« Algorithm Design » by Jon Kleinberg and Éva Tardos

§ Chapter 13 : Randomized Algorithms

When one thinks about random process, it is usually in one of two distinct ways.

1) average-case analysis (randomly generated input)
2) consider algorithms that behave randomly $\longrightarrow$ randomized algorithm

13.1 A First Application: Contention Resolution

Suppose n processes $P_1, P_2, \dots, P_n$, each competing for access to a single shared database. time as being divided into discrete rounds. Database can be accessed by at most one process in a single round. Processes can't communicate with one another at all.

Algorithm

each process will attempt to access the database in each round with probability p, independently of the decisions of the other processes.

Analyzing the Algorithm

let $A[i,t]$ denote the event that P_i attempts to access database in round t. $Pr[A[i,t]] = p$

let $\mathcal{S}[i,t]$ denote the event that P_i succeeds in accessing the database in round t. $Pr[\overline{A[i,t]}] = 1-p$

$$Pr[\mathcal{S}[i,t]] = Pr[A[i,t]] \cdot \prod_{j\neq i} Pr[\overline{A[j,t]}] = p(1-p)^{n-1}$$

$$f(p) = p(1-p)^{n-1}$$

$$f'(p) = (1-p)^{n-1} - p\cdot(n-1)(1-p)^{n-2}$$

$$(1-p)^{n-1} - (n-1)\cdot p(1-p)^{n-2} \geq 0$$

$$(1-p) \geq (n-1)p$$

$$\frac{1}{p} - 1 \geq n-1$$

$$p \leq \frac{1}{n}$$

we set $p = \frac{1}{n}$, $\quad Pr[\mathcal{S}[i,t]] = \frac{1}{n}(1-\frac{1}{n})^{n-1}$

(13.1)
(a) The function $(1-\frac{1}{n})^n$ converges monotonically from $\frac{1}{4}$ up to $\frac{1}{e}$ as n increases from 2.
(b) The function $(1-\frac{1}{n})^{n-1}$ converges monotonically from $\frac{1}{2}$ down to $\frac{1}{e}$ as n increases from 2.

$$\therefore \quad \frac{1}{en} \leq \underbrace{Pr[\mathcal{S}[i,t]]}_{\Theta(\frac{1}{n})} \leq \frac{1}{2n}$$

let $F[i,t]$ denote the "failure event" that process P_i does not succeed in any of the rounds 1 through t.

$$Pr[F[i,t]] = Pr\left[\bigcap_{r=1}^{t} \overline{\mathcal{S}[i,r]}\right] = \left[1 - \frac{1}{n}(1-\frac{1}{n})^{n-1}\right]^t \leq \left(1-\frac{1}{en}\right)^t \underset{\text{set } t = \lceil en \rceil}{\leq} \left(1-\frac{1}{en}\right)^{en} \leq \frac{1}{e}$$

$t = \lceil en \rceil \cdot c \ln n$

$$Pr[F[i,t]] \le \left(1 - \frac{1}{en}\right)^t = \left(\left(1 - \frac{1}{en}\right)^{\lceil en \rceil}\right)^{c \ln n} \le e^{-c \ln n} = n^{-c}$$

Conclusion: After $\Theta(n)$ rounds ($t = \lceil en \rceil$), the probability that P_i has not succeeded is bounded by a constant ($\frac{1}{e}$); and between then and $\Theta(n \ln n)$, this probability drops to a quantity that is quite small, bounded by an inverse polynomial in n.

Waiting for All Processes to Get Though

$$\mathcal{F}_t = \bigcup_{i=1}^{n} F[i,t]$$

(13.2) (The Union Bound) Given events $\mathcal{E}_1, \mathcal{E}_2, \ldots, \mathcal{E}_n$, we have

$$Pr\left[\bigcup_{i=1}^{n} \mathcal{E}_i\right] \le \sum_{i=1}^{n} Pr[\mathcal{E}_i].$$

$$\therefore \quad Pr[\mathcal{F}_t] \le \sum_{i=1}^{n} Pr[F[i,t]]$$

choose $t = \lceil en \rceil (c \ln n)$, $Pr[F[i,t]] \le n^{-c}$

choose $t = 2 \lceil en \rceil \ln n$

$$Pr[\mathcal{F}_t] \le \sum_{i=1}^{n} Pr[F[i,t]] \le n \cdot n^{-2} = n^{-1}$$

(13.3) With probability at least $1 - n^{-1}$, all processes succeed in accessing the database at least once within $t = 2\lceil en \rceil \ln n$ rounds.

13.2 Finding the Global Minimum Cut

undirected graph $G = (V, E)$, define a cut of G to be a partition of V into two nonempty sets A and B. For a cut (A, B), the size of (A, B) is the number of edges with one end in A and the other in B. A global minimum cut is a cut of minimum size.

(13.4) There is a polynomial-time algorithm to find a global min-cut in an undirected graph G.

convert to directed graph

fix s. for every $t \in V - \{s\}$, run push-relabel.

the best among these will be a global min-cut of G.

David Karger 1992.

Algorithm. (Contraction Algorithm)

└ works with connected multigraph.

The Contraction Algorithm applied to a multigraph $G = (V, E)$:

For each node v, we will record the set $S(v)$ of the nodes that have been contracted into v.

Initially $S(v) = \{v\}$ for each v

If G has two nodes v_1 and v_2, then return the cut $(S(v_1), S(v_2))$.

Else choose an edge $e = (u, v)$ of G uniformly at random

Let G' be the graph resulting from the contraction of e,
with a new node z_{uv} replacing u and v.

Define $S(z_{uv}) = S(u) \cup S(v)$

Apply the Contraction Algorithm recursively to G'

End if

Analyzing the Algorithm

(13.5) The Contraction Algorithm returns a global min-cut of G with probability at least $\frac{1}{\binom{n}{2}}$

Suppose the global min-cut has size k, a set F of k edges with one end in A and the other in B.
$\hookrightarrow$ every node in G has degree at least k.

We want an upper bound on the probability that an edge in F is contracted, and for this we need a lower bound on the size of E.

$$|E| \geq \frac{1}{2} kn$$

Hence the probability than an edge in F is contracted is at most

$$\frac{k}{\frac{1}{2}kn} = \frac{2}{n}.$$

Consider the situation after j iterations, there are $n-j$ supernodes in G'.
Thus G' has at least $\frac{1}{2}k(n-j)$ edges. So the probability that an edge of F is contracted in the next iteration $j+1$ is at most.

$$\frac{k}{\frac{1}{2}k(n-j)} = \frac{2}{n-j}$$

We write $\mathcal{E}_j$ for the event that an edge of F is not contracted in iteration j, then we have shown $Pr[\mathcal{E}_1] \geq 1 - \frac{2}{n}$ and $Pr[\mathcal{E}_{j+1} \mid \mathcal{E}_1 \cap \mathcal{E}_2 \cap \cdots \cap \mathcal{E}_j] \geq 1 - \frac{2}{n-j}$

We are interested in lower-bounding the quality $Pr[\mathcal{E}_1 \cap \mathcal{E}_2 \cdots \cap \mathcal{E}_{n-2}]$.

$$Pr[\mathcal{E}_1] \cdot Pr[\mathcal{E}_2 \mid \mathcal{E}_1] \cdots Pr[\mathcal{E}_{j+1} \mid \mathcal{E}_1 \cap \mathcal{E}_2 \cdots \cap \mathcal{E}_j] \cdots Pr[\mathcal{E}_{n-2} \mid \mathcal{E}_1 \cap \mathcal{E}_2 \cdots \cap \mathcal{E}_{n-3}]$$

$$\geq \left(1 - \frac{2}{n}\right) \cdot \left(1 - \frac{2}{n-1}\right) \cdots \left(1 - \frac{2}{n-j}\right) \cdots \left(1 - \frac{2}{3}\right)$$

$$= \left(\frac{n-2}{n}\right) \cdot \left(\frac{n-3}{n-1}\right)\left(\frac{n-4}{n-2}\right) \cdots \left(\frac{3}{5}\right)\left(\frac{2}{4}\right)\left(\frac{1}{3}\right)$$

$$= \frac{2}{n(n-1)} = \binom{n}{2}^{-1}$$

So we know that a single run of the Contraction Algorithm fails to find a global min-cut with probability at most $(1 - 1/\binom{n}{2})$.

If we run the algorithm $\binom{n}{2}$ times,

$$\left(1 - 1/\binom{n}{2}\right)^{\binom{n}{2}} \le \frac{1}{e}$$

If we run the algorithm $\binom{n}{2} \ln n$ times

$$\left[\left(1 - 1/\binom{n}{2}\right)^{\binom{n}{2}}\right]^{\ln n} \le e^{-\ln n} = n^{-1}$$

Further Analysis: The Number of Global Minimum Cuts

Given an undirected graph $G = (V, E)$, what is the maximum number of global min-cuts it can have (as a function of n)?

undirected graph has $\binom{n}{2}$ global min-cuts.

(13.6) An undirected graph $G = (V, E)$ on n nodes has at most $\binom{n}{2}$ global min-cuts

Let G be a graph, and let $C_1, \ldots, C_r$ denote all its global min-cuts.
Let $\mathcal{E}_i$ denote the event that C_i is returned by the Contraction Algorithm,
let $\mathcal{E} = \bigcup_{i=1}^{r} \mathcal{E}_i$ denote the event that the algorithm returns any global min-cut.

$$Pr[\mathcal{E}] = Pr[\cup_{i=1}^{r} \mathcal{E}_i] = \sum_{i=1}^{r} Pr[\mathcal{E}_i] \ge r / \binom{n}{2}$$

$$Pr[\mathcal{E}] \le 1 \Rightarrow r \le \binom{n}{2}$$

13.3 Random Variables and Their Expectations

random variable: Given a probability space, a random variable X is a function from the underlying sample space to the natural numbers, such that for each natural number j, the set $X^{-1}(j)$ of all sample points taking value j is an event.

$Pr[X=j]$ as loose shorthand for $Pr[X^{-1}(j)]$.

expectation: the "average value" assumed by X.

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

$$S = \sum_{j=1}^{\infty} j (1-p)^j = 1 \cdot (1-p)^1 + 2(1-p)^2 + 3(1-p)^3 + \cdots$$

$$(1-p)S = 1(1-p)^2 + 2(1-p)^3 + 3(1-p)^4 + \cdots$$

$$S - (1-p)S = (1-p)^1 + (1-p)^2 + (1-p)^3 + \cdots = \frac{(1-p)}{p}$$

$$pS = \frac{(1-p)}{p} \Rightarrow S = \frac{(1-p)}{p^2}$$

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

$$= \frac{p}{1-p} \cdot \frac{(1-p)}{p^2} = \frac{1}{p}$$

(13.8) Linearity of Expectation. Given two random variables X and Y defined over the same probability space, we can define $X+Y$ to be the random variable equal to $X(\omega)+Y(\omega)$ on a sample point ω. For any X and Y, we have

$$E[X+Y] = E[X] + E[Y].$$

(13.10) $H(n) = \sum_{i=1}^{n} \frac{1}{i} = 1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n}$ harmonic number $H(n)$.

$$\int_1^{n+1} \frac{1}{x}\,dx = \ln(n+1)$$

$\ln(n+1) < H(n) < 1 + \ln n$, and more loosely $H(n) = \Theta(\log n)$.

Conditional Expectation

Suppose we have a random variable X and an event $\mathcal{E}$ of positive probability. Then we define the conditional expectation of X, given $\mathcal{E}$, to be the expected value of X computed only over the part of sample space corresponding to $\mathcal{E}$.

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

13.4 A Randomized Approximation Algorithm for MAX 3-SAT

Algorithm: set each variable $x_1, \ldots, x_n$ independently to 0 or 1 with probability $\frac{1}{2}$ each.

let $Z_i = 1$ if clause C_i is satisfied, and 0 otherwise.

Thus $Z = Z_1 + Z_2 + \cdots + Z_k$. $E[Z_i]$ is equal to the probability that C_i is satisfied.

In order for C_i not to be satisfied, each of its three variables must be assigned the value that fails to make it true; $(\frac{1}{2})^3 = \frac{1}{8}$ so $E[Z_i] = \frac{7}{8}$.

$$E[Z] = E[Z_1] + E[Z_2] + \cdots + E[Z_k] = \frac{7}{8}k.$$

(13.14) Consider a 3-SAT formula, where each clause has three different variables. The expected number of clauses satisfied by a random assignment is within an approximation factor $\frac{7}{8}$ of optimal.

(13.15) For every instance of 3-SAT, there is a truth assignment that satisfies at least a $\frac{7}{8}$ fraction of all clauses

Cute application: Every instance of 3-SAT with at most seven clauses is satisfiable. $P \geq \frac{1}{8k}$

(13.16) There is a randomized algorithm with polynomial expected running time that is guaranteed to produce a truth assignment satisfying at least $\frac{7}{8}$ fraction of all clauses $n \leq 8k$