Algorithms from (proofs of)

Circuit Lower Bounds

**Thm** [..., Impagliazzo, Matthews, Paturi]

$k$-SAT $\in \mathsf{ZPTIME}(2^{n(1-\frac{1}{40k})})$

($\varphi : k$-CNF on $n$ variables)

Moreover, in the same time, can count $\#$ satisfying assignments!

Håstad's Switching Lemma

Before: $A k, t, p$

$A k$-CNF $\varphi$ on $n$ vars.

$\Pr\left[ \varphi \mid \varphi \text{ is not a } t$-$\mathsf{DNF} \right] < (5pk)^t$
Actually works for a particular
Simplification Procedure: \( \psi, \varphi \rightarrow \psi/\varphi \)
building a t-\(\chi\)NF for \(\psi/\varphi\).

\[ f(x, y, z, w) \]
\[ f(0, 0, *, *) = 0 \]
\[ f(0, 1, 0, *) = 1 \]
\[ f(1, *, 1, *) = 1 \]

Decision Trees

Observation: Depth-\(d\) Decision Tree for \(f\) \[ \Rightarrow f \text{ has } d-\chi\text{NF (and } d-\chi\text{CNF).} \]
Proof: OR of all branches with a leaf 1
Proof: \( \text{OR of all branches, with a depth 1 AND of literals on the branch (\( \leq d \))} \)

**Canonical Decision Tree for** a given \( K\text{-CNF} \ \varphi(x_1, \ldots, x_n) \)

\[ \varphi : (x \lor \overline{y} \lor w) \land (y \lor z) \land \left( \overline{x} \lor \overline{v} \lor \overline{z} \right) \]

\( x = 0 \)
\( y = 1 \)
\( w = 0 \)

The Canonical DT for a given \( \varphi \): \( \text{CST}(\varphi) \)
Hästad's Switching Lemma
(actual statement)

\[ \forall \kappa, t, p. \quad \forall K\text{-CNF } \varphi \text{ on } n \text{ vars.} \]

\[ \Pr \left[ \text{depth of } \text{CUT}(\varphi \mid S) > t \right] \leq (5p^k)^t \]

\[ p \sim R_p \]

\text{K-SAT Algorithm (} p = \frac{1}{20 \cdot \kappa} \text{)}

\text{Input:} \quad K\text{-CNF } \varphi(x_1, \ldots, x_n)

1. Form a Set \( S \subseteq \{1, 2, \ldots, n\} \):
   
   \text{for } i = 1 \text{ to } n \\
   \quad \text{put } i \text{ into } S \text{ with prob. } p

2. For every partial assignment \( a \) that sets all vars not in \( S \),
   
   \text{Check if } \varphi \mid a \quad (\text{in vars from } S)
is satisfiable: by computing $CST(\varphi_{1a})$.
If discover that $\varphi$ is satisfiable,
output 'Yes'; otherwise, 'No'.

Observation: Algo is always correct.
In the worst case, it looks at all $2^n$ assignments to see if $\varphi$ is satisfiable.
BUT, actually, quite often $\varphi_{1a}$ is much simpler & hence is faster to check for being SAT.

Claim: The described randomized algo runs in Expected Time

\[ n \cdot \left( 1 - \frac{1}{90.\kappa} \right)^2 \]

Proof: (over simplified)
For a given input \( k \)-CNF \( \varphi(x_1, \ldots, x_n) \), define a random variable

\[ T(S) = \text{runtime of our algo given a random set } S \subseteq \{1, \ldots, b\} \]

Want to upper-bound:

\[ \mathbb{E}_S \left[ T(S) \right] \]

\[ T(S) = \sum_{b: \ \text{SAT check via CST construction}} \left| \text{CST}(\varphi|_{S,b}) \right| \cdot \text{poly}(n) \]

\[ \text{Assume } |S| = n \cdot p \text{ (the expected size)} \]

\[ T(S) \leq 2^{n-np} \cdot \mathbb{E}_b \left[ 2^{\text{depth-CST}(\varphi|_{S,b})} \right] \cdot \text{poly}(n) \]

\[ \mathbb{E}_S \left[ T(S) \right] \leq \left( n(1-p) - \text{depth-CST}(\varphi|_{S,b}) \right) \]
\[\begin{align*}
\mathcal{S} &= \text{poly}(n) \cdot \exp \left[ \frac{n(1-p)}{2} \cdot \exp \left[ 2^{\text{depth-CST}(\psi|_{\mathcal{S},b})} \right] \right] \\
&= \text{poly}(n) \cdot 2^{n(1-p)} \cdot \exp \left[ 2^{\text{depth-CST}(\psi|_{\mathcal{S},b})} \right] \\
&= \text{poly}(n) \cdot 2^{n(1-p)} \cdot \exp \left[ 2^{\text{depth-CST}(\psi|_{\mathcal{S}})} \right] \\
&= \text{poly}(n) \cdot 2^{n(1-p)} \cdot \exp \left[ 2^{\text{depth-CST}(\psi|_{\mathcal{S}})} \right]
\end{align*}\]

**Fact:** For every random variable \(X\) s.t.
1. \(X \geq 0\)
2. \(X\) is integer-valued,
\[
\exp [X] = \sum_{i=0}^{\infty} \Pr [X > i].
\]

(by Fact)
\[
\geq \sum_{i=1}^{\infty} \Pr \left[ \text{depth-CST}(\psi|_{s}) > \log_{2} i \right] \\
\leq (5pK) \log_{2} i
\]
Set \( p = \frac{1}{20 \cdot k} \). Then \((5p_k) = \frac{1}{4}\).

\[
\sum_{i=1}^{88} \frac{1}{2} \log_2 i \leq \sum_{i=1}^{88} \frac{1}{2} \cdot \frac{1}{i} \leq \frac{1}{2} \leq O(1)
\]

Overall,

\[
\text{Exp}_{S}[T(S)] \leq p^{(\frac{1}{2})/14} \cdot 2^{n(1-p)} \cdot O(1) = \text{poly}(n) \cdot 2^{n(1-\frac{1}{20 \cdot k})}
\]

This concludes the proof, subject to our over-simplifying assumption that \( |S| = p \cdot n \).

Actually, by Chernoff, \( |S| \geq \frac{p\cdot n}{2} \), w.h.p. + extra (technical) work \( \Rightarrow \) rigorous proof.
Remark: We can modify the described \textsc{K-Sat} algorithm so that it counts the number of satisfiable assignments it sees! The expected run-time stays the same!

Last time we saw:

\textbf{Algo:} always correct, expected run-time $T$

\textbf{Algo:} zero-error, run-time $\leq 4T$

Corollary: \#\textsc{K-Sat} is in \textsc{Zptime} $\left( 2^{\frac{n(1-\frac{1}{40k})}{2}} \right)$. 
Musings

\( K\text{-SAT} \) is (randomized) time \( 2^{\eta(1 - \frac{c}{k})} \) (const \( c > 0 \))

with a number of different algorithms

"whatever algo strategy you try, you only get \( 2^{\eta(1 - \frac{c}{k})} \) time"

Maybe it's a correct answer?

Exponential Time Hypothesis (ETH)

Exponential Time Hypothesis (ETH)

\[ \text{Exponential Time Hypothesis (ETH)} \]

K-SAT requires time \( 2^{\eta(1 - \frac{c}{k})} \):

\forall k \geq 3, \exists \delta = \delta(k) > 0 \text{ s.t.} \]

\[ K\text{-SAT} \not\in \text{BPTIME}(2^{\delta n}) \]

Strong ETH (SETH):

\[ \text{SAT requires time } 2^{\eta(1 - o(1))} \]
$K$-SAT requires Time $2^{\eta(1-o(1))}$ as $k \to \infty$, $\delta(k) \to 1$.

ETH & SETH strengthen $P \neq NP$ assumption:

many implication on precise time complexity (even for problems in $P$).