The Probabilistic Method

1. If $E[X] = C$, then there are values $c_1 \leq C$ and $c_2 \geq C$ such that $Pr(X = c_1) > 0$ and $Pr(X = c_2) > 0$.
2. If a random object in a set satisfies some property with positive probability then there is an object in that set that satisfies that property.

Theorem

Given any graph $G = (V, E)$ with n vertices and m edges, there is a partition of V into two disjoint sets A and B such that at least $m/2$ edges connect vertex in A to a vertex in B.

Proof.

Construct sets A and B by randomly assign each vertex to one of the two sets.
The probability that a given edge connect A to B is $1/2$, thus the expected number of such edges is $m/2$.
Thus, there exists such a partition. $\square$

Maximum Satisfiability

Given m clauses in CNF (Conjunctive Normal Form), assume that no clause contains a variable and its complement.

Theorem

For any set of m clauses there is a truth assignment that satisfy at least $m/2$ of the clauses.

Proof.

Assign random values to the variables. The probability that a given clause (with k literals) is not satisfied is bounded by

$$1 - 2^{-k} \geq \frac{1}{2}.$$

$\square$

Monochromatic Complete Subgraphs

Given a complete graph on 1000 vertices, can you color the edges in two colors such that no clique of 20 vertices is monochromatic?

Theorem

If $n \leq 2^{(k-1)/2}$ then it is possible to edge color the edges of a complete graph on n points (K_n), such that is has no monochromatic K_k subgraph.

Proof.

Consider a random coloring.
For a given set of k vertices, the probability that the clique defined by that set is monochromatic is bounded by

$$2 \times 2^{-\binom{k}{2}}.$$

There are $\binom{n}{k}$ such cliques, thus the probability that **any** clique is monochromatic is bounded by

$$\binom{n}{k} 2 \times 2^{-\binom{k}{2}} \leq \frac{n^k}{k!} 2 \times 2^{-\binom{k}{2}}$$

$$\leq 2^{(k-1)^2/2 - k(k-1)/2 + 1} \frac{1}{k!} < 1.$$

Thus, there is a coloring with the required property. $\square$

Sample and Modify

An *independent set* in a graph G is a set of vertices with no edges between them.
Finding the largest independent set in a graph is an NP-hard problem.

Theorem

Let $G = (V, E)$ be a graph on n vertices with $dn/2$ edges. Then G has an independent set with at least $n/2d$ vertices.

Algorithm:

1. Delete each vertex of G (together with its incident edges) independently with probability $1 - 1/d$.
2. For each remaining edge, remove it and one of its adjacent vertices.

$X =$ number of vertices that survive the first step of the algorithm.

$$E[X] = \frac{n}{d}.$$

$Y =$ number of edges that survive the first step.
An edge survives if and only if its two adjacent vertices survive.

$$E[Y] = \frac{nd}{2}\left(\frac{1}{d}\right)^2 = \frac{n}{2d}.$$

The second step of the algorithm removes all the remaining edges, and at most Y vertices.
Size of output independent set:

$$E[X - Y] = \frac{n}{d} - \frac{n}{2d} = \frac{n}{2d}.$$

Conditional Expectation

Definition

$$E[Y \mid Z = z] = \sum_{y} y \Pr(Y = y \mid Z = z),$$

where the summation is over all y in the range of Y.

Lemma

For any random variables X and Y,

$$E[X] = \sum_{y} \Pr(Y = y) E[X \mid Y = y],$$

where the sum is over all values in the range of Y.

Derandomization using Conditional Expectations

Given a graph $G = (V, E)$ with n vertices and m edges, we showed that there is a partition of V into A and B such that at least $m/2$ edges connect A to B.

How do we find such a partition?

$C(A, B) =$ number of edges connecting A to B.
If A, B is a random partition $E[C(A, B)] = \frac{m}{2}$.
Algorithm:

1. Let $v_1, v_2, \ldots, v_n$ be an arbitrary enumeration of the vertices.
2. Let x_i be the set where v_i is placed ($x_i \in \{A, B\}$).
3. For $i = 1$ to n do:
    1. Place v_i such that

$$E[C(A, B) \mid x_1, x_2, \ldots, x_i] \\ \geq E[C(A, B) \mid x_1, x_2, \ldots, x_{i-1}] \geq m/2.$$

Lemma

For all $i = 1, \ldots, n$ *there is an assignment of* v_i *such that*

$$E[C(A,B) \mid x_1, x_2, \ldots, x_i]$$
$$\geq E[C(A,B) \mid x_1, x_2, \ldots, x_{i-1}] \geq m/2.$$

Proof.

By induction on i.
For $i = 1$, $E[C(A, B) \mid x_1] = E[C(A, B)] = m/2$
For $i > 1$, if we place v_i randomly in one of the two sets,

$$\begin{aligned} & E[C(A, B) \mid x_1, x_2, \ldots, x_{i-1}] \\ = \; & \frac{1}{2} E[C(A, B) \mid x_1, x_2, \ldots, x_i = A] \\ & + \frac{1}{2} E[C(A, B) \mid x_1, x_2, \ldots, x_i = B]. \end{aligned}$$

$$\begin{aligned} & \max(E[C(A, B) \mid x_1, x_2, \ldots, x_i = A], \\ & E[C(A, B) \mid x_1, x_2, \ldots, x_i = B]) \\ \geq \; & E[C(A, B) \mid x_1, x_2, \ldots, x_{i-1}] \\ \geq \; & m/2 \end{aligned}$$

How do we compute

$$\begin{aligned} & \max(E[C(A,B) \mid x_1, x_2, \ldots, x_i = A], \\ & E[C(A,B) \mid x_1, x_2, \ldots, x_i = B]) \\ \geq \quad & E[C(A,B) \mid x_1, x_2, \ldots, x_{i-1}] \end{aligned}$$

We just need to consider edges between v_i and $v_1, \ldots, v_{i-1}$.

Simple Algorithm:

1. Place v_1 arbitrarily.
2. For $i = 2$ to n do
    1. Place v_i in the set with smaller number of neighbors.

Randomization as a Resource

Complexity is usually studied in terms of resources, TIME and SPACE.

We add a new resource, RANDOMNESS, measured by the number of independent random bits used by the algorithm (= the entropy of the random source).

Example: Packet Routing

We proved:

Theorem

There is an algorithm for permutation routing on an $N = 2^n$-cube that uses a total of $O(nN)$ random bits and terminates with high probability in cn steps, for some constant c.

Can we achieve the same result with fewer random bits?

Theorem

There is an algorithm for permutation routing on an $N = 2^n$-cube that uses a total of $O(n)$ random bits and terminates with high probability in cn steps, for some constant c.

Proof

Let $A(X)$ be a randomized algorithm with input x that uses (up to) s random bits.
Let $A(x, r)$ be the execution of algorithm A with input x and a fixed sequence r on s bits.
We can write $A(X)$ as

1. Choose r uniformly at random in $[0, 2^s - 1]$.
2. Run $A(X, r)$.

In the two phase routing algorithm $s = \log(N^N) = nN$ (it chooses a random destination independently for each packet).
Let $\mathcal{B} = \{B_1, \ldots, B_r\}$ be the a collection of 2^s deterministic algorithms $A(I, r)$.
We proved:

Lemma

For a given input permutation π and a deterministic algorithm B_i chosen uniformly at random from $\mathcal{B}$, the probability that B_i fails to route π in cn steps is bounded by $1/N$.

Choose a random set $\mathcal{D} = \{\mathcal{D}_\infty, \ldots, \mathcal{D}_{\mathcal{N}^\ni}\}$ of N^3 elements in $\mathcal{B}$. Let $X_i^\pi = 1$ is algorithm D_i does NOT route permutation π in cn steps, else $X_i^\pi = 0$

$$E[\sum_{i=1}^{N^3} X_i^\pi] \leq N^2$$

$$Prob(\sum_{i=1}^{N^3} X_i^\pi \geq 2N^2) \leq e^{-N^2/3}$$

$$Prob(\exists\pi \quad \sum_{i=1}^{N^3} X_i^\pi \geq 2N^2) \leq N! e^{-N^2/3} < 1$$

$$Prob(\exists \pi, \quad \sum_{i=1}^{N^3} X_i^{\pi} \geq 2N^2) \leq N! e^{-N^2/3} < 1$$

Theorem

There exists a set $\mathcal{D}$ of N^3 deterministic algorithms, such that for any given permutation π and an algorithm D chosen uniformly at random from $\mathcal{D}$, algorithm D routes π in cn steps with probability $1 - 1/N$. The random choice requires $O(n)$ random bits.

Can we do better?

Do we need any random bits?

Definition

A routing algorithm is **oblivious** if the path taken by one packet is independent of the source and destinations of any other packets in the system.

Theorem

Given an N-node network with maximum degree d the routing time of any deterministic oblivious routing scheme is

$$\Omega\left(\sqrt{\frac{N}{d^3}}\right).$$

Theorem

For any deterministic oblivious algorithm for permutation routing on the $N = 2^n$ cube there is an input permutation that requires $\Omega(\sqrt{N}/n^3)$ steps.

Theorem

Any randomized oblivious routing algorithm for permutation routing on the $N = 2^n$ cube must use $\Omega(n)$ random bits to route an arbitrary permutation in $O(n)$ expected time.

proof

Assume that the algorithm uses k random bits.
It can choose between no more than 2^k possible deterministic executions.
There is a deterministic execution $\tilde{A}$ that is chosen with probability $\geq 1/2^k$.
Let π be an input permutation that requires $\Omega(\sqrt{N}/n^3)$ steps in $\tilde{A}$.
The expected running time of this input permutation on the randomized algorithm is $\Omega(\sqrt{N}/(2^k n^3))$

Should Tables Be Sorted?

Goal: Store a **static dictionary** of n items in a table of $O(n)$ space such that any search takes $O(1)$ time.

Universal hash functions

Definition

Let U be a universe with $|U| \geq n$ and $V = \{0, 1, \ldots, n-1\}$. A family of hash functions $\mathcal{H}$ from U to V is said to be k*-universal* if, for any elements $x_1, x_2, \ldots, x_k$, when a hash function h is chosen uniformly at random from $\mathcal{H}$,

$$\Pr(h(x_1) = h(x_2) = \ldots = h(x_k)) \leq \frac{1}{n^{k-1}}.$$

Example of 2-Universal Hash Functions

Universe $U = \{0, 1, 2, \ldots, m-1\}$
Table keys $V = \{0, 1, 2, \ldots, n-1\}$, with $m \geq n$.
A family of hash functions obtained by choosing a prime $p \geq m$,

$$h_{a,b}(x) = ((ax + b) \bmod p) \bmod n,$$

and taking the family

$$\mathcal{H} = \{h_{a,b} \mid 1 \leq a \leq p-1, 0 \leq b \leq p\}.$$

Lemma

$\mathcal{H}$ *is 2-universal.*

Proof.

For a given pair (a, b), if $ax_1 + b = ax_2 + b \mod p$ then $x_1 = x_2$.
Let $x_1 \neq x_2$.
For each pair $u \neq v$ there is exactly one pair (a, b) (i.e. one hash function) such that $ax_1 + b = u \mod p$ and $ax_2 + b = v \mod p$.
For each choice of v there are at most $\lceil p/n \rceil - 1 \leq (p-1)/n$ values $u \neq v$ such that $u = v \mod p$.
Thus,

$$\Pr(h_{a,b}(x_1) = h_{a,b}(x_2)) \leq \frac{p(p-1)/n}{p(p-1)} = \frac{1}{n}.$$

□

A **collision** occurs when two elements are hashed to the same bin.

Lemma

For any set set $S \subset U$ of size m, and $|V| = n$ there is a mapping (hash function) that maps S to V with no more than m^2/n collisions.

Proof.

Choose $h \in \mathcal{H}$ uniformly at random from a 2-universal family of hash functions mapping the universe U to $[0, n-1]$.
Let $s_1, s_2, \ldots, s_m$ be the m items of S.
Let X_{ij} be 1 if the $h(s_i) = h(s_j)$ and 0 otherwise. Let $X = \sum_{1 \leq i < j \leq n} X_{ij}$.

$$\mathbf{E}[X] = \mathbf{E}\left[\sum_{1 \leq i < j \leq n} X_{ij}\right] = \sum_{1 \leq i < j \leq m} \mathbf{E}[X_{ij}] \leq \binom{m}{2}\frac{1}{n} < \frac{m^2}{2n},$$

This implies that there exists a hash function with this property. $\square$

Note: a random hash function in the family has this property with probability $\geq 1/2$.

Two-level Approach

1. Hash the m elements to n bins using a hash function with a total of m collisions.
2. Each bin with $t \geq 2$ elements is replaced by a table with t^2 slots and a hash function for the t elements into the t^2 slots with no collisions.

Theorem

The two-level approach gives a perfect hashing scheme for m items using $O(m)$ bins.

Proof.

There exists a choice of a hash function in the first stage that gives at most m collisions.
Let c_i be the number of items in the i-th bin, then there are $\binom{c_i}{2}$ collisions between items in the i-th bin.

$$\sum_{i=1}^{m} \binom{c_i}{2} \leq m.$$

For each bin with $c_i > 1$ items, we find a second hash function that gives no collisions using space c_i^2. The total number of space used is

$$m + \sum_{i=1}^{m} c_i^2 \leq m + 2\sum_{i=1}^{m} \binom{c_i}{2} + \sum_{i=1}^{m} c_i \leq m + 2m + m = 4m.$$

$\square$

The Lovasz Local Lemma

Let $A_1, ..., A_n$ be a set of "bad" events. We want to show that

$$Pr(\cap_{i=1}^{n} \bar{A}_i) > 0.$$

1. If $\sum_{i=1}^{n} Pr(A_i) < 1$ then $Pr(\cap_{i=1}^{n} \bar{A}_i) > 0.$
2. If all the A_i's are mutually independent and for all i $Pr(A_i) < 1$ then $Pr(\cap_{i=1}^{n} \bar{A}_i) > 0.$.
3. If each A_i depends only on a few other events: *The Lovasz Local Lemma*.

Definition

An event E is mutually independent of the events $E_1, \ldots, E_n$, if for any $T \subset [1, \ldots, n]$,

$$Pr(E \mid \cap_{j \in T} E_j) = Pr(E).$$

Definition

A dependency graph for a set of events $E_1, \ldots, E_n$ has n vertices $1, \ldots, n$. Events E_i is mutually independent of any set of events $\{E_j \mid j \in T\}$ iff there is no edge in the graph connecting i to any $j \in T$.

Theorem

Let $E_1, ..., E_n$ *be a set of events. Assume that*

1. *For all* i, $Pr(E_i) \leq p$;
2. *The degree of the dependency graph is bounded by* d.
3. $4dp \leq 1$

then

$$Pr(\cap_{i=1}^{n} \bar{E}_i) > 0.$$

Let $S \subset \{1, ..., n\}$. We prove by induction on $s = 0, ..., n$ that if $|S| \leq s$, for all k

$$Pr(E_k \mid \cap_{j \in S} \bar{E}_j) \ \leq \ 2p.$$

For $s = 0$, $S = \emptyset$ obvious.
W.l.o.g. renumber so that $S = \{1, ..., s\}$, and (k, j) is not and edge of the dependency graph for $j > d$.

$$Pr(E_k \mid \bar{E}_1 ...., \bar{E}_s) = \frac{Pr(E_k \bar{E}_1 .... \bar{E}_s)}{Pr(\bar{E}_1 .... \bar{E}_s)}$$

$$Pr(E_k \bar{E}_1 ...., \bar{E}_s) =$$

$$Pr(E_k \bar{E}_1 ...., \bar{E}_d \mid \bar{E}_{d+1}, ..., \bar{E}_s) Pr(\bar{E}_{d+1}, ..., \bar{E}_s)$$

$$Pr(\bar{E}_1 ...., \bar{E}_s) =$$

$$Pr(\bar{E}_1 ...., \bar{E}_d \mid \bar{E}_{d+1}, ..., \bar{E}_s) Pr(\bar{E}_{d+1}, ..., \bar{E}_s)$$

$$Pr(E_k \mid \bar{E}_1 ...., \bar{E}_s) = \frac{Pr(E_k \bar{E}_1 ...., \bar{E}_d \mid \bar{E}_{d+1}, ..., \bar{E}_s)}{Pr(\bar{E}_1 ...., \bar{E}_d \mid \bar{E}_{d+1}, ..., \bar{E}_s)}$$

$$Pr(E_k \bar{E}_1 ...., \bar{E}_d \mid \bar{E}_{d+1}, ..., \bar{E}_s)$$

$$\leq Pr(E_k \mid \bar{E}_{d+1}, ..., \bar{E}_s) = Pr(E_k) \leq p.$$

Using the induction hypothesis we prove:

$$Pr(\bar{E}_1 ...., \bar{E}_d \mid \bar{E}_{d+1}, ..., \bar{E}_s)$$

$$\geq 1 - \sum_{i=1}^{d} Pr(E_i \mid \bar{E}_{d+1}, ..., \bar{E}_s) \geq 1 - \sum_{i=1}^{d} 2p \geq 1 - 2pd \geq 1/2.$$

$$Pr(E_k \mid \bar{E}_1 ...., \bar{E}_s) \leq \frac{p}{1/2} = 2p$$

proving the induction hypothesis.

$$Pr(\bar{E}_1, ..., \bar{E}_n) = \Pi_{i=1}^{n} Pr(\bar{E}_i \mid \bar{E}_1 ...., \bar{E_{i-1}})$$

$$= \Pi_{i=1}^{n} \left(1 - Pr(E_i \mid \bar{E}_1 ...., \bar{E_{i-1}})\right) \geq \Pi_{i=1}^{n} (1 - 2p) > 0.$$

Application: Edge-Disjoint Paths

Assume that n pairs of users need to communicate using edge-disjoint paths on a given network.
Each pair $i = 1, \ldots, n$ can choose a path from a collection F_i of m paths.

Theorem

If for each $i \neq j$, any path in F_i shares edges with no more than k paths in F_j, where $\frac{8nk}{m} \leq 1$, then there is a way to choose n edge-disjoint paths connecting the n pairs.

Proof

Consider the probability space defined by each pair choosing a path independently uniformly at random from its set of m paths.
$E_{i,j}$ = the paths chosen by pairs i and j share at least one edge.
A path in F_i shares edges with no more than k paths in F_j,

$$p = \Pr(E_{i,j}) \le \frac{k}{m}.$$

Let d be the degree of the dependency graph.
Since event $E_{i,j}$ is independent of all events $E_{i',j'}$ when $i' \notin \{i,j\}$ and $j' \notin \{i,j\}$, we have $d < 2n$.

$$4dp < \frac{8nk}{m} \le 1$$

$$\Pr(\cap_{i \neq j} \bar{E}_{i,j}) > 0.$$

Theorem

Consider a CNF formula with k *literals per clause. Assume that each variable appears is no more than* $T = \frac{2^k}{4k}$ *clauses, then the formula has a satisfying assignment,*

Proof.

Assume that the formula has m clauses.
For $i = 1, ..., m$, let E_i be the event "The random assignment does not satisfy clause i".

$$Pr(E_i) = \frac{1}{2^k}.$$

The event E_i is mutually independent of all the events related to clauses that do not share variables with clause i.
The degree of E_i in the dependency graph is bounded by kT.
Since

$$4dp \leq 4kT2^{-k} = 4k\frac{2^k}{4k}2^{-k} \leq 1$$

$$Pr(\bar{E_i}, ..., \bar{E_m}) > 0.$$

$\square$

Algorithm

Assume m clauses, ℓ variables, each clause has k literals, each variable appears in no more than $T = 2^{\alpha k}$ clauses.

First Part:

A clause is **Dangerous** at a given step if both

1. The clause is not satisfied;
2. At least $k/2$ of its variables were fixed.

For $i = 1$ to ℓ

If x_i is not in a dangerous clause assign it a random value in $\{0, 1\}$.

A **surviving clause** is a clause that is not satisfied at the end of phase one.

A surviving clause has no more than $k/2$ of its variables fixed.

A **deferred** variable is a variable that was no assigned value in the first part.

Lemma

There is an assignment of values to the deferred variables such that all the surviving clauses are satisfied (thus the formula is satisfied).

Lemma

Let G' be the dependency graph on the surviving clauses. With high probability all connected components in G' have size $O(\log m)$.

Part Two:

Using exhaustive search assign values to the deferred variable to complete the truth assignment for the formula.

If a connected component has $O(\log m)$ clauses it has $O(k \log m)$ variables. Assuming $K = O(1)$ we can check all assignments in polynomial in m number of steps.

Lemma

There is an assignment of values to the deferred variables such that all the surviving clauses are satisfied (thus the formula is satisfied).

At the end of the first phase we have m' “surviving clauses' (all the rest are satisfied), each surviving clause has at least $k/2$ deferred variables.
Consider a random assignment of the deferred variables.
Let E_i be the event clause i (of the surviving clauses) is not satisfied.

$$p = Pr(E_i) \leq 2^{-k/2}.$$

The degree of the dependency graph is bounded by

$$d = kT \leq k2^{\alpha k}.$$

Since

$$4dp = 4k2^{\alpha k}2^{-k/2} \leq 1$$

there is a satisfying assignment of the deferred variables that (together with the assignment of the other variables) satisfies the formula.

Lemma

Let G' be the dependency graph on the surviving clauses. With high probability all connected components in G' have size $O(\log m)$.

Assume that there is a connected component R of size $r = |R|$.
Since the degree of a vertex in R is bounded by d, there must be a set T of $t = r/d^3$ vertices in R which are at distance at least 4 from each other.
A clause "survives" the first part if it is at distance at most 1 from a dangerous clause. Thus, for each clause in T there is a **distinct** dangerous clause, and these dangerous clauses are at distance 2 from each other.
The probability that a given clause is dangerous is at most $2^{-k/2}$.
The probability that a clause survives is at most $(d+1)2^{-k/2}$.
These events are independent for vertices in T. Thus the probability of a particular connected component of r vertices is bounded by

$$\left((d+1)2^{k/2}\right)^{r/d^3}$$

How many possible connected components of size r are in a graph of m nodes and maximum degree d?

Lemma

There are no more than md^{2r} possible connected components of size r in a graph of m vertices and maximum degree d.

Proof.

A connected component of size r has a spanning tree of $r-1$ edges.
We can choose a "root" for the tree in m ways.
A tree can be defined by an Euler tour that starts and ends at the root and traverses each edge twice.
At each node the tour can continue in up to d ways. Thus, for a given root there are no more than d^{2r} different Euler tours. □

Thus, the probability that at the end of the first phase there is a connected component of size $r = \Omega(\log m)$ is bounded by

$$md^{2r}\left((d+1)2^{-k/2}\right)^{r/d^3} = o(1)$$

for $d = k2^{\alpha k}$, $\alpha > 0$ sufficiently small.

Each deferred variable appears in only one component. A component of size $O(\log m)$ has only $O(\log m)$ variables. Thus, we can enumerate (try) all possibilities in time polynomial in m.

Theorem

Given a CNF formula of m clauses, each clause has $k = O(1)$ literals, each variables appears in up to $2^{\alpha k}$ clauses. For a sufficiently small $\alpha > 0$ there is an algorithm that finds a satisfying assignment to the formula in time polynomial in m.