut is harder than min $s$-$t$ cut.
**Contraction Algorithm**

**Contraction algorithm.** [Karger 1995]

- Pick an edge $e = (u, v)$ uniformly at random.
- **Contract** edge $e$.
  - replace $u$ and $v$ by single new super-node $w$
  - preserve edges, updating endpoints of $u$ and $v$ to $w$
  - keep parallel edges, but delete self-loops
- Repeat until graph has just two nodes $v_1$ and $v_2$.
- Return the cut (all nodes that were contracted to form $v_1$).
**Claim.** The contraction algorithm returns a min cut with prob $\geq 2/n^2$.

**Pf.** Consider a global min-cut $(A^*, B^*)$ of $G$. Let $F^*$ be edges with one endpoint in $A^*$ and the other in $B^*$. Let $k = |F^*| = $ size of min cut.

- In first step, algorithm contracts an edge in $F^*$ probability $k / |E|$.
- Every node has degree $\geq k$ since otherwise $(A^*, B^*)$ would not be min-cut. $\Rightarrow |E| \geq \frac{1}{2}kn$.
- Thus, algorithm contracts an edge in $F^*$ with probability $\leq 2/n$. 

![Diagram showing contraction algorithm with sets $A^*$, $B^*$, and edges $F^*$]
**Claim.** The contraction algorithm returns a min cut with prob $\geq 2/n^2$.

**Pf.** Consider a global min-cut $(A^*, B^*)$ of $G$. Let $F^*$ be edges with one endpoint in $A^*$ and the other in $B^*$. Let $k = |F^*| = \text{size of min cut.}$

- Let $G'$ be graph after $j$ iterations. There are $n' = n-j$ supernodes.
- Suppose no edge in $F^*$ has been contracted. The min-cut in $G'$ is still $k$.
- Since value of min-cut is $k$, $|E'| \geq \frac{1}{2} kn'$.
- Thus, algorithm contracts an edge in $F^*$ with probability $\leq 2/n'$.

Let $E_j = \text{event that an edge in } F^* \text{ is not contracted in iteration } j$.

$$
\Pr[E_1 \cap E_2 \cap \cdots \cap E_{n-2}] = \Pr[E_1] \times \Pr[E_2 \mid E_1] \times \cdots \times \Pr[E_{n-2} \mid E_1 \cap E_2 \cap \cdots \cap E_{n-3}]
\geq (1-\frac{2}{n}) \times (1-\frac{2}{n-1}) \times \cdots \times (1-\frac{2}{3})
\geq \left(\frac{n-2}{n}\right) \times \left(\frac{n-3}{n-1}\right) \times \cdots \times \left(\frac{2}{4}\right) \times \left(\frac{1}{3}\right)
= \frac{2}{n(n-1)} \times \cdots \times \frac{2}{n-2}
= \frac{2^{\frac{n}{2}}}{n^2}
$$
**Contraction Algorithm**

**Amplification.** To amplify the probability of success, run the contraction algorithm many times.

**Claim.** If we repeat the contraction algorithm \( n^2 \ln n \) times with independent random choices, the probability of failing to find the global min-cut is at most \( 1/n^2 \).

**Pf.** By independence, the probability of failure is at most

\[
(1 - \frac{2}{n^2})^{n^2 \ln n} = \left[ (1 - \frac{2}{n^2})^{\frac{1}{2} n^2} \right]^{2 \ln n} \leq (e^{-1})^{2 \ln n} = \frac{1}{n^2}
\]

\[
(1 - \frac{1}{x})^x \leq 1/e
\]
Global Min Cut: Context

**Remark.** Overall running time is slow since we perform $\Theta(n^2 \log n)$ iterations and each takes $\Omega(m)$ time.

**Improvement.** [Karger-Stein 1996] $O(n^2 \log^3 n)$.

- Early iterations are less risky than later ones: probability of contracting an edge in min cut hits 50% when $n / \sqrt{2}$ nodes remain.
- Run contraction algorithm until $n / \sqrt{2}$ nodes remain.
- Run contraction algorithm *twice* on resulting graph, and return best of two cuts.

**Extensions.** Naturally generalizes to handle positive weights.

**Best known.** [Karger 2000] $O(m \log^3 n)$. faster than best known max flow algorithm or deterministic global min cut algorithm
13.3 Linearity of Expectation
Expectation

Expectation. Given a discrete random variables $X$, its expectation $E[X]$ is defined by:

$$E[X] = \sum_{j=0}^{\infty} j \Pr[X = j]$$

Waiting for a first success. Coin is heads with probability $p$ and tails with probability $1-p$. How many independent flips $X$ until first heads?

$$E[X] = \sum_{j=0}^{\infty} j \cdot \Pr[X = j] = \sum_{j=0}^{\infty} j (1-p)^{j-1} p = \frac{p}{1-p} \sum_{j=0}^{\infty} j (1-p)^j = \frac{p}{1-p} \cdot \frac{1-p}{p^2} = \frac{1}{p}$$

\[\text{j-1 tails \quad 1 head}\]
Expectation: Two Properties

Useful property. If \( X \) is a 0/1 random variable, \( \mathbb{E}[X] = \Pr[X = 1] \).

Pf. \[
\mathbb{E}[X] = \sum_{j=0}^{\infty} j \cdot \Pr[X = j] = \sum_{j=0}^{1} j \cdot \Pr[X = j] = \Pr[X = 1] 
\]

Linearity of expectation. Given two random variables \( X \) and \( Y \) defined over the same probability space, \( \mathbb{E}[X + Y] = \mathbb{E}[X] + \mathbb{E}[Y] \).

Decouples a complex calculation into simpler pieces.
13.4 MAX 3-SAT
Maximum 3-Satisfiability

MAX-3SAT. Given 3-SAT formula, find a truth assignment that satisfies as many clauses as possible.

\[
\begin{align*}
C_1 &= x_2 \lor \overline{x}_3 \lor \overline{x}_4 \\
C_2 &= x_2 \lor x_3 \lor \overline{x}_4 \\
C_3 &= \overline{x}_1 \lor x_2 \lor x_4 \\
C_4 &= \overline{x}_1 \lor \overline{x}_2 \lor x_3 \\
C_5 &= x_1 \lor x_2 \lor x_4
\end{align*}
\]

Remark. NP-hard search problem.

Simple idea. Flip a coin, and set each variable true with probability \(\frac{1}{2}\), independently for each variable.
Claim. Given a 3-SAT formula with k clauses, the expected number of clauses satisfied by a random assignment is $7k/8$.

Pf. Consider random variable $Z_j = \begin{cases} 1 & \text{if clause } C_j \text{ is satisfied} \\ 0 & \text{otherwise} \end{cases}$.

- Let $Z = \text{weight of clauses satisfied by assignment } Z_j$.

\[
E[Z] = \sum_{j=1}^{k} E[Z_j]
\]

(linearity of expectation)

\[
= \sum_{j=1}^{k} \text{Pr}[\text{clause } C_j \text{ is satisfied}]
\]

\[
= \frac{7}{8} k
\]
The Probabilistic Method

Corollary. For any instance of 3-SAT, there exists a truth assignment that satisfies at least a 7/8 fraction of all clauses.

Pf. Random variable is at least its expectation some of the time. □

Probabilistic method. We showed the existence of a non-obvious property of 3-SAT by showing that a random construction produces it with positive probability!
Maximum 3-Satisfiability: Analysis

Q. Can we turn this idea into a 7/8-approximation algorithm? In general, a random variable can almost always be below its mean.

Lemma. The probability that a random assignment satisfies \( \geq 7k/8 \) clauses is at least \( 1/(8k) \).

Pf. Let \( p_j \) be probability that exactly \( j \) clauses are satisfied; let \( p \) be probability that \( \geq 7k/8 \) clauses are satisfied.

\[
\frac{7}{8}k = E[Z] = \sum_{j \geq 0} j p_j
\]
\[
= \sum_{j < 7k/8} j p_j + \sum_{j \geq 7k/8} j p_j
\]
\[
\leq \left( \frac{7k}{8} - \frac{1}{8} \right) \sum_{j < 7k/8} p_j + k \sum_{j \geq 7k/8} p_j
\]
\[
\leq \left( \frac{7}{8}k - \frac{1}{8} \right) \cdot 1 + kp
\]

Rearranging terms yields \( p \geq 1/(8k) \).
Maximum 3-Satisfiability: Analysis

**Johnson's algorithm.** Repeatedly generate random truth assignments until one of them satisfies $\geq 7k/8$ clauses.

**Theorem.** Johnson's algorithm is a 7/8-approximation algorithm.

**Pf.** By previous lemma, each iteration succeeds with probability at least $1/(8k)$. By the waiting-time bound, the expected number of trials to find the satisfying assignment is at most $8k$. □
Maximum Satisfiability

Extensions.
- Allow one, two, or more literals per clause.
- Find max weighted set of satisfied clauses.

Theorem. [Asano-Williamson 2000] There exists a 0.784-approximation algorithm for MAX-SAT.

Theorem. [Karloff-Zwick 1997, Zwick+computer 2002] There exists a 7/8-approximation algorithm for version of MAX-3SAT where each clause has at most 3 literals.

Theorem. [Håstad 1997] Unless P = NP, no \( \rho \)-approximation algorithm for MAX-3SAT (and hence MAX-SAT) for any \( \rho > 7/8 \).

very unlikely to improve over simple randomized algorithm for MAX-3SAT
RP and ZPP

**RP.** [Monte Carlo] Decision problems solvable with one-sided error in poly-time.

*One-sided error.*
- If the correct answer is **no**, always return **no**.
- If the correct answer is **yes**, return **yes** with probability \( \geq \frac{1}{2} \).

**ZPP.** [Las Vegas] Decision problems solvable in expected poly-time.

Theorem. \( P \subseteq ZPP \subseteq RP \subseteq NP \).

Fundamental open questions. To what extent does randomization help? Does \( P = ZPP \)? Does \( ZPP = RP \)? Does \( RP = NP \)?