Algorithms for k-SAT

Variables: $x_1, \dots, x_N$

Assignment: $\alpha \in \{0,1\}^N$

Literals: $x_i \,/\, \overline{x_i}$ satisfied if $\alpha_i = 1 \,/\, \alpha_i = 0$

k-Clause: $\ell_1 \vee \dots \vee \ell_k$ satified if some lit. is sat.

k-CNF: $C_1 \wedge \dots \wedge C_M$ satisfied if all clauses are sat.

k-SAT Problem

Input: A k-CNF formula with N vars and M clauses

Output: Is there a satisfying assignment?

$$\boxed{P \neq NP \iff \text{3-SAT is not in time } poly(N)}$$

Trivial algorithm: Test all assignments

$\Rightarrow$ time $O(2^N \, poly(N))$

Can we do better? At least for 3-SAT? Let's try!

These questions were asked before fine-grained complexity existed in the 1990's – 2000's

Algorithm 1

Take an arbitrary clause $C = (\ell_1 \vee \ell_2 \vee \ell_3)$
Try all 7 satisfying assignments of C, and
recur on the rest

$$T(N) \le 7 \cdot T(N-3) + \text{poly}(N)$$
$$= O(7^{N/3}\,\text{poly}(N))$$
$$\le O(1.913^N \cdot \text{poly}(N))$$

Algorithm 2 [Monien, Speckenmeyer '85]

Take an arbitrary clause $C = (\ell_1 \vee \ell_2 \vee \ell_3)$
Guess the first satisfied literal.

$$T(N) \le T(N-1) + T(N-2) + T(N-3)$$

By induction: $T(N) \le O(\alpha^N \text{poly}(N)) \le O(1.840^N \text{poly}(N))$
where

$$\alpha^N \le \alpha^{N-1} + \alpha^{N-2} + \alpha^{N-3}$$
$$\Leftrightarrow \alpha^3 \le \alpha^2 + \alpha + 1 \Rightarrow \alpha \le 1.840$$

Algorithm 3: Random Walk [Schöning '99]

Repeat $(4/3)^N \text{poly}(N)$ times:

- Start with random assignment $\alpha \in \{0,1\}^N$
- Repeat $3N$ times:
  - If α is satisfying: done
  - Take an unsatisfied clause $C = (\ell_1 \vee \ell_2 \vee \ell_3)$
  - Randomly flip ℓ_1, ℓ_2, or ℓ_3 in α

Intuition: We perform a random walk on assignments that is biased towards satisfying clauses.

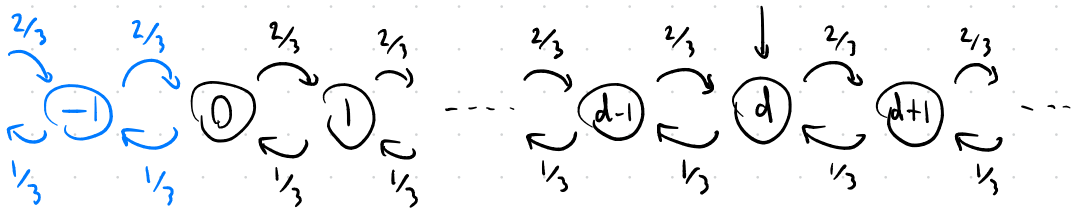
Analysis

"distance" d = Hamming distance from the initial α to closest satisfying assignment

<u>Claim:</u> $\mathbb{P}(\text{algo finds sat. assignment}) \geq 2^{-d}$

<u>Proof:</u>

Distance to satisfying is a <u>random walk</u>:



$$\mathbb{P}(\text{algo finds sat. assignment in} \leq 3N \text{ steps})$$

$$\geq \mathbb{P}(\text{algo finds sat. assignment in} = 3d \text{ steps})$$

$$\geq \mathbb{P}(2d \text{ steps left and } d \text{ steps right})$$

$$= \binom{3d}{d} \cdot \left(\tfrac{1}{3}\right)^{2d} \cdot \left(\tfrac{2}{3}\right)^{d}$$

$$\approx 3^{d} \cdot \left(\tfrac{3}{2}\right)^{2d} \cdot \left(\tfrac{1}{3}\right)^{2d} \cdot \left(\tfrac{2}{3}\right)^{d} \qquad \binom{n}{\alpha n} \approx \left(\frac{1}{\alpha}\right)^{\alpha n} \left(\frac{1}{1-\alpha}\right)^{(1-\alpha) n}$$

$$= 2^{-d}$$

$\square$

$\Longrightarrow$ Need to repeat the algorithm

First try:

Distance $d \leq \frac{N}{2}$ with prob. $\geq 1/2$.

$\Rightarrow$ Then succeeds with prob $\geq 2^{-d} \geq 2^{-N/2}$

$\Rightarrow$ $100 \cdot 2^{N/2}$ repetitions suffice

$\Rightarrow$ Time $O(2^{N/2} \text{poly}(N)) = O(1.415^N \text{poly}(N))$

Optimization:

$$
\begin{aligned}
&\mathbb{P}(\text{algo succeeds}) \\
&\geq \sum_{k=0}^{N} \mathbb{P}(\text{distance } d = k) \cdot 2^{-k} \\
&= \sum_{k=0}^{N} \frac{\binom{N}{k}}{2^N} \cdot \frac{1}{2^k} \\
&= \frac{1}{2^N} \cdot \left(\tfrac{1}{2} + 1\right)^N \\
&= (3/4)^N
\end{aligned}
$$

Binomial theorem!

$$(a+b)^N = \sum_{k=0}^{N} \binom{N}{k} a^k b^{N-k}$$

$\Rightarrow$ $100\,(4/3)^N$ repetitions suffice

$\Rightarrow$ Time $O((4/3)^N \cdot \text{poly}(N))$

Literature on 3-SAT Algorithms

1.619^N	[Monien, Speckenmeyer '85]	Branching-based
1.476^N	[Rodošek '96]	
1.5^N	[Schöning '99]	Local Search
1.333^N	[Schöning '02]	Random Walk
1.308^N	[Paturi, Pudlák, Zane '97]	Resolution-based
	[Paturi, Pudlák, Saks, Zane '98]	
	[Hertli '11]	

Take-away: There are nontrivial algorithms for 3-SAT!

But: Even though we have tried many techniques...
And the algorithms become very complicated...
We are still stuck at exponential time!

So perhaps 3-SAT is truly exponential-time-hard?

Exponential Time Hypothesis (ETH)
There is some $\delta > 0$ s.t. 3-SAT cannot be solved in time $O(2^{\delta N})$.

What about k-SAT?

- Branching-based techiques:

$$O\left(2^{(1-\frac{c}{2^k})N}\right)$$ (for some constant c)

- Local search,
- Random walks
- Resolution-based techniques
- Polynomial Method
- ...:

fine-grained algorithms technique

$$O\left(2^{(1-\frac{c}{k})N}\right)$$ (for some constant c)

Take-away: For $k \to \infty$ we do not know nontrivial algorithms for k-SAT!

Strong Exponential Time Hypothesis (SETH)
For any $\varepsilon > 0$ there is some $k \geq 3$ such that k-SAT cannot be solved in time $O(2^{(1-\varepsilon)N})$.

Exercise: SETH $\Rightarrow$ ETH