Maximum 3-Satisfiability

exactly 3 distinct literals per clause

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

$$
\begin{aligned}
C_1 &= x_2 \vee \overline{x_3} \vee \overline{x_4} \\
C_2 &= x_2 \vee x_3 \vee \overline{x_4} \\
C_3 &= \overline{x_1} \vee x_2 \vee x_4 \\
C_4 &= \overline{x_1} \vee \overline{x_2} \vee x_3 \\
C_5 &= x_1 \vee \overline{x_2} \vee \overline{x_4}
\end{aligned}
$$

- Unsurprising result: decision version is NP-complete

- Randomized algorithm:

  Set each variable independently to true with probability ½

Maximum 3-Satisfiability: Analysis

- **Lemma:** Given a 3-SAT formula with k clauses, the expected number of clauses satisfied by a random assignment is $7k/8$

- **Proof:** Let $X_j = \begin{cases} 1 & \text{if clause } C_j \text{ is satisfied} \\ 0 & \text{otherwise.} \end{cases}$

- $E[X_j] = Pr[\ X_j = 1\ ] = 1 - (1/2)^3$, since all three literals in the clause must independently be false if the clause is unsatisfied

The Probabilistic Method

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

- **Proof**: $E[X] = 7k/8$, and the random variable $X \geq E[X]$ 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

- Can we get a 7/8-approximation algorithm for MAX-3SAT?
  - a random variable can almost always be below its mean

- **Lemma:** Pr[random assignment satisfies $\geq 7k/8$ clauses] $\geq 1/(8k)$

- **Proof:** p_j = probability that exactly j clauses are satisfied
  p = probability that $\geq 7k/8$ clauses are satisfied

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

Hence $p \geq 1 / (8k)$

Johnson's algorithm for MAX-3SAT

- Repeatedly generate random truth assignments until one of them satisfies $\geq 7k/8$ clauses
- Running time?
- Each assignment succeeds with probability $p \geq 1/(8k)$
- Let X = number of trials to find the satisfying assignment

$$E[X] = \sum_{j=0}^{\infty} j \cdot \Pr[X = j] = \sum_{j=0}^{\infty} j\,(1-p)^{j-1} p = \frac{p}{1-p} \sum_{j=0}^{\infty} j\,(1-p)^{j} = \frac{p}{1-p} \cdot \frac{1-p}{p^2} = \frac{1}{p} \leq 8k$$

- **Theorem** [Håstad 1997] Unless P = NP, no ρ-approximation algorithm for MAX-3SAT for any $\rho > 7/8$

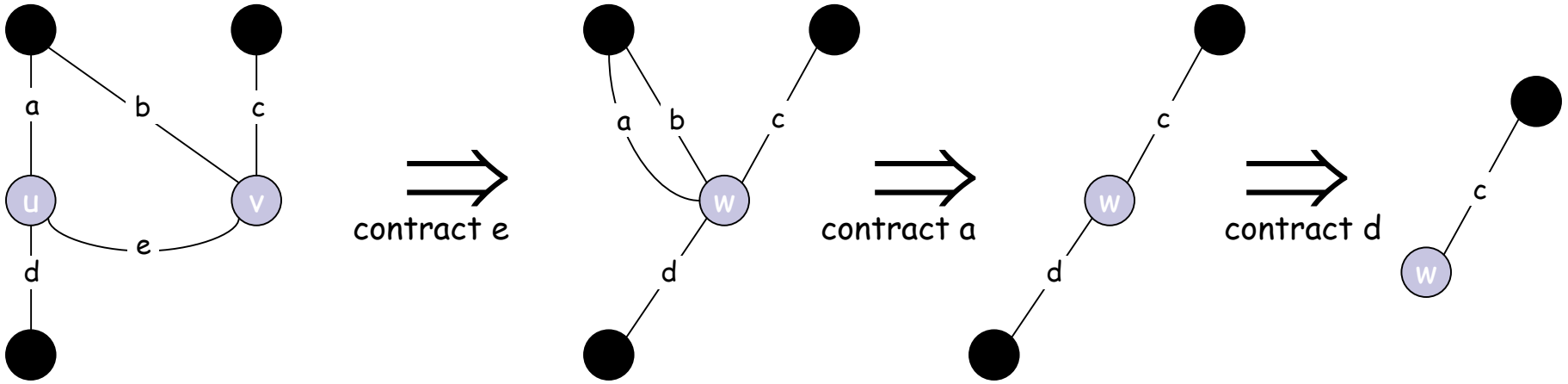
very unlikely to improve over simple randomized algorithm for MAX-3SAT

Global Minimum Cut

- **Input**: connected, undirected graph $G = (V, E)$
- **Output**: a cut (A, B) of minimum cardinality

- Network flow solution:
  - Replace undirected edge $\{u, v\}$ with directed edges (u, v) and (v, u)
  - Pick some vertex s and compute min s-v cut for each $v \in V$

- Randomized algorithm: pick an edge uniformly at random, contract it



Contraction Algorithm [Karger 1995]

1. Pick an edge $e = \{u, v\}$ uniformly at random
2. Contract edge e (keep parallel edges, but delete self-loops)
3. Repeat until graph has just two nodes v_1 and v_2
4. Return the cut (all nodes that were contracted to form v_1)

- Suppose G has a mincut with k edges
- Observation 1: Algorithm "safe" unless it contracts one of these edges
- Observation 2: After j contractions
  - mincut(new G) ≥ mincut(original G)
  - the graph has $n - j$ vertices and at least $k(n - j)/2$ edges

After j contractions, "failure probability" $\leq \frac{k}{k(n-j)/2} = \frac{2}{n-j}$

Algorithm returns mincut with probability $\geq 2/n^2$

- Analysis: Let E_j be the event that algorithm succeeds in contraction j

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

Probability Amplification

- To increase success probability, do many independent repetitions
- In each repetition, failure probability = $\left(1-\frac{2}{n^2}\right)$
- After k independent repetitions, failure probability = $\left(1-\frac{2}{n^2}\right)^k$
- Lemma: $(1 - 1/x)^x \leq (1/e)$

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

- Run $n^2 \log n$ times (each run takes $O(m)$ time)
- Best known: $O(m \log^3 n)$

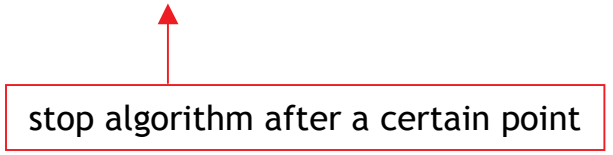
faster than best known max flow algorithm or deterministic global min cut algorithm

Monte Carlo vs. Las Vegas Algorithms

- Monte Carlo: Guaranteed poly runtime, likely to find correct answer
  - example: contraction algorithm for global min cut

- Las Vegas: Guaranteed to find correct answer, likely to run in poly-time
  - example: randomized quicksort, Johnson's MAX-3SAT algorithm

- **Remark**: Can always convert a Las Vegas algorithm into Monte Carlo



- No known method 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 ≥ ½

- **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?

Guessing Cards 1

- Shuffle a deck of n cards; try to guess each card one at a time
- **Memoryless guessing:** Can't even remember what's been turned over already (guess a card from full deck uniformly at random)
- What is the expected number of correct guesses?

Let $X_j = 1$ if j^{th} prediction is correct and 0 otherwise

Let X = number of correct guesses = $X_1 + \ldots + X_n$

$$E[X_j] = \Pr[X_j = 1] = 1/n$$

$$E[X] = E[X_1] + \ldots + E[X_n] = 1/n + \ldots + 1/n = 1$$

linearity of expectation

Guessing Cards 2

- Shuffle a deck of n cards; try to guess each card one at a time
- **Guessing with memory**: Guess uniformly from cards not yet seen
- What is the expected number of correct guesses?
- **Claim**: It is $\Theta(\log n)$

Let $X_j = 1$ if j^{th} prediction is correct and 0 otherwise
Let X = number of correct guesses = $X_1 + \ldots + X_n$

$E[X_j] = \Pr[X_j = 1] = 1/(n - j - 1)$
$E[X] = E[X_1] + \ldots + E[X_n] = 1/n + \ldots + 1/2 + 1/1 = H(n) = \Theta(\log n)$

Coupon Collector

- There are n different types of coupons. Each box of cereal contains one coupon (all types are equally likely).
- How many boxes before you have ≥ 1 coupon of each type?
- *Hint*: let X_j = number of boxes until j+1st coupon seen ($X_0 = 1$)
- **Claim:** The expected number of boxes is $\Theta(n \log n)$
- Let X = number of steps in total = $X_0 + X_1 + \ldots + X_{n-1}$

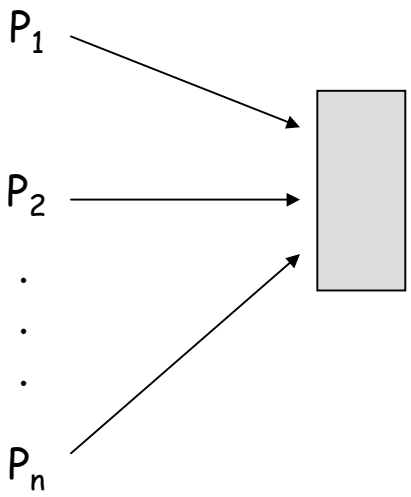
$$E[X] = \sum_{j=0}^{n-1} E[X_j] = \sum_{j=0}^{n-1} \frac{n}{n-j} = n \sum_{i=1}^{n} \frac{1}{i} = n H(n)$$

prob of success = (n-j)/n
⇒ expected waiting time = n/(n-j)

Contention Resolution in a Distributed System

- n processes P_1, …, P_n compete for access to a shared database
  - if two or more processes access the database in the same round, all processes are locked out for that round

- **Randomized protocol**: At each time-step, each P_j accesses the database independently with probability p



Prove the following:

1. Pr[P_j accesses in a given round] $\geq 1/(e \cdot n)$
   (choose p to maximize probability)

2. Pr[P_j fails to access in $2e \cdot n \ln n$ rounds] $\leq 1/n^2$

3. Pr[all processes access in $2e \cdot n \ln n$ rounds] $\geq 1 - 1/n$

Contention Resolution: Randomized Protocol

- **Claim:** Pr[P_j accesses in a given round] $\geq 1/(e \cdot n)$

- **Proof:** By independence,

  Pr[P_j accesses in a given round] = $p(1 - p)^{n-1}$

  process j accesses

  none of remaining n-1 processes request access

  - Maximized when $p = 1/n$

- 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 $1/e$

Contention Resolution: Randomized Protocol

- **Claim:** Pr[P_j fails to access in $ce \cdot n \ln n$ rounds] $\leq n^{-c}$

- **Proof:** By independence and previous claim, we have

  Pr[P_j fails to access in $ce \cdot n \ln n$ rounds] $\leq [(1 - 1/(en))^{e \cdot n}]^{c \ln n}$

  $\leq [1/e]^{c \ln n}$

  $= n^{-c}$

- **Claim:** Pr[all processes access in $2e \cdot n \ln n$ rounds] $\geq 1 - 1/n$

- **Proof:** Pr[at least one process fails in $2e \cdot n \ln n$ rounds]

  $= \Pr[\bigcup_{j=1}^{n} P_j$ fails to access in $2e \cdot n \ln n$ rounds$]$

  $\leq \sum_{j=1}^{n} \Pr[P_j$ fails to access in $2e \cdot n \ln n$ rounds$] \leq n \cdot n^{-2} = 1/n$

Independent Set

- Let $G = (V, E)$ be a graph with n vertices and m edges ($m \geq n/2$)

```
RandIndenpendentSet(G) {
 A = empty set
 for each v in V
   add v to A with probability p

 for each edge e between vertices in A
   delete one endpoint of e from A
 return A
}
```

- After Phase 1:
  - let X be the size of A
  - let Y be the number of edges between vertices in A
- After Phase 2, $E[\,|A|\,] \geq E[\,X - Y\,] = E[X] - E[Y] = np - mp^2$ (max value $= n^2/4m$, when $p = n/2m$)
- By the probabilistic method, G has an independent set of size $\geq n^2/4m$