Randomization

Algorithmic design patterns.
• Greedy.
• 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.
Contestion resolution in a distributed system

Contestion resolution. Given \( n \) processes \( P_1, \ldots, P_n \), each competing for access to a shared database. If two or more processes access the database simultaneously, all processes are locked out. Devise protocol to ensure all processes get through on a regular basis.

Restriction. Processes can’t communicate.

Challenge. Need symmetry-breaking paradigm.

\[ P_1 \]
\[ P_2 \]
\[ \vdots \]
\[ P_n \]

Contestion resolution: randomized protocol

Protocol. Each process requests access to the database at time \( i \) with probability \( p = 1/n \).

Claim. Let \( S(i, t) \) = event that process \( i \) succeeds in accessing the database at time \( t \). Then \( 1/(e \cdot n) \leq \Pr[S(i, t)] \leq 1/(2n) \).

Pf. By independence, \( \Pr[S(i, t)] = n \cdot (1 - p) \cdot n^{-1} \cdot (1 - (1 - p/2))^n \).

- Setting \( p = 1/n \), we have \( \Pr[S(i, t)] = 1/n \cdot (1 - 1/n)^{n-1} \).

Useful facts from calculus. As \( n \) increases from 2, the function:
- \( (1 - 1/n)^n \) converges monotonically from \( 1/4 \) up to \( 1/e \).
- \( (1 - 1/n)^{n-1} \) converges monotonically from \( 1/2 \) down to \( 1/e \).
Contention Resolution: randomized protocol

Claim. The probability that process $i$ fails to access the database in $en$ rounds is at most $1/e$. After $en \ln n$ rounds, the probability is $\leq n^{-c}$.

Pf. Let $F[i,t]$ be the event that process $i$ fails to access the database in rounds 1 through $t$. By independence and previous claim, we have

$$\Pr[F[i,t]] \leq (1 - \frac{1}{en})^t \leq \frac{1}{e}$$

- Choose $t = [en]$:
  $$\Pr[F(i,t)] \leq (1 - \frac{1}{en})^{en} \leq \frac{1}{e}$$

- Choose $t = [en] \ln n$:
  $$\Pr[F(i,t)] \leq (\frac{1}{e})^{en \ln n} = n^{-c}$$

$$\Pr[R_i] \geq 1 - n^{-c}$$

Contention Resolution: randomized protocol

Claim. The probability that all processes succeed within $2en \ln n$ rounds is $\geq 1 - 1/n$.

Pf. Let $F[i]$ be the event that at least one of the $n$ processes fails to access the database in any of the rounds 1 through $i$.

$$\Pr[F[i]] = \Pr[\bigcup_{i=1}^{n} F[i,t]] \leq \sum_{i=1}^{n} \Pr[F[i,t]] \leq n(1 - \frac{1}{en})^t$$

Choosing $t = 2[en] \ln n$ yields $\Pr[F[i]] \leq n \cdot n^2 = 1/n$.

$$\Pr[\text{overall success}] \geq 1 - \frac{1}{n}$$

$$\Pr\left[\bigcup_{i=1}^{n} E_i\right] \leq \sum_{i=1}^{n} \Pr[E_i]$$
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 $v_1$).
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 $v_1$).

Reference: Thore Husfeldt

Contraction algorithm

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 $k = |F^*| = \text{size of min cut}$.
- 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 a min-cut $\Rightarrow |E| \geq \frac{1}{2} kn$.
- Thus, algorithm contracts an edge in $F^*$ with probability $\leq 2 / n$.

$\Rightarrow \frac{|F^*|}{|E|} \leq \frac{k}{\frac{1}{2} kn} = \frac{2}{n}$
Contraction algorithm

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 $k$, $|E'| \geq \frac{1}{2} k n'$.
- Thus, algorithm contracts an edge in $F^*$ with probability $\leq 2/n'$.
- Let $E_j$ = event that an edge in $F^*$ is not contracted in iteration $j$.

$$\Pr[E_1 \cap E_2 \cdots \cap E_{n-2}] = \Pr[E_1] \times \Pr[E_2 \mid E_1] \times \cdots \times \Pr[E_{n-2} \mid E_1 \cap E_2 \cdots \cap E_{n-3}]$$

$$\geq \left(1 - \frac{2}{n}\right) \left(1 - \frac{2}{n-1}\right) \cdots \left(1 - \frac{2}{n-j}\right) \left(1 - \frac{2}{n-j}\right)$$

$$= \left(\frac{n}{n-1}\right) \left(\frac{n-1}{n-2}\right) \cdots \left(\frac{n-j}{n-j}\right) \left(\frac{2}{n-j}\right)$$

$$= \frac{2}{n(n-1)} \cdots \frac{2}{n-j}$$
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, then the probability of failing to find the global min-cut is \(\leq 1/n^2\).

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

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

Repeat \(t\) times

\[P_r[\text{success}] \geq 1 - \frac{1}{n^2}\]

\[P_r[\text{fail}] \leq \left(1 - \frac{2}{n^2}\right)^t \leq e^{-\frac{2}{n^2} \cdot t}\]

Pick \(t \sim n^2 \ln n\)

---

**Contraction algorithm: example execution**

<table>
<thead>
<tr>
<th>trial 1</th>
<th>trial 2</th>
<th>trial 3</th>
<th>trial 4</th>
<th>trial 5 (finds min cut)</th>
<th>trial 6</th>
</tr>
</thead>
</table>

---

Reference: Thore Husfeldt
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] $O(m \log^3 n)$.

\[ \text{faster than best known max flow algorithm or deterministic global min cut algorithm} \]