COMPSCI 311: Introduction to Algorithms

Lecture 24: Randomized Algorithms

Marius Minea

University of Massachusetts Amherst

Why Randomized Algorithms ?

- Efficient in expectation
- Optimal with high probability
- Break (undesired) symmetry
- Show some solution exists, or derive bound on number

Types of randomized algorithms:

- Fail with some small probability
- Always succeed, but running time is random (perhaps non-polynomial)

Examples

- Min-Cut
- Contention Resolution
- Max 3-SAT

Min-Cut Revisited

Given *undirected* $G = (V, E)$, partition V in two sets $(S, V \setminus S)$ minimizing $|C_{\min}| = |\{(u, v) \in E, u \in S, v \in V \setminus S\}|$

Cut minimum number of edges to disconnect graph

Can find minimum *emph{s – t cut* (fixed s and t)) in *directed* graph (network flow)

This is *global* min-cut – is it harder ?

Global Min-Cut: Reduction to Network Flows

- duplicate edges, make directed: (u, v) and (v, u)
- pick arbitrary s (must be on some part of cut)
- pick each node in $V \setminus \{s\}$ as t, run network flow
- choose smallest of $n - 1$ $s - t$ cuts

Complexity? $O(mn^2)$

Contraction Algorithm (Karger, 1995)

Idea: only edges across cut matter.

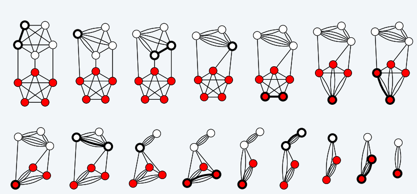
Like collapsing S and $V \setminus S$ into one node each

Allow multiple edges between nodes (*multigraph*)

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 u_1 and v_1.
- Return the cut (all nodes that were contracted to form v_1).



Reference: Thore Husfeldt

11

slide credit: Kevin Wayne / Pearson

Contraction algorithm: example execution

trial 1

trial 2

trial 3

trial 4

trial 5
(finds min cut)

trial 6

...

Reference: Thore Husfeldt

What chance to get a min cut ?

Let $C_{\min}$ be the set of edges in the minimum cut.

We can fail at each step, if an edge in $C_{\min}$ is contracted.

Let $k = |C_{\min}|$. Fail in step 1 with probability $p_{\text{fail}}(1) = k/|E|$.

But if any node v has degree $< k$, min cut is $< k$ (isolate v)

Thus, $|E| \geq nk/2$, and $p_{\text{fail}}(1) \leq \dfrac{k}{kn/2} = \dfrac{2}{n}$

Need to merge (contract) nodes $n-2$ times, until two left.

Each time, $|E'| \geq kn'/2$

$p_{\text{suc}} = p_{\text{suc}}(1)p_{\text{suc}}(2)\cdots p_{\text{suc}}(n-2) \geq \frac{n-2}{n}\frac{n-3}{n-1}\cdots\frac{1}{3} = \frac{2}{n(n-1)}$

Expected *polynomial* number of runs to succeed!

Contraction algorithm

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

with independent random choices,

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

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

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

How many minimum cuts?

A graph may have several minimum cuts

When computing probability to succeed, we actually proved more!

We've shown the probability to return *any* minimum cut is $\geq \frac{2}{n(n-1)}$

But any two cuts are distinct! Let their number be c

$p_{\min-\text{cut}} = p_{\min-\text{cut}_1} + \ldots + p_{\min-\text{cut}_c} \geq c\frac{2}{n(n-1)}$

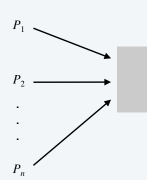
But $p_{\min-\text{cut}} \leq 1 \;\Rightarrow\; c \leq n(n-1)/2$

Contention resolution in a distributed system

Contention 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.



Contention Resolution: randomized protocol

Counterproductive if all request at once

Protocol: Each process requests access with probability p.

Probability of process i to succeed:
 it requests access: p
 noone else requests access: $(1-p)^{n-1} \Rightarrow p_{\text{suc}} = p(1-p)^{n-1}$

Maximize: derivative $f'(p) = 0$: $((1-p)-(n-1)p)(1-p)^{n-2} = 0$

$\Leftrightarrow (1-p)-(n-1)p = 0 \quad \Leftrightarrow p = 1/n$

Success probability: $p_{\text{suc}} = \frac{1}{n}(1-\frac{1}{n})^{n-1} = \frac{1}{n-1}(1-\frac{1}{n})^n \in [\frac{1}{en}, \frac{1}{2n}]$

We know $\lim_{n\to\infty}(1-\frac{1}{n})^n = \frac{1}{e} \Rightarrow$ in the limit $p_{\text{suc}} \simeq \frac{1}{en} = \Theta(1/n)$

Success probability of *some* process in a given round: $n \cdot p_{\text{suc}} \simeq \frac{1}{e}$

Expected waiting time for one process: $\frac{1}{p_{\text{suc}}} \simeq e \cdot n$

Randomization can be **efficient**!

Maximum 3-satisfiability

exactly 3 distinct literals per clause

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

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

Remark. **NP**-hard search problem.

Simple idea. Flip a coin, and set each variable true with probability ½, independently for each variable.

What does this simple idea get us ?

Probability of all 3 literals in a clause false: $(\frac{1}{2})^3 = \frac{1}{8}$

Probability of a clause satisfied: $1 - \frac{1}{8} = \frac{7}{8}$

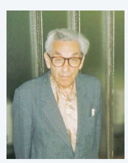
$\Rightarrow$ for k clauses, expected $\frac{7}{8}k$ are satisfied

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. [Paul Erdös] Prove the existence of a non-obvious property by showing that a random construction produces it with positive probability!



What else does this simple idea get us ?

Every 3-SAT instance with ≤ 7 clauses is satisfiable!
because there is some assignment satisfying $\geq \frac{7}{8}k$, and $\frac{7}{8}k > k - 1$ for $k \leq 7$, thus all k must be satisfied

Probability of random assignment satisfying $\geq \frac{7}{8}k$ clauses is $\geq \frac{1}{8k}$

Let p_j = probability that j clauses are satisfied.
Group expected number by $j < \frac{7}{8}k$ and $j \geq \frac{7}{8}k$.

$\frac{7}{8}k = \sum_{j<\frac{7}{8}k} j \cdot p_j + \sum_{j \geq \frac{7}{8}k} j \cdot p_j$

largest j in left sum is $< \frac{7}{8}k \leq \frac{7k-1}{8}$

$\leq (\frac{7}{8}k - \frac{1}{8}) \sum_{j<\frac{7}{8}k} p_j + k \sum_{j \geq \frac{7}{8}k} p_j$

$\leq (\frac{7}{8}k - \frac{1}{8}) \cdot 1 + k p_{\text{suc}}$

Thus, $p_{\text{suc}} \geq \frac{1}{8k}$

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 $\geq 1 / (8k)$. By the waiting-time bound, the expected number of trials to find the satisfying assignment is at most $8k$. ▪

Monte Carlo vs. Las Vegas algorithms

Monte Carlo. Guaranteed to run in 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.