



南方科技大学
SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

Algorithm Design and Analysis (H)

CS216

Instructor: Shan CHEN (陈杉)

chens3@sustech.edu.cn

(slides edited from Prof. Shiqi Yu)



Randomized Algorithms



Randomization

- **Algorithm design patterns.**

- Greedy
- Divide and conquer
- Dynamic programming
- Duality (e.g., network flow)

- **Randomization**

↙ in practice, access to a pseudo-random number generator

- **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, etc.

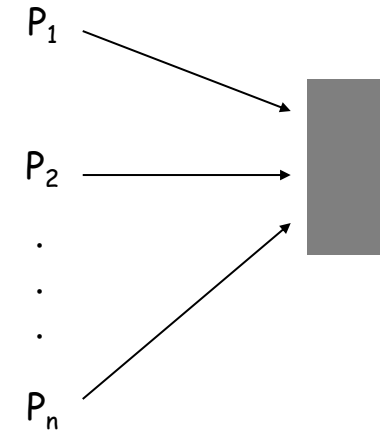


1. Content Resolution



Contention Resolution in a Distributed System

- **Contention resolution.** Given n processes P_1, \dots, P_n , each competing for access to a shared database. If 2 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.





Contention Resolution: Randomized Protocol

- **Protocol.** Each process requests access to the database at time t with probability $p = 1/n$.
- **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 $1/e$.
- **Lemma 1.** 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, $\Pr[S(i, t)] = p(1 - p)^{n-1}$.
 - Setting $p = 1/n$, we have $\Pr[S(i, t)] = \underbrace{1/n (1 - 1/n)^{n-1}}_{\text{between } 1/e \text{ and } 1/2}$. ■

process i requests access, none of remaining processes request access



Contention Resolution: Randomized Protocol

- **Protocol.** Each process requests access to the database at time t with probability $p = 1/n$.
- **Lemma 2.** The probability that process i fails to access the database in $e \cdot n$ rounds is at most $1/e$. After $e \cdot n (c \ln n)$ rounds, the probability $\leq n^{-c}$.
- **Pf.** Let $F[i, t]$ = event that process i fails to access database in rounds 1 through t . By independence and Lemma 1, $\Pr[F(i, t)] \leq (1 - 1/(en))^t$.

➤ Choose $t = \lceil e \cdot n \rceil$: $\Pr[F(i, t)] \leq \left(1 - \frac{1}{en}\right)^{\lceil en \rceil} \leq \left(1 - \frac{1}{en}\right)^{en} \leq \frac{1}{e}$

➤ Choose $t = \lceil e \cdot n \rceil \lceil c \cdot \ln n \rceil$: $\Pr[F(i, t)] \leq \left(\frac{1}{e}\right)^{c \ln n} = n^{-c}$



Contention Resolution: Randomized Protocol

- **Claim.** The probability that **all** processes succeed within $2en \ln n$ rounds is $\geq 1 - 1/n$.
- **Pf.** Let $F[t]$ = event that at least one of the n processes fails to access database in any of the rounds 1 through t .

$$\Pr[F[t]] = \Pr\left[\bigcup_{i=1}^n F[i, t]\right] \leq \sum_{i=1}^n \Pr[F[i, t]]$$

union bound

- Choosing $t = \lceil e \cdot n \rceil \lceil 2 \ln n \rceil$ yields $\Pr[F[t]] \leq n \cdot n^{-2} = 1/n$. ■
- Lemma 2

Union bound. Given events E_1, \dots, E_n ,

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



2. Global Min 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, circuit design, TSP solvers, etc.
- **Network flow solution.**
 - Replace every edge (u, v) with two antiparallel edges (u, v) and (v, u) .
 - Pick some vertex $s \in V$ and compute min s - v cut separating s from each other node $v \in V$.
- **False intuition.** Global min-cut is harder than min s - t cut.



Global Min Cut: Contraction Algorithm

- **Contraction algorithm.** [Karger 1995]

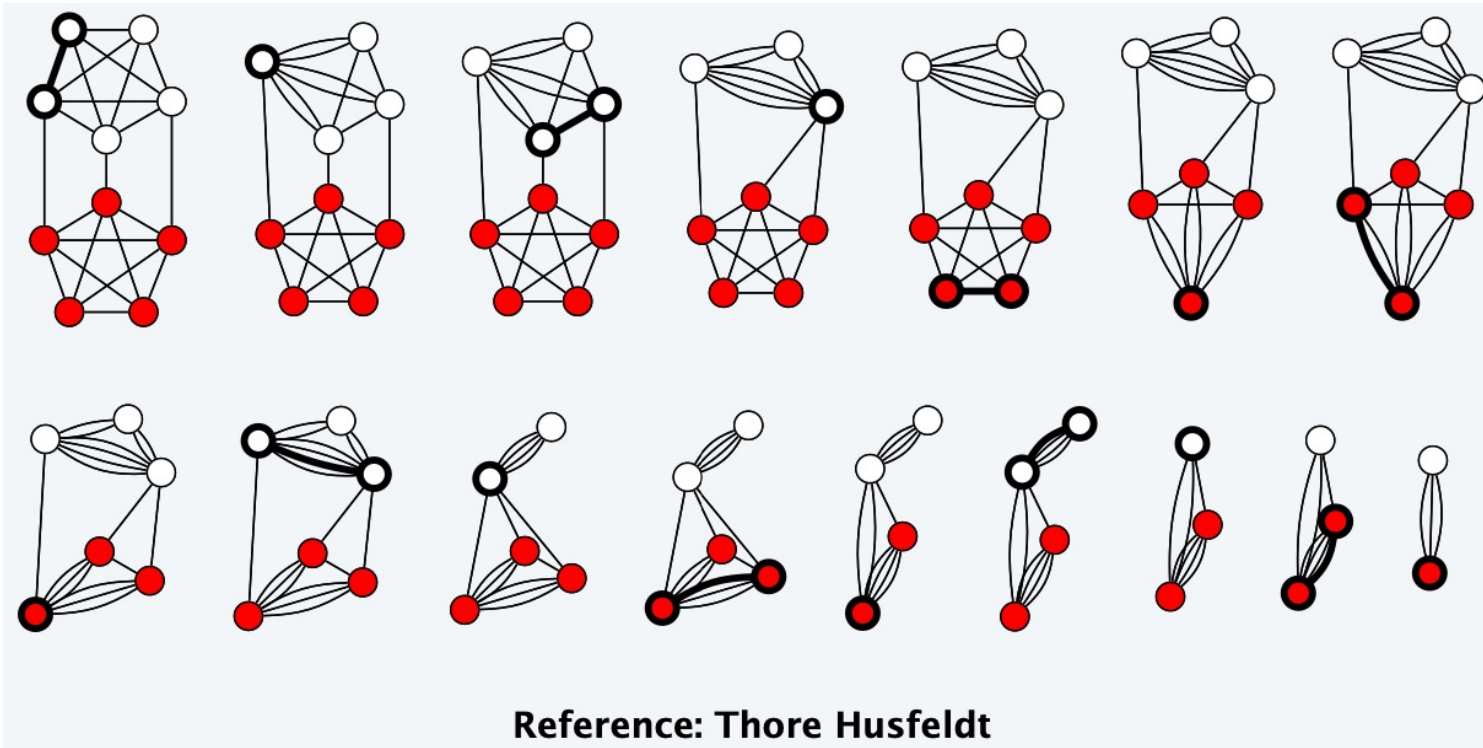
- Pick an edge $e = (u, v)$ uniformly at random. **Contract edge e .**
 - ✓ replace u and v by single new super-node w
 - ✓ preserve edges, updating endpoints of u and v to w
 - ✓ **keep parallel edges**, but delete self-loops
- Repeat until graph has just two super-nodes v_1 and v_2 .
- Return the cut $(S(v_1), S(v_2))$ ($S(v_i)$: all nodes that were contracted to v_i).





Global Min Cut: Contraction Algorithm

- **Contraction algorithm.** [Karger 1995]
 - Pick an edge $e = (u, v)$ uniformly at random. Contract edge e .
 - Repeat until graph has just two super-nodes v_1 and v_2 .
 - Return the cut $(S(v_1), S(v_2))$ ($S(v_i)$: all nodes that were contracted to v_i).

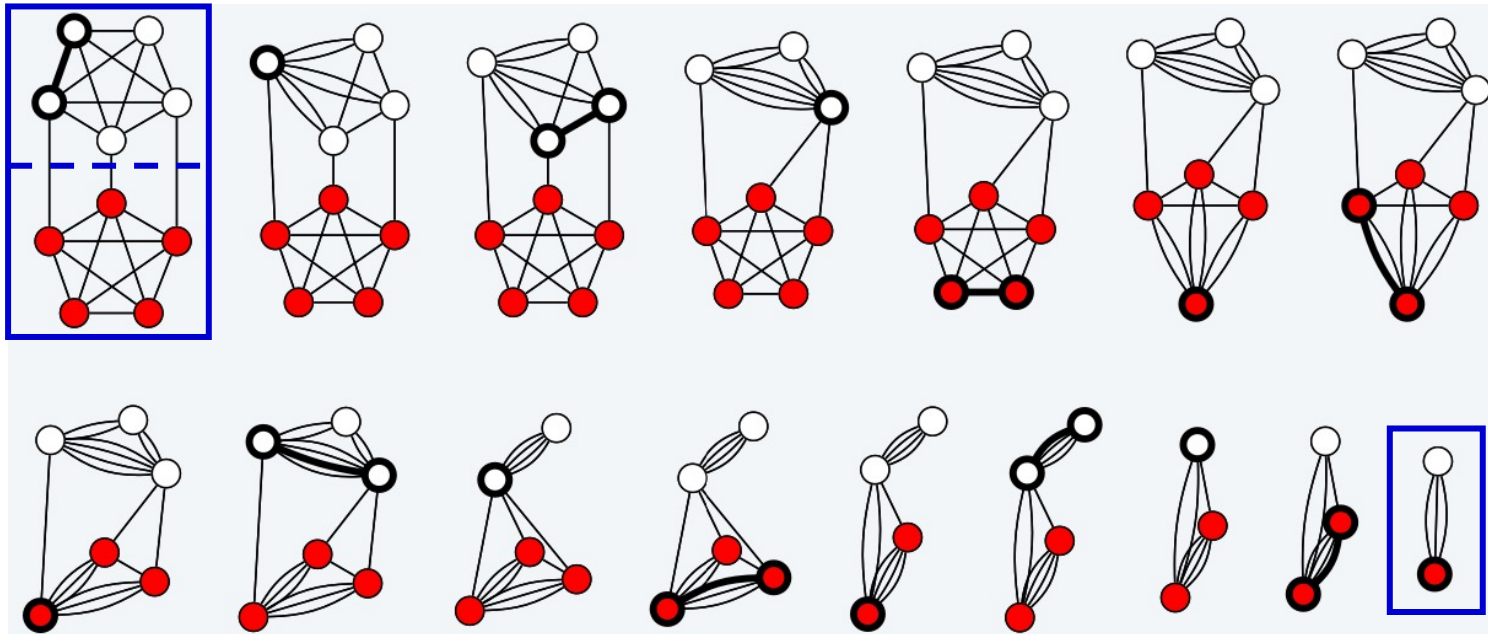


Reference: Thore Husfeldt



Global Min Cut: Contraction Algorithm

- **Contraction algorithm.** [Karger 1995]
 - Pick an edge $e = (u, v)$ uniformly at random. Contract edge e .
 - Repeat until graph has just two super-nodes v_1 and v_2 .
 - **Return the cut $(S(v_1), S(v_2))$** ($S(v_i)$: all nodes that were contracted to v_i).

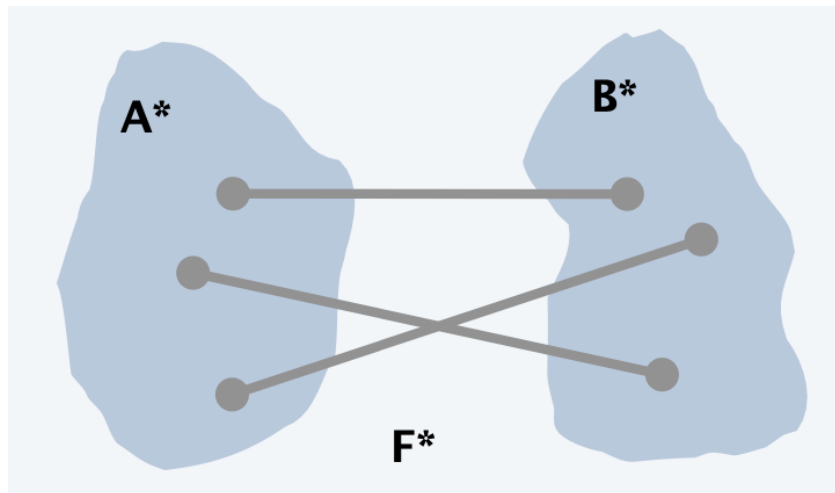


Reference: Thore Husfeldt



Contraction Algorithm: Analysis

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





Contraction Algorithm: Analysis

- **Theorem.** 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.
 - Let G' be graph after j iterations. There are $n' = n - j$ super-nodes.
 - Suppose no edge in F^* has been contracted. The min-cut in G' is still k .
 - Thus, algorithm contracts an edge in F^* with probability $\leq 2 / n'$.
 - Let E_j = event that no edge in F^* is contracted in iteration j .

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



Contraction Algorithm: Amplification

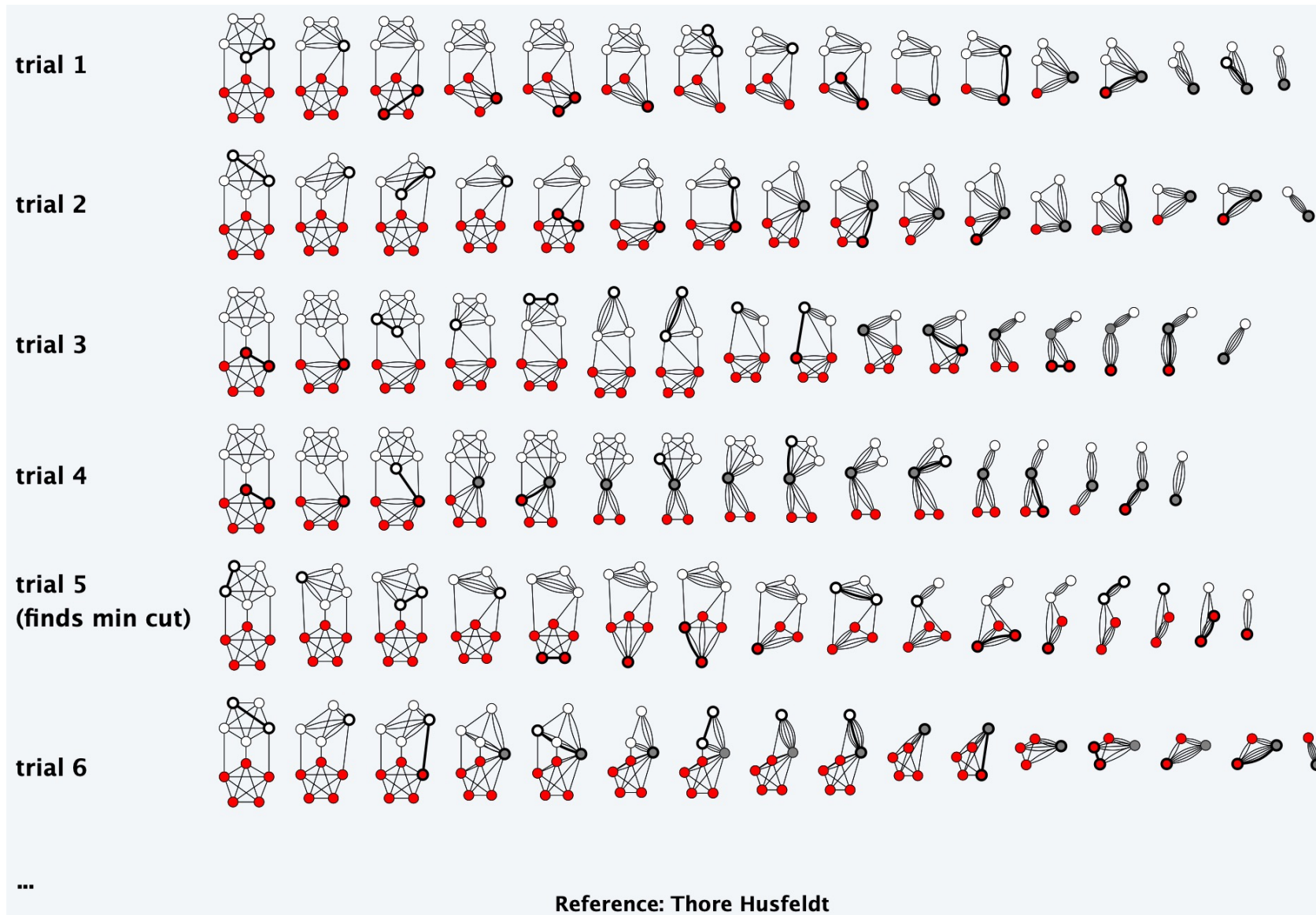
- **Amplification.** To amplify the probability of success, run the contraction algorithm many times **with independent randomness**.
- **Claim.** If we repeat the contraction algorithm $n^2 \ln n$ times, then the probability of failing to find the global min-cut is $\leq 1 / n^2$.
- **Pf.** By independence, the probability of failure is at most

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

\uparrow
 $(1 - 1/x)^x \leq 1/e$



Contraction Algorithm: Demo





More on Global Minimum Cut

- **Remark.** Overall running time $\Theta(n^2 m \log n)$ 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



3. MAX 3-SAT



Maximum 3-Satisfiability

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

$$C_1 = x_2 \vee \overline{x_3} \vee \overline{x_4}$$

$$C_2 = x_2 \vee x_3 \vee \overline{x_4}$$

$$C_3 = \overline{x_1} \vee x_2 \vee x_4$$

$$C_4 = \overline{x_1} \vee \overline{x_2} \vee x_3$$

$$C_5 = x_1 \vee \overline{x_2} \vee \overline{x_4}$$

- **Remark.** NP-hard optimization problem.
- **Simple idea.** Flip a coin, and set each variable true with probability $\frac{1}{2}$, independently for each variable.



Maximum 3-Satisfiability: Analysis

- **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 = number of clauses satisfied by random assignment.

$$\begin{aligned} E[Z] &= \sum_{j=1}^k E[Z_j] \\ \text{linearity of expectation} &\nearrow \\ &= \sum_{j=1}^k \Pr[\text{clause } C_j \text{ is satisfied}] \\ &= \frac{7}{8}k \end{aligned}$$

disjunction of 3 literals
each literal corresponds to a different variable



The Probabilistic Method

- **Claim.** Given a 3-SAT formula with k clauses, the **expected number** of clauses satisfied by a random assignment is $7k / 8$.
- **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.** [Paul Erdős] Prove the existence of a non-obvious property 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?
- **A.** Yes (but 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.

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

➤ 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 $p = 1/(8k)$.
 - The expected number of trials to find the satisfying assignment is
$$\sum_{j=0}^{\infty} j \Pr[j \text{ trials}] \leq \sum_{j=0}^{\infty} (1-p)^{j-1} p = \frac{1}{p} \geq 8k \quad \blacksquare$$
- **Takeaway.** **NP**-hard problems may have good approximation algorithms.



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 3-SAT in which each clause has **at most** 3 literals.
- **Theorem.** [Håstad 1997] Unless **P = NP**, no ρ -approximation algorithm for MAX 3-SAT (and hence MAX SAT) for any $\rho > 7/8$.

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



Monte Carlo vs Las Vegas Algorithms

- **Monte Carlo.** Guaranteed to run poly-time, likely to find correct answer.
- **Ex:** Contraction algorithm for global min cut.
- **Las Vegas.** Guaranteed to find correct answer, likely to run in poly-time.
- **Ex:** Randomized quicksort, Johnson's MAX 3-SAT algorithm.
- stop algorithm after a certain point


• **Remark.** Can always convert a Las Vegas algorithm into Monte Carlo, but no known method (in general) to convert the other way.



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 $\geq 1/2$.
- **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 $RP = NP$?

can decrease probability of false negative
to 2^{-100} by 100 independent repetitions



running time can be unbounded, but fast on average