Lecture 3: Fuzzing, Counting, and Sampling

Tuesday, August 6, 2019 9:31 AM

Follow along on
http://tiny.cc/91ssaz
https://1drv.ms/o/s!AufrpyzkMX1pcZcVPUGorRH7xMI

Fuzzing Algorithm

T = initial set of (randomly chosen) starting points

repeat
- t = pick(T) → do a local search around t
- t' = mutate(t)
- if t' crashes : T_{bug} += t
- if t' is interesting, add t' to T

Boolean Satisfiability

Given: a Boolean formula φ in conjunctive normal form over n Boolean variables $x_1 \ldots x_n$ and m clauses

Is there a truth assignment to $x_1 \ldots x_n$ s.t. φ evaluates to true.

Ex: $\varphi = (x_1 \vee \bar{x}_2) \wedge (\bar{x}_1 \vee \bar{x}_3) \wedge (x_1 \vee x_2) \wedge (x_4 \vee \bar{x}_3) \wedge (x_4 \vee \bar{x}_1)$ ← 2SAT

(clause)

$x_1 \to 1 \quad x_2 \to 1 \quad x_3 \to 0 \quad x_4 \to 1$

k-SAT → each clause has exactly k literals

Th: SAT is NP-complete. 3SAT is NP-complete. k SAT, $k \geq 3$ . . .

2SAT is in poly time.

A fuzzing approach to SAT

- Start with an arbitrary truth assignment
- Repeat up to ℓ times, $\ell = 2n^2$ terminating if a sat. assn. has been found, the following steps.:
  - (1) Pick an arbitrary clause C that is not satisfied
  - (2) Pick u.a.r. one of the literals of C and switch the value of its variable.
- if you have found a sat. assn. then return "SAT" o/w return "Maybe UNSAT"

Let φ be a 2SAT formula. [Papadimitriou]

Let S be a sat. assignment.

A_i is the truth assignment after i iterations

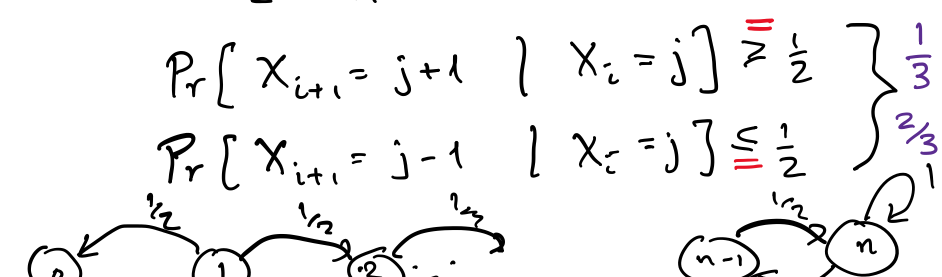
X_i : # of variables in A_i with the same values as in S.

What can we say about one iteration ?

→ $\Pr[X_{i+1} = 1 \mid X_i = 0] = 1$

$\Pr[X_{i+1} = j+1 \mid X_i = j] \geq \frac{1}{2}$ ($\frac{1}{3}$)

$\Pr[X_{i+1} = j-1 \mid X_i = j] \leq \frac{1}{2}$ ($\frac{2}{3}$)



Let h_j is the expected # of steps to reach n when starting from j

$h_n = 0$

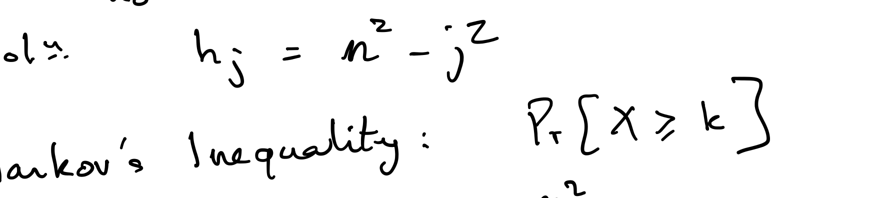
$h_j = \frac{1}{2}(h_{j-1} + h_{j+1}) + 1 \qquad j \in \{1 \ldots n-1\}$

$h_0 = h_1 + 1$

Sol^n: $h_j = n^2 - j^2$

Markov's Inequality: $\Pr[X \geq k] \leq \frac{E[X]}{k}$, X is a positive rv

$\Pr[h \geq 2n^2] \leq \frac{n^2}{2n^2} = \frac{1}{2}$ (k)



Ex: φ: $x_1 \wedge x_2 \wedge \ldots \wedge x_n \wedge \bigwedge_{i,j,k} (x_i \vee \neg x_j \vee \neg x_k)$

$h_n = 0$

$h_j = \frac{2}{3} h_{j-1} + \frac{1}{3} h_{j+1} + 1$

$h_0 = h_1 + 1$

Unique sol^n $h_j = 2^{n+2} - 2^{j+2} - 3(n-j)$

[Schöning's algo]

If A_0 is chosen uar

$\Pr[X_0 = j] = \binom{n}{j}\left(\frac{1}{2}\right)^n \leftarrow E[X_0] = \frac{n}{2}$

Repeat ℓ times, terminating if a sat. assn. is found
- (a) Pick an assn. uar.
- (b) Repeat 3n steps,
  - (1) Pick an arbitrary unsat. C
  - (2) Pick a literal of C uar and flip the variable

The prob of exactly k left moves and k+j right moves in a seq of j+2k moves is

$$\binom{j+2k}{k}\left(\frac{2}{3}\right)^k\left(\frac{1}{3}\right)^{j+k}$$

Let q_j be a lower bound on the Pr that the algorithm reaches n when it starts at node n−j

$$q_j \geq \max_{k \in \{0..j\}} \binom{j+2k}{k}\left(\frac{2}{3}\right)^k\left(\frac{1}{3}\right)^{j+k}$$

$$q_j \geq \binom{3j}{j}\left(\frac{2}{3}\right)^j\left(\frac{1}{3}\right)^{2j}$$

Stirling's appx

$$\binom{3j}{j} = \frac{(3j)!}{j!(2j)!} \geq \frac{\sqrt{2\pi 3j}\left(\frac{3j}{3}\right)^{3j}}{4\sqrt{2\pi j}\sqrt{2\pi(2j)}\left(\frac{e}{2j}\right)^{2j}\left(\frac{e}{j}\right)^j}$$

$$\sim \text{const.} \frac{1}{\sqrt{j}} \cdot \left(\frac{27}{4}\right)^j$$

So : $q_j \geq a \cdot \frac{1}{\sqrt{j}} \frac{1}{2^j} \qquad q_0 = 1$

$$q \geq \sum_{j=0}^{n} \Pr[X_0 = n-j] \cdot q_j$$

$$\geq \underbrace{\frac{1}{2^n}}_{j=0} + \sum_{j=1}^{n} \underbrace{\binom{n}{j}\left(\frac{1}{2}\right)^n}_{\Pr[X_0 = n-j]} \cdot \underbrace{a \frac{1}{\sqrt{j}} \frac{1}{2^j}}_{q_j} \approx \frac{\text{Const}}{\text{poly}(n)}\left(\frac{3}{4}\right)^n$$

The expected overall # of assignments that you consider $\sim \frac{1}{q}$

$$\sim \left(\frac{4}{3}\right)^n$$

1.34^n

Counting the number of satisfying assignments

(1) #SAT : Given a Boolean formula φ count how many sat. assn. it has.

#P = "Counting version" of NP problems

(2) Counting ~ Sampling

$\varphi(x_1 \ldots x_n) = x_1 \wedge \varphi(1, x_2 \ldots x_n) \vee \bar{x}_1 \wedge \varphi(0, x_2 \ldots x_n)$

$\varphi(x_1 \ldots x_n)$ is sat iff $\varphi(0, x_2 \ldots x_n)$ is sat or $\varphi(1, x_2 \ldots x_n)$ is sat

$\#\varphi = \#\varphi[x_1 \to 0] + \#\varphi[x_1 \to 1]$

To randomly sample the sat. assn.

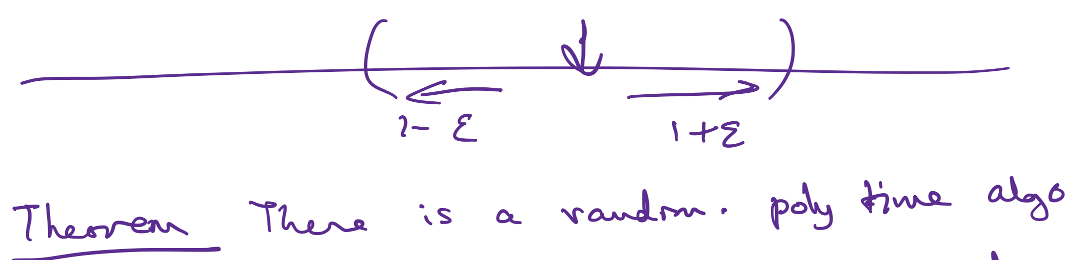
Pick $x_1 \to 0$ with prob $\frac{\#\varphi[x_1 \to 0]}{\#\varphi}$

and $x_1 \to 1$ " " $\frac{\#\varphi[x_1 \to 1]}{\#\varphi}$

and then recur.

An algo. $A(\varphi, \varepsilon)$ is an appx counter if

$$\left\{\frac{\#\varphi}{1+\varepsilon} \leq A(\varphi, \varepsilon) \leq \#\varphi(1+\varepsilon)\right\}$$ with high prob



Theorem There is a random. poly time algo A with access to a SAT procedure that is an appx counter for SAT.

Arora/Barak → #P

Valiant-Vazirani