Randomized Algorithms

Learning Goals.

- Introduce Randomized Algorithms.
  - Review probabilistic concepts and calculations.
- Randomized Contention Resolution.
- Waiting for Success.
- Randomized Global Min-Cut.
- Max 3-SAT

Readings: Reading Chp. 13, up to end of Section 13.6.

Based on Chapter 13, Kleinberg and Tardos

Randomization

Algorithmic design patterns.

- Greed.
- Divide-and-conquer.
- Dynamic programming.
- Network flow.
- Linear Programming.
- Approximation Algs.
- Randomization.
- Local Search.

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.

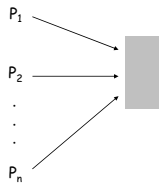
Ex. Symmetry breaking protocols, graph algorithms, quicksort, hashing, load balancing, Monte Carlo integration, cryptography.

Contention Resolution in a Distributed System

Contention resolution. Given $n > 1$ 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 a protocol to ensure all processes get through on a regular basis.

Restriction. Processes can't communicate.
Challenge. Need symmetry-breaking paradigm.

Randomized Rounds. In one round each process bids for access with probability $p = 1/n$. If it turns out exactly one process bids, it is granted access. Otherwise no process gets access, and another round of bidding is done.



What is the expected number of bids per round?

Expectation

Expectation. Given a discrete random variable X over the natural numbers, its expectation E[X] is defined by:

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

Number of Bids: $B = X_1 + X_2 + \ldots X_n \in \{0, 1, 2, \ldots, n\}$,
here $X_j = 1$ denotes process P_j bids (otherwise, $X_j = 0$).

Expected Number of Bids: $E[B] = \sum_{j=0}^{n} j \Pr[B = j]$

Binomial Distribution: $\Pr[B = j] = \binom{n}{j} p^j (1-p)^{n-j}$ where $p = 1/n$.

Number of ways to choose j from n processes.

Prob. of each of j selected processes bidding.

Prob. of each of n-j other processes not bidding.

Calculating E[B] looks messy ...

Expectation: Two Properties

Useful property. If X is a 0/1 random variable, E[X] = Pr[X = 1].

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

Linearity of expectation. Given two random variables X and Y defined over the same probability space, E[X + Y] = E[X] + E[Y].

Not necessarily independent.

Decouples a complex calculation into simpler pieces.

Application. Expected number of bids B:

$$E[B] = \sum_{j=1}^{n} E[X_j] = \sum_{j=1}^{n} \Pr[X_j = 1] = \sum_{j=1}^{n} 1/n = 1$$

Therefore, for each round the expected number of bids is exactly 1, which would then be successful. This is not to say that some bid is successful every round,

$$\Pr[B \neq 1] = 1 - \Pr[B = 1] = 1 - np(1-p)^{n-1} = 1 - (1-1/n)^{n-1} > 0$$

Contention Resolution: Randomized Protocol

Randomized Protocol. Each process requests access to the database on round r with probability p = 1/n.

With some non-zero probability every process has a chance of getting access. So eventually, every process will get access. (Assuming each process releases the database when done.)

Two Questions:

1. How many rounds are needed so that a given process gets access with a probability close to one?

2. How many rounds are needed so that all processes get access with a probability close to one? (Assuming all n keep asking for access.)

Contention Resolution: Randomized Protocol

Answer to Question 1. The probability that a given process i fails to access the database in [ceil(en) ceil(c ln n)] rounds is at most $p_f = n^{-c}$. (Successful with probability $p_s = (1 - p_f)$.)

Answer to Question 2. The probability that all processes succeed within r = [ceil(en) ceil((d+1) ln n)] rounds is at least $1 - 1/n^d$.

That is, O(n logn) rounds are needed to have a strong probability of success $p_s = 1 - p_f$.

Doubling the number of rounds squares p_f (i.e., $p_f = 10^{-2}$, becomes 10^{-4}).

Randomization breaks the symmetry between processes which cannot communicate.

The analysis is in Extra Slides.

Las Vegas Methods

A Las Vegas method is a randomized algorithm which always generates a solution or, stops and reports failure. The runtime is probabilistic (with expected runtime typically restricted to be polynomial).

If the set of possible solutions is not too large, and it is easy to check whether a given element of this set is actually a solution, then a Las Vegas approach is possible.

Examples:
- At most N trials of randomized contention resolution (above).
- Quicksort and randomized median (section 13.5 of K&T).

In the cases above, it is easy to check if you have a solution.

However integration problems or the minimization/maximization of some objective function are hard to check (without a dual). For these problems Monte Carlo methods are useful.

Monte Carlo Methods

A Monte Carlo method is a randomized algorithm which generates a candidate for a solution, which may not be correct.

If the probability of obtaining the correct solution (or a sufficiently accurate approximation) is not exponentially small, then repeated runs can be used to obtain the solution (or an approximate solution) with high probability.

Examples:

- Monte Carlo integration
  - $\int_\Omega f(x)\,dx \sim (1/n)\sum_k f(x_k)$ where $\{x_k\}$ are n random samples from Ω.
  - Gibbs sampling, particle filtering, markov chains.
- Randomized global min cut (below).
- Metropolis-Hastings algorithm (in Local Search lectures)

Global Minimum Cut

Global min cut. Given a connected, undirected graph G = (V, E) find a cut (A, B) of minimum cardinality (i.e., all weights are one).

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

Network flow solution.

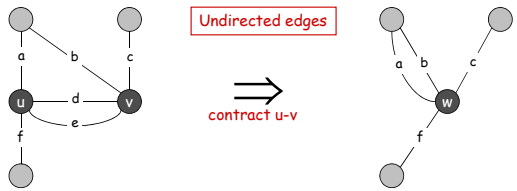
- Replace every edge (u, v) with two anti-parallel directed edges (u, v) and (v, u).
- Pick some vertex s and compute min s-v cuts separating s from each other vertex $v \in V$.

Global min-cut requires at worst (|V|-1) separate runs of min s-t cut (i.e., poly-time).

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

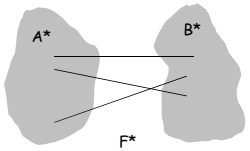
Claim. The contraction algorithm returns A*, B* which, with

prob $\geq 2/n^2$,

is a min cut.
Proof in Extra Slides.

This is a Monte Carlo Algorithm. Is this of any use? It is simple and has O(m) runtime. On any one run it is almost certainly wrong.

What if we run this algorithm many times?

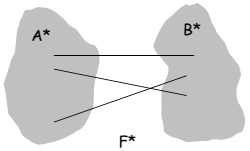


Contraction Algorithm

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*| = size of min cut.

- In first step, algorithm contracts an edge in F* probability k / |E|.
- Every vertex has degree $\geq$ k (otherwise there is a cut of size less than k that separates that vertex). $\Rightarrow$ $|E| \geq \frac{1}{2}kn$.
- Thus, algorithm contracts an edge in F* with probability $k/|E| \leq 2/n$.



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*| = size of min cut.

- Let G' be graph after j iterations. There are n' = n-j vertices in G'.
- 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 $k/|E'| \leq 2/n'$.

- Let E_j = event that an edge in F* is not contracted in iteration j.

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

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:

number of trials

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

Pr[Success in 1 trial]

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

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.

A Monte Carlo method.

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 deterministic algorithm: $O(mn+n^2\log(n))$ Nagamochi-Ibaraki, 1992

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.

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

Remark. NP-hard search 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 = weight of clauses satisfied by assignment Z_j.

linearity of expectation

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

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 probabilistic algorithm guaranteed to be within 7/8 of an optimal solution?

Lemma. The probability that a random assignment satisfies ≥ 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 ≥ 7k/8 clauses are satisfied.

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

Rearranging terms yields $p \geq 1 / (8k)$. ▪

Maximum 3-Satisfiability: Analysis

Johnson's algorithm. Repeatedly generate random truth assignments until one of them satisfies ≥ 7k/8 clauses. This guarantees that it achieves at least 7/8 of the optimum number of satisfiable clauses.

By previous lemma, each iteration succeeds (i.e. satisfies 7k/8 clauses) with probability at least 1/(8k). With repeated random trials, probability of failure decreases exponentially to zero:

Failure Probability:

In 1 trial: (1 - 1/(8k)) (e.g., k = 100, prob ~ 0.9988)

In 2 trials: $(1 - 1/(8k))^2$

. . .

In n trials: $(1 - 1/(8k))^n$ (e.g. k = 100, n = 1000, prob ~ 0.2863,

In 2n trials: $(1 - 1/(8k))^{2n}$ prob ~ $(0.2863)^2$ ~ 0.0820)

converges to 0 as n → ∞

Random Trials: Waiting for Success

Waiting for a first success. Coin is heads with probability p and tails with probability 1-p. Let X be the number of independent flips until the first heads.

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

j-1 tails

1 head

For 0 < x < 1:

$$\sum_{j=1}^{\infty} j x^{j-1} = \frac{d}{dx}\left[\sum_{j=0}^{\infty} x^j\right] = \frac{d}{dx}\left[\frac{1}{1-x}\right] = \frac{1}{(1-x)^2}$$

Therefore:

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

Expected wait is 1/p

Maximum 3-Satisfiability: Analysis

Johnson's algorithm. Each iteration succeeds with probability at least p = 1/(8k).

The expected number of trials to find the an assignment satisfying 7k/8 clauses is at most 1/p = 8k.

Summary: Johnson's algorithm is a randomized 7/8-algorithm for MAX-3SAT with an expected runtime of $O(8k^2)$, where k is the number of clauses.

Definition of a Las Vegas algorithm. The runtime is probabilistic and (in the typical definition) has a finite expected value. The algorithm either finds a solution or stops and reports failure.

Extra Slides

Probability Exercises: Sampling with Replacement

Trial:

- Shuffle a deck of n cards;
- Turn over the top one;
- If it is the ace of spades, stop;
- Otherwise, put the top card back in the deck and repeat.

Claim. If the ace of spades is in the deck, the expected number of trials before stopping is n.

Pf. Same as flipping a coin with probability p = 1/n of landing heads, and counting the number X of flips before the first heads. Previous slide showed E[X] = 1/p. Here p = 1/n, so E[X] = n.

Guessing Cards: Memoryless

Game. Shuffle a deck of n cards; turn them over one at a time; try to guess each card.

Memoryless guessing. No psychic abilities; can't even remember what's been turned over already. Guess a card from full deck uniformly at random.

Claim. The expected number of correct guesses is 1.

Pf. (surprisingly effortless using linearity of expectation)

- Let $X_i = 1$ if i^{th} prediction is correct and 0 otherwise.
- Let X = number of correct guesses = $X_1 + \ldots + X_n$.
- $E[X_i] = \Pr[X_i = 1] = 1/n$.
- $E[X] = E[X_1] + \ldots + E[X_n] = 1/n + \ldots + 1/n = 1$. ▪

linearity of expectation

Guessing Cards: With Memory

Game. Shuffle a deck of n cards; turn them over one at a time; try to guess each card.

Guessing with memory. Guess a card uniformly at random from cards not yet seen.

Claim. The expected number of correct guesses is $\Theta(\log n)$.

Pf.

- Let $X_i = 1$ if i^{th} prediction is correct and 0 otherwise.
- Let X = number of correct guesses = $X_1 + \ldots + X_n$.
- On the i^{th} trial we have (n – i + 1) cards left.
- $E[X_i] = \Pr[X_i = 1] = 1 / (n - i + 1)$.
- $E[X] = E[X_1] + \ldots + E[X_n] = 1/n + \ldots + 1/2 + 1/1 = H(n)$. ▪

H(n) = n^{th} harmonic number

Linearity of expectation

ln(n+1) < H(n) < 1 + ln(n)

Coupon Collector

Coupon collector. Each box of cereal contains a coupon. There are n different types of coupons. Assuming all boxes are equally likely to contain each coupon, how many boxes before you have ≥ 1 coupon of each type?

Claim. The expected number of boxes (i.e., trials) is $\Theta(n \log n)$.

Pf.

- Phase j = time between j and j+1 distinct coupons.
- Let X_j = number of steps you spend in phase j.
- Let X = number of steps in total = $X_0 + X_1 + \ldots + X_{n-1}$.

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

Prob of success = (n-j)/n. Therefore expected waiting time = n/(n-j).

Contention Resolution: Randomized Protocol

Claim 1. Let S[i, r] = event that process i succeeds in accessing the database on round r. Then $1/(e \cdot n) \le \Pr[S(i, r)] \le 1/(2n)$.

Pf. From the protocol, $\Pr[S(i, r)] = p\,(1-p)^{n-1}$

Prob. of process i bidding.

Prob. of all other processes not bidding.

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

For $f(x) = (1-1/x)^{x-1}$ we have $1/e < f(x) \le \frac{1}{2}$ for all $x \ge 2$ (see next slide).

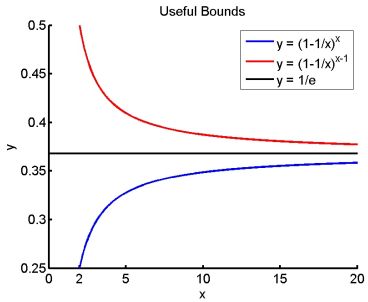
Therefore, the result follows since $n \ge 2$ ▪

Brief Aside: Useful Bounds

Useful facts from calculus. As x increases from 2, the function:

- $f(x) = (1 - 1/x)^x$ increases monotonically from 1/4 up to 1/e
- $g(x) = (1 - 1/x)^{x-1}$ decreases monotonically from 1/2 down to 1/e.

In place of a proof, we have a plot:



Contention Resolution: Randomized Protocol

Question 1: How many rounds are needed so that a given process gets access with a probability close to one? (All processes continue to make requests.)

Answer. The probability that process i fails to access the database in ceil(en) rounds is at most 1/e. This probability is at most n^{-c} after r = [ceil(en) ceil(c ln n)] rounds.

Pf. Let F[i, r] = event that process i fails to access database in rounds 1 through r. By Claim 1, we have $\Pr[S(i, k)] \ge 1/(en)$. By independence of each round we have:

$$\Pr[F(i,r)] = \prod_{k=1}^{r} (1 - \Pr[S(i,k)]) \le \left(1 - \tfrac{1}{en}\right)^r$$

- Choose $r = \lceil e\,n \rceil$: $\Pr[F(i,r)] \le \left(1 - \tfrac{1}{en}\right)^{\lceil en \rceil} \le \left(1 - \tfrac{1}{en}\right)^{en} < 1/e$
- Choose $r = \lceil e\,n \rceil \lceil c \ln n \rceil$: $\Pr[F(i,r)] \le \left(\tfrac{1}{e}\right)^{c \ln n} = n^{-c}$

Contention Resolution: Randomized Protocol

Answer to Question 2. The probability that all processes succeed within r = [ceil(en) ceil((d+1) ln n)] rounds is at least $1 - 1/n^d$.

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

$$\Pr[F[r]] = \Pr\left[\bigcup_{i=1}^{n} F[i,r]\right] \le \sum_{i=1}^{n} \Pr[F[i,r]] \le n\left(1 - \tfrac{1}{en}\right)^r$$

Union bound.

Shown above.

- Choosing c=(d+1), $r = \lceil en \rceil \lceil (d+1) \ln n \rceil$, yields $\Pr[F[t]] \le n \cdot n^{-(d+1)} = 1/n^d$. ▪

Union bound. Given events $E_1, \ldots, E_n$, $\Pr\left[\bigcup_{i=1}^{n} E_i\right] \le \sum_{i=1}^{n} \Pr[E_i]$