Combinatorial Properties of $k$-CNF
Connection to Upper and Lower Bounds

Mohan Paturi

University of California, San Diego

August 2015
Outline

1. Introduction
2. Satisfiability Coding Lemma
3. Sparsification Lemma
4. Switching Lemma
Motivation

- Faster Satisfiability Algorithms
- Circuit Lower Bounds
Problem: Prove stronger exponential lower bounds for depth-3 OR-AND-OR (ΣΠΣ) circuits. Also for depth-3 ΣΠΣ_k circuits with bottom fan-in bounded by k
Problem: Prove stronger exponential lower bounds for depth-3 OR-AND-OR (ΣΠΣ) circuits. Also for depth-3 ΣΠΣ_k circuits with bottom fan-in bounded by k.

What was known?

1. $2^{c\sqrt{n}}$ (for $c < 1/8$) for computing parity (Switching Lemma)
2. $2^{0.687\sqrt{n}}$ for computing parity (Top-down method)
Problem: Prove stronger exponential lower bounds for depth-3 OR-AND-OR ($\Sigma \Pi \Sigma$) circuits. Also for depth-3 $\Sigma \Pi \Sigma_k$ circuits with bottom fan-in bounded by $k$.

What was known?
1. $2^{c\sqrt{n}}$ (for $c < 1/8$) for computing parity (Switching Lemma)
2. $2^{0.687\sqrt{n}}$ for computing parity (Top-down method)

Parity can be computed by $O(n^{\frac{1}{4}}2^{\sqrt{n}})$ size depth-3 circuits of bottom fan-in $O(\sqrt{n})$. 
Problem: Prove stronger exponential lower bounds for depth-3 OR-AND-OR (ΣΠΣ) circuits. Also for depth-3 ΣΠΣ_\(k\) circuits with bottom fan-in bounded by \(k\).

What was known?

1. \(2^c\sqrt{n}\) (for \(c < 1/8\)) for computing parity (Switching Lemma)
2. \(2^{0.687\sqrt{n}}\) for computing parity (Top-down method)

Parity can be computed by \(O(n^{1/4}2^{\sqrt{n}})\) size depth-3 circuits of bottom fan-in \(O(\sqrt{n})\).

Better lower bounds?
Connections to Other Circuit Models

- Linear-size log-depth Boolean circuits of fan-in 2 $\rightarrow$ depth-3 circuits of size $2^{O(n/\log \log n)}$ and bottom fan-in $n^{o(1)}$
Connections to Other Circuit Models

- Linear-size log-depth Boolean circuits of fan-in 2 $\rightarrow$ depth-3 circuits of size $2^{O(n/\log \log n)}$ and bottom fan-in $n^{o(1)}$
- Linear-size log-depth series-parallel circuits $\rightarrow$ $\bigvee 2^{O(n/\log d)}$ linear size $2^d$-CNF
Connections to Other Circuit Models

- Linear-size log-depth Boolean circuits of fan-in 2 $\rightarrow$ depth-3 circuits of size $2^{O(n/\log \log n)}$ and bottom fan-in $n^{o(1)}$
- Linear-size log-depth series-parallel circuits $\rightarrow$ $\bigvee 2^{O(n/\log d)}$ linear size $2^d$-$\text{CNF}$
- $\text{NC}^1$ circuits of depth $k \log n$ $\rightarrow$ depth $d + 1$ unbounded fan-in Boolean circuits of size $2^{n^{k/d}}$ and bottom fan-in $n^{k/d}$
A Lower Bound Problem

- Prove that for a function in NP that it cannot be computed by bottom fan-in $k$ depth-3 circuits of size $2^{n/2}$ for any $k$
A Lower Bound Problem

- Prove that for a function in \textbf{NP} that it cannot be computed by bottom fan-in $k$ depth-3 circuits of size $2^{n/2}$ for any $k$.
- An even weaker open problem: proving a size lower bound of $2^{2n/k}$ on depth-3 circuits with bottom fan-in at most $k$. Or proving a size lower bound of $2^{2\sqrt{n}}$ for depth-3 circuits without any bottom fan-in restriction.
A Lower Bound Problem

- Prove that for a function in \textbf{NP} that it cannot be computed by bottom fan-in $k$ depth-3 circuits of size $2^{n/2}$ for any $k$.

- An even weaker open problem: proving a size lower bound of $2^{2n/k}$ on depth-3 circuits with bottom fan-in at most $k$. Or proving a size lower bound of $2^{2\sqrt{n}}$ for depth-3 circuits without any bottom fan-in restriction.

- A more immediate challenge: prove a $2^{n/k}$ size lower bound for computing parity with depth-3 circuits of bottom fan-in $k$ and a $2^{\sqrt{n}}$ size lower bound for circuits without any restriction on bottom fan-in.
Satisfiability

- Input: a formula or circuit $F$ on $n$ variables.
Satisfiability

- Input: a formula or circuit $F$ on $n$ variables.
- Check if $F$ is satisfiable
Satisfiability

- Input: a formula or circuit $F$ on $n$ variables.
- Check if $F$ is satisfiable
- Examples for $F$: $k$-CNF, CNF, formula, $\text{AC}^0$ circuit, $\text{NC}^1$ circuit, polynomial size circuit
Satisfiability

- Input: a formula or circuit $F$ on $n$ variables.
- Check if $F$ is satisfiable.
- Examples for $F$: $k$-CNF, CNF, formula, $AC^0$ circuit, $NC^1$ circuit, polynomial size circuit.
- Decidable in $|F|2^n$ time.
Satisfiability

- Input: a formula or circuit $F$ on $n$ variables.
- Check if $F$ is satisfiable
- Examples for $F$: $k$-CNF, CNF, formula, $\mathbf{AC}^0$ circuit, $\mathbf{NC}^1$ circuit, polynomial size circuit
- Decidable in $|F|2^n$ time.
- Can we improve upon the exhaustive search? Can we obtain a $|F|2^{n(1-\mu)}$ bound for $\mu > 0$?
Satisfiability

- Input: a formula or circuit $F$ on $n$ variables.
- Check if $F$ is satisfiable.
- Examples for $F$: $k$-CNF, CNF, formula, $AC^0$ circuit, $NC^1$ circuit, polynomial size circuit.
- Decidable in $|F|2^n$ time.
- Can we improve upon the exhaustive search? Can we obtain a $|F|2^{n(1-\mu)}$ bound for $\mu > 0$?
- $\mu$ is called the satisfiability savings. $\mu$ can be a function of the parameters of the class of formulas/circuits and $n$, the number of variables.
Satisfiability

- Input: a formula or circuit $F$ on $n$ variables.
- Check if $F$ is satisfiable
- Examples for $F$: $k$-CNF, CNF, formula, $\mathbf{AC}^0$ circuit, $\mathbf{NC}^1$ circuit, polynomial size circuit
- Decidable in $|F|2^n$ time.
- Can we improve upon the exhaustive search? Can we obtain a $|F|2^{n(1-\mu)}$ bound for $\mu > 0$?
- $\mu$ is a called the satisfiability savings. $\mu$ can be a function of the parameters of the class of formulas/circuits and $n$, the number of variables.
- What is the savings for the class of $k$-CNF formulas?
Satisfiability

- Input: a formula or circuit $F$ on $n$ variables.
- Check if $F$ is satisfiable
- Examples for $F$: $k$-CNF, CNF, formula, $AC^0$ circuit, $NC^1$ circuit, polynomial size circuit
- Decidable in $|F|2^n$ time.
- Can we improve upon the exhaustive search? Can we obtain a $|F|2^{n(1-\mu)}$ bound for $\mu > 0$?
- $\mu$ is called the satisfiability savings. $\mu$ can be a function of the parameters of the class of formulas/circuits and $n$, the number of variables.
- What is the savings for the class of $k$-CNF formulas?
- Earlier (to 1997) results showed that $\mu$ is $\Omega(1/2^k)$.
Let $C$ is a $\Sigma\Pi\Sigma_k$ circuit with top fan-in $s$. 

Let $C$ compute the parity function $\rightarrow$ one of the $s_k$-cnf. 

$\rightarrow$ must accept at least $\Omega(2^{n/s})$ many inputs of odd parity and accept no input of even parity.

Argue that a $k$-cnf cannot accept too many such inputs while avoiding all inputs of even parity.
Let $C$ be a $\Sigma\Pi\Sigma_k$ circuit with top fan-in $s$

Let $C$ compute the parity function $\longrightarrow$ one of the $s \ k$-CNFs must accept at least $\Omega(2^n/s)$ many inputs of odd parity and accept no input of even parity.
Let $C$ is a $\Sigma \Pi \Sigma_k$ circuit with top fan-in $s$

Let $C$ compute the parity function $\longrightarrow$ one of the $s \ k$-$\text{CNFS}$ must accept at least $\Omega(2^n/s)$ many inputs of odd parity and accept no input of even parity.

Argue that a $k$-$\text{CNF}$ cannot accept too many such inputs while avoiding all inputs of even parity.
Isolated Solutions

A satisfying solution for $F$ is isolated if all its distance 1 neighbors are not solutions.
Isolated Solutions

A satisfying solution for $F$ is isolated if all its distance 1 neighbors are not solutions.

What is the maximum number of isolated solutions for a $k$-CNF?
Isolated Solutions

- A satisfying solution for $F$ is isolated if all its distance 1 neighbors are not solutions.
- What is the maximum number of isolated solutions for a $k$-CNF?
- We show that this number is at most $2^n(1-1/k)$
Critical Clauses

- Let $F$ be a $k$-CNF and $x$ be an isolated satisfying solution of $x$. 
Let $F$ be a $k$-CNF and $x$ be an isolated satisfying solution of $x$.

For each variable $i$ and isolated solution $x$, $F$ must have a clause with exactly one true literal corresponding to the variable $i$ at solution $x$. 
Critical Clauses

- Let $F$ be a $k$-CNF and $x$ be an isolated satisfying solution of $x$.
- For each variable $i$ and isolated solution $x$, $F$ must have a clause with exactly one true literal corresponding to the variable $i$ at solution $x$.
- Such clause is called a critical clause for the variable $i$ at the solution $x$. 
Compressing Isolated Satisfying Solutions

- Let $F$ be a $k$-CNF and $\sigma$ a permutation of $\{1, \cdots, n\}$.
Compressing Isolated Satisfying Solutions

- Let $F$ be a $k$-CNF and $\sigma$ a permutation of $\{1, \ldots, n\}$.
- Let $x \in \{0, 1\}^n$ be an isolated satisfying solution of $F$.
- Compression Function $F_\sigma$: 
  
  1. Permute the bits of $x$ according to $\sigma$.
  2. For each $i$, delete the $i$'th bit of $x$ if all other variables in a critical clause $C_{x, \sigma}(i)$ (for the variable $\sigma(i)$ at $x$) occur before the variable $\sigma(i)$ in the order $\sigma$. 
  3. $F_\sigma(x)$ is the resulting compressed string.
Compressing Isolated Satisfying Solutions

- Let $F$ be a $k$-CNF and $\sigma$ a permutation of $\{1, \cdots, n\}$.
- Let $x \in \{0, 1\}^n$ be an isolated satisfying solution of $F$.
- **Compression Function** $F_\sigma$:
  1. Permute the bits of $x$ according to $\sigma$
Compressing Isolated Satisfying Solutions

- Let $F$ be a $k$-CNF and $\sigma$ a permutation of $\{1, \ldots, n\}$.
- Let $x \in \{0, 1\}^n$ be an isolated satisfying solution of $F$.
- **Compression Function** $F_\sigma$:
  1. Permute the bits of $x$ according to $\sigma$.
  2. For each $i$, delete the $i$'th bit of $x$ if all other variables in a critical clause $C_{x,\sigma(i)}$ (for the variable $\sigma(i)$ at $x$) occur before the variable $\sigma(i)$ in the order $\sigma$. 

Paturi

Properties of $k$-CNF
Compressing Isolated Satisfying Solutions

- Let $F$ be a $k$-CNF and $\sigma$ a permutation of $\{1, \cdots, n\}$.
- Let $x \in \{0, 1\}^n$ be an isolated satisfying solution of $F$.
- **Compression Function** $F_\sigma$:
  1. Permute the bits of $x$ according to $\sigma$.
  2. For each $i$, delete the $i$’th bit of $x$ if all other variables in a critical clause $C_{x, \sigma(i)}$ (for the variable $\sigma(i)$ at $x$) occur before the variable $\sigma(i)$ in the order $\sigma$.
  3. $F_\sigma(x)$ is the resulting compressed string.
We can recover $x$ from $y = F_\sigma(x)$, $F$, and $\sigma$. 
$F_\sigma$ is Lossless

- We can recover $x$ from $y = F_\sigma(x)$, $F$, and $\sigma$.
- Decompression Algorithm:

1. $F_1 = F$
2. \textbf{for} $i = 1, \ldots, n$
3. \hspace{1em} \textbf{if} $F_i$ has a clause of length one with the variable $\sigma(i)$,
4. \hspace{1em} \textbf{then} set the variable $\sigma(i)$ so that the clause is true
5. \hspace{1em} \textbf{else} set the variable $\sigma(i)$ to the next unused bit of $y$.
6. $F_{i+1} = \text{substitute for } \sigma(i) \text{ in } F \text{ and simplify}$
Satisfiability Coding Lemma

Lemma (Satisfiability Coding Lemma)

If $x$ is an isolated solution of a $k$-CNF $F$, then its average (over all permutations $\sigma$) compressed length $|F_\sigma(x)|$ is at most $n(1 - 1/k)$.

Proof Sketch: For each variable $i$ with a critical clause at $x$, the probability (under a random permutation) $i$ appears last among all the variables in its critical clause is at least $1/k$. 

Satisfiability Coding Lemma

Lemma (Satisfiability Coding Lemma)

If $x$ is an isolated solution of a $k$-CNF $F$, then its average (over all permutations $\sigma$) compressed length $|F_\sigma(x)|$ is at most $n(1 - 1/k)$.

Proof Sketch: For each variable $i$ with a critical clause at $x$, the probability (under a random permutation) $i$ appears last among all the variables in its critical clause is at least $1/k$.

The compression algorithm deletes $n/k$ bits on average.
Maximum Number of Isolated Solutions

**Lemma**

A $k$-CNF can have at most $2^n(1-1/k)$ isolated solutions.

Proof Sketch:

- For every isolated solution, the average (over permutations) compressed length is at most $n - n/k$
Maximum Number of Isolated Solutions

Lemma

A $k$-CNF can have at most $2^{n(1-1/k)}$ isolated solutions.

Proof Sketch:

- For every isolated solution, the average (over permutations) compressed length is at most $n - n/k$.
- There exists a permutation such that the average (over all isolated solutions) compressed length is at most $n - n/k$. 
Maximum Number of Isolated Solutions

Lemma

A $k$-CNF can have at most $2^{n(1-1/k)}$ isolated solutions.

Proof Sketch:

- For every isolated solution, the average (over permutations) compressed length is at most $n - n/k$.
- There exists a permutation such that the average (over all isolated solutions) compressed length is at most $n - n/k$.
- Hence, the number of isolated solutions is at most $2^{n(1-1/k)}$ using a convexity argument.
Maximum Number of Isolated Solutions

Lemma

A $k$-CNF can have at most $2^{n(1-1/k)}$ isolated solutions.

Proof Sketch:

- For every isolated solution, the average (over permutations) compressed length is at most $n - n/k$
- There exists a permutation such that the average (over all isolated solutions) compressed length is at most $n - n/k$.
- Hence, the number of isolated solutions is at most $2^{n(1-1/k)}$ using a convexity argument.

Fact

If $\Phi : S \rightarrow \{0, 1\}^*$ is a prefix-free encoding (one-to-one function) with average code length $l$, the $|S| \leq 2^l$. 


Lower Bounds for Parity

Theorem

Computing the parity function requires $2^{n/k}$ size $\Sigma\Pi\Sigma_k$ circuits.
Lower Bounds for Parity

Theorem

Computing the parity function requires $2^{n/k}$ size $\Sigma \Pi \Sigma_k$ circuits.

Theorem

Computing the parity function requires $\Omega(n^{1/4}2^{\sqrt{n}})$ size depth-3 circuits.
Problem: clause lengths are not uniform.
Problem: clause lengths are not uniform.

Let $N_l(x)$ be the number of critical clauses of length $l$ at the solution $x$. 

$$\sum_{l} N_l(x) = n$$ for an isolated solution $x$.

Define weight of $x$, $w(x) = \frac{1}{|C(x,i)|} = \frac{1}{l} \sum N_l(x)$. 

Argue that for a $k$-cnf $F$, the number of isolated solutions with weight greater or equal to $\mu$ is at most $2^{n-\mu}$.
Parity Lower Bound for General Depth-3 Circuits

- Problem: clause lengths are not uniform.
- Let $N_i(x)$ be the number of critical clauses of length $i$ at the solution $x$.
- $\sum_i N_i(x) = n$ for an isolated solution $x$. 

\[ \text{Define weight of } x, \quad w(x) = \sum_{i=1}^{n} \frac{1}{|C(x, i)|} = \frac{\sum_i N_i(x)}{i}. \]
Parity Lower Bound for General Depth-3 Circuits

- Problem: clause lengths are not uniform.
- Let $N_l(x)$ be the number of critical clauses of length $l$ at the solution $x$.
- $\sum_l N_l(x) = n$ for an isolated solution $x$.
- Define weight of $x$, $w(x) = \sum_{i=1}^n 1/|C(x,i)| = \sum_l N_l(x)/l$. 

Properties of $k$-CNF
Problem: clause lengths are not uniform.

Let $N_l(x)$ be the number of critical clauses of length $l$ at the solution $x$.

$\sum_l N_l(x) = n$ for an isolated solution $x$.

Define weight of $x$, $w(x) = \sum_{i=1}^n 1/|C(x,i)| = \sum_l N_l(x)/l$.

Argue that for a $k$-CNF $F$, the number of isolated solutions with weight greater or equal to $\mu$ is at most $2^{n-\mu}$. 
Consider a depth-3 circuit computing parity. Let $S$ be the set of odd inputs accepted by the circuit. $|S| = 2^{n-1}$. 
Consider a depth-3 circuit computing parity. Let $S$ be the set of odd inputs accepted by the circuit. $|S| = 2^{n-1}$.

For each $x \in S$, there exists a CNF $F_x$ accepting $x$ and $x$ is an isolated solution of $F_x$. 

Many clauses (level-1 OR gates) are needed to accept low-weighted isolated solutions. A clause of length $l$ can only be critical for at most $l^2 n - l$ solution-variable pairs $(x, i)$. 

Paturi Properties of $k$-CNF
Consider a depth-3 circuit computing parity. Let $S$ be the set of odd inputs accepted by the circuit. $|S| = 2^{n-1}$.

For each $x \in S$, there exists a CNF $F_x$ accepting $x$ and $x$ is an isolated solution of $F_x$.

Define the weight of $x$ with respect to $F_x$.
Lower Bound Proof

- Consider a depth-3 circuit computing parity. Let $S$ be the set of odd inputs accepted by the circuit. $|S| = 2^{n-1}$.
- For each $x \in S$, there exists a CNF $F_x$ accepting $x$ and $x$ is an isolated solution of $F_x$.
- Define the weight of $x$ with respect to $F_x$.
- Let $\mu = \sqrt{n} + \frac{\log n}{4}$
Consider a depth-3 circuit computing parity. Let $S$ be the set of odd inputs accepted by the circuit. $|S| = 2^{n-1}$.

For each $x \in S$, there exists a CNF $F_x$ accepting $x$ and $x$ is an isolated solution of $F_x$.

Define the weight of $x$ with respect to $F_x$.

Let $\mu = \sqrt{n} + \frac{\log n}{4}$

$S_1 \subseteq S$ be the set of $x$ with $w(x) \geq \mu$. $S_2 = S - S_1$ be the set of $x$ with $w(x) < \mu$,
Consider a depth-3 circuit computing parity. Let $S$ be the set of odd inputs accepted by the circuit. $|S| = 2^{n-1}$.

For each $x \in S$, there exists a CNF $F_x$ accepting $x$ and $x$ is an isolated solution of $F_x$.

Define the weight of $x$ with respect to $F_x$.

Let $\mu = \sqrt{n} + \frac{\log n}{4}$

$S_1 \subseteq S$ be the set of $x$ with $w(x) \geq \mu$. $S_2 = S - S_1$ be the set of $x$ with $w(x) < \mu$,

Number of CNFs (level-2 AND gates) is at least $|S_1|2^{\mu-n}$. 
Consider a depth-3 circuit computing parity. Let $S$ be the set of odd inputs accepted by the circuit. $|S| = 2^{n-1}$.

For each $x \in S$, there exists a CNF $F_x$ accepting $x$ and $x$ is an isolated solution of $F_x$.

Define the weight of $x$ with respect to $F_x$. Let $\mu = \sqrt{n} + \frac{\log n}{4}$.

$S_1 \subseteq S$ be the set of $x$ with $w(x) \geq \mu$. $S_2 = S - S_1$ be the set of $x$ with $w(x) < \mu$.

Number of CNFs (level-2 AND gates) is at least $|S_1|2^{\mu-n}$.

Many clauses (level-1 OR gates) are needed to accept low-weighted isolated solutions.
Lower Bound Proof

- Consider a depth-3 circuit computing parity. Let $S$ be the set of odd inputs accepted by the circuit. $|S| = 2^{n-1}$.
- For each $x \in S$, there exists a CNF $F_x$ accepting $x$ and $x$ is an isolated solution of $F_x$.
- Define the weight of $x$ with respect to $F_x$.
- Let $\mu = \sqrt{n} + \frac{\log n}{4}$
- $S_1 \subseteq S$ be the set of $x$ with $w(x) \geq \mu$. $S_2 = S - S_1$ be the set of $x$ with $w(x) < \mu$.
- Number of CNFs (level-2 AND gates) is at least $|S_1|2^{\mu-n}$.
- Many clauses (level-1 OR gates) are needed to accept low-weighted isolated solutions.
- A clause of length $l$ can only be critical for at most $l2^{n-l}$ solution-variable pairs $(x, i)$. 

Paturi Properties of $k$-CNF
Hence, the number of clauses in all CNFs together must be at least
Hence, the number of clauses in all CNFs together must be at least

$$\sum_{l=1}^{n} \sum_{x \in S_2} N_l(x)/(l2^{n-l}) = \sum_{x \in S_2} n2^{-n} \sum_{i=1}^{n} \frac{N_l(x) 2^l}{n/l} \geq \sum_{x \in S_2} \mu 2^{-n+n/\mu} = |S_2| \mu 2^{-n+n/\mu}$$
Hence, the number of clauses in all CNFs together must be at least

\[
\sum_{l=1}^{n} \sum_{x \in S_2} \frac{N_l(x)}{(l2^{n-l})} = \sum_{x \in S_2} n2^{-n} \sum_{i=1}^{n} \frac{N_l(x) 2^l}{n} \frac{1}{l} \geq \sum_{x \in S_2} \mu 2^{-n+n/\mu} = |S_2|\mu 2^{-n+n/\mu}
\]

Total number of gates is at least \(|S_1|2^{\mu-n} + |S_2|\mu 2^{-n+n/\mu} \).
Hence, the number of clauses in all CNFs together must be at least

\[
\sum_{l=1}^{n} \sum_{x \in S_2} \frac{N_l(x)}{(l2^{n-l})} = \sum_{x \in S_2} n2^{-n} \sum_{i=1}^{n} \frac{N_l(x) 2^l}{n l} \geq \sum_{x \in S_2} \mu 2^{-n+n/\mu} = |S_2| \mu 2^{-n+n/\mu}
\]

Total number of gates is at least \(|S_1|2^{\mu-n} + |S_2|\mu 2^{-n+n/\mu}.

Minimizing the count subject to the condition \(|S_1| + |S_2| = 2^{n-1}\) will yield the desired bound.
Algorithm PPZ:

1. Let $F$ be a $k$-CNF and $\sigma$ a random permutation on variables
2. for $i = 1, \ldots, n$
3. if there is a unit clause for the variable $\sigma(i)$
4. then set the variable $\sigma(i)$ so that the clause true
5. else set the variable $\sigma(i)$ randomly
6. Simplify $F$
7. if $F$ is satisfied, output the assignment
### Analysis

**Lemma**

Algorithm PPZ outputs $x$ with probability at least $\frac{1}{n}2^{-n+I(x)/k}$ for any satisfying solution $x$ with $I(x)$ many neighbors which are not solutions.
Lemma

Algorithm **PPZ** outputs $x$ with probability at least $\frac{1}{n} 2^{-n+I(x)/k}$ for any satisfying solution $x$ with $I(x)$ many neighbors which are not solutions.

Proof Sketch:

- $E_1$ — for at least $I(x)/k$ variables, the critical variable appears as the last variable among the variables in the critical clause.
Algorithm \textbf{PPZ} outputs $x$ with probability at least $\frac{1}{n} 2^{-n + I(x)/k}$ for any satisfying solution $x$ with $I(x)$ many neighbors which are not solutions.

Proof Sketch:

- $E_1$ — for at least $I(x)/k$ variables, the critical variable appears as the last variable among the variables in the critical clause
- $E_2$ — values assigned to the variables in the \textbf{for} loop agree with $x$
Algorithm PPZ outputs $x$ with probability at least $\frac{1}{n} 2^{-n + I(x)/k}$ for any satisfying solution $x$ with $I(x)$ many neighbors which are not solutions.

Proof Sketch:

- $E_1$ — for at least $I(x)/k$ variables, the critical variable appears as the last variable among the variables in the critical clause
- $E_2$ — values assigned to the variables in the for loop agree with $x$
- $P(E_1) \geq 1/n$
Lemma

Algorithm **PPZ** outputs $x$ with probability at least $\frac{1}{n}2^{-n+I(x)/k}$ for any satisfying solution $x$ with $I(x)$ many neighbors which are not solutions.

Proof Sketch:

- $E_1$ — for at least $I(x)/k$ variables, the critical variable appears as the last variable among the variables in the critical clause
- $E_2$ — values assigned to the variables in the for loop agree with $x$
- $\Pr(E_1) \geq 1/n$
- $\Pr(E_2|E_1) \geq 2^{-n+I(x)/k}$
Algorithm PPZ outputs x with probability at least \(\frac{1}{n}2^{-n+I(x)/k}\) for any satisfying solution x with I(x) many neighbors which are not solutions.

Proof Sketch:

- **E₁** — for at least \(I(x)/k\) variables, the critical variable appears as the last variable among the variables in the critical clause
- **E₂** — values assigned to the variables in the for loop agree with x
- \(P(E₁) \geq 1/n\)
- \(P(E₂|E₁) \geq 2^{-n+I(x)/k}\)
- \(P(x \text{ is output by PPZ}) \geq \frac{1}{n}2^{-n+I(x)/k}\)
Let \( S \) be the set of satisfying solutions of \( F \).
Let $S$ be the set of satisfying solutions of $F$.

For $x \in S$, define $value(x) = 2^{-n+I(x)}$.
Let \( S \) be the set of satisfying solutions of \( F \).

For \( x \in S \), define \( \text{value}(x) = 2^{-n + l(x)} \).

Fact: \( \sum_{x \in S} \text{value}(x) \geq 1 \).
Let $S$ be the set of satisfying solutions of $F$.

For $x \in S$, define $\text{value}(x) = 2^{-n+I(x)}$

Fact: $\sum_{x \in S} \text{value}(x) \geq 1$

$$P(x \text{ is output by PPZ}) \geq \sum_{x \in S} \frac{1}{n} 2^{-n+I(x)/k}$$

$$= \frac{1}{n} 2^{-n+n/k} \sum_{x \in S} 2^{(-n+I(x))/k}$$

$$\geq \frac{1}{n} 2^{-n+n/k}$$
Theorem

If $S \neq \emptyset$ is the set of satisfying solutions of a $k$-CNF $F$, then $\text{PPZ}$ finds a satisfying assignment with probability at least $\frac{1}{n} \left( \frac{2^n}{|S|} \right)^{1 - 1/k}$.
Dense Case

Theorem

If $S \neq \emptyset$ is the set of satisfying solutions of a $k$-CNF $F$, then PPZ finds a satisfying assignment with probability at least $\frac{1}{n} \left( \frac{2^n}{|S|} \right)^{1-1/k}$

Proof Sketch: Use the edge isoperimetric inequality for the hypercube to conclude that among all sets $S \subseteq \{0,1\}^n$ of a given size, the subcube of dimension $\log |S|$ minimizes the number of edges between $S$ and $\bar{S}$. 
Further Improvements

- PPZ analysis shows that on average we can expect to find $n/k$ unit clauses for an isolated solution $z$. Can we improve the expected number of unit clauses?
Further Improvements

- PPZ analysis shows that on average we can expect to find $n/k$ unit clauses for an isolated solution $z$. Can we improve the expected number of unit clauses?
- PPZ argument only uses the fact that there is at least one critical clause for each variable at $z$. If there is more than one critical clause per variable we could get a better bound. Let $(x_1 \lor \overline{x}_2 \lor \overline{x}_3)$ and $(x_1 \lor \overline{x}_4 \lor \overline{x}_5)$ be critical clauses for $x_1$ at $z = 1$. The probability that $x_1$ is the last variable among the variables in one of its critical clauses is now at least $7/15$ rather than $1/3$. In general, even if $z$ is the only solution, there need not be more than one critical clause per variable.
Further Improvements

- PPZ analysis shows that on average we can expect to find $n/k$ unit clauses for an isolated solution $z$. Can we improve the expected number of unit clauses?
- PPZ argument only uses the fact that there is at least one critical clause for each variable at $z$.
- If there is more than one critical clause per variable we could get a better bound. Let $(x_1 \lor \overline{x_2} \lor x_3)$ and $(x_1 \lor \overline{x_4} \lor x_5)$ be critical clauses for $x_1$ at $z = 1^n$. 

Paturi Properties of $k$-CNF
Further Improvements

- PPZ analysis shows that on average we can expect to find $n/k$ unit clauses for an isolated solution $z$. Can we improve the expected number of unit clauses?
- PPZ argument only uses the fact that there is at least one critical clause for each variable at $z$.
- If there is more than one critical clause per variable we could get a better bound. Let $(x_1 \lor \bar{x}_2 \lor x_3)$ and $(x_1 \lor \bar{x}_4 \lor x_5)$ be critical clauses for $x_1$ at $z = 1^n$.
- The probability that $x_1$ is the last variable among the variables in one of its critical clauses is now at least $7/15$ rather than $1/3$. 
Further Improvements

• PPZ analysis shows that on average we can expect to find $n/k$ unit clauses for an isolated solution $z$. Can we improve the expected number of unit clauses?

• PPZ argument only uses the fact that there is at least one critical clause for each variable at $z$.

• If there is more than one critical clause per variable we could get a better bound. Let $(x_1 \lor \overline{x}_2 \lor \overline{x}_3)$ and $(x_1 \lor \overline{x}_4 \lor \overline{x}_5)$ be critical clauses for $x_1$ at $z = 1^n$.

• The probability that $x_1$ is the last variable among the variables in one of its critical clauses is now at least $7/15$ rather than $1/3$.

• In general, even if $z$ is the only solution, there need not be more than one critical clause per variable.
Further Improvements — Resolution

Let $F$ contain the clauses $C_1 = (x_1 \lor \bar{x}_2 \lor \bar{x}_3)$, critical for $x_1$, and $C_2 = (x_2 \lor \bar{x}_4 \lor \bar{x}_5)$, critical for $x_2$. 
Further Improvements — Resolution

- Let $F$ contain the clauses $C_1 = (x_1 \lor \overline{x_2} \lor \overline{x_3})$, critical for $x_1$, and $C_2 = (x_2 \lor \overline{x_4} \lor \overline{x_5})$, critical for $x_2$.

- By resolution, we can derive another critical clause $(x_1 \lor \overline{x_3} \lor \overline{x_4} \lor \overline{x_5})$ for $x_1$. With two critical clauses for $x_1$, we can improve the probability of the occurrence of a unit clause for $x_1$. 

Critical clauses alone will not suffice: instead of $C_2$, if we have $C_3 = (x_2 \lor \overline{x_1} \lor \overline{x_4})$ as a critical clause for $x_2$, resolution will not help.

In fact, we cannot have any critical clause for $x_1$ at all without $\overline{x_2}$ in it if $001^n - 2$ is also a satisfying solution.
Further Improvements — Resolution

- Let $F$ contain the clauses $C_1 = (x_1 \lor \bar{x}_2 \lor \bar{x}_3)$, critical for $x_1$, and $C_2 = (x_2 \lor \bar{x}_4 \lor \bar{x}_5)$, critical for $x_2$.

- By resolution, we can derive another critical clause $(x_1 \lor \bar{x}_3 \lor \bar{x}_4 \lor \bar{x}_5)$ for $x_1$. With two critical clauses for $x_1$, we can improve the probability of the occurrence of a unit clause for $x_1$.

- Critical clauses alone will not suffice: instead of $C_2$, if we have $C_3 = (x_2 \lor \bar{x}_1 \lor \bar{x}_4)$ as a critical clause for $x_2$, resolution will not help.
Further Improvements — Resolution

- Let $F$ contain the clauses $C_1 = (x_1 \lor \bar{x}_2 \lor \bar{x}_3)$, critical for $x_1$, and $C_2 = (x_2 \lor \bar{x}_4 \lor \bar{x}_5)$, critical for $x_2$.

- By resolution, we can derive another critical clause $(x_1 \lor \bar{x}_3 \lor \bar{x}_4 \lor \bar{x}_5)$ for $x_1$. With two critical clauses for $x_1$, we can improve the probability of the occurrence of a unit clause for $x_1$.

- Critical clauses alone will not suffice: instead of $C_2$, if we have $C_3 = (x_2 \lor \bar{x}_1 \lor \bar{x}_4)$ as a critical clause for $x_2$, resolution will not help.

- In fact, we cannot have any critical clause for $x_1$ at $z$ without $\bar{x}_2$ in it if $001^{n-2}$ is also a satisfying solution.
We assume that $z$ is $d$-isolated: no other satisfying solution within Hamming distance $d$. We take $d = \omega_n(1)$. 
Further Improvements — $d$-isolation

- We assume that $z$ is $d$-isolated: no other satisfying solution within Hamming distance $d$. We take $d = \omega_n(1)$.
- If $001^{n-2}$ is not a satisfying solution, there must be another critical clause for $x_1$ at $z$. 
Further Improvements — $d$-isolation

- We assume that $z$ is $d$-isolated: no other satisfying solution within Hamming distance $d$. We take $d = \omega_n(1)$.
- If $001^{n-2}$ is not a satisfying solution, there must be another critical clause for $x_1$ at $z$.
- There must be an unsatisfied clause at $001^{n-2}$ involving the literals $x_1$ or $x_2$. Let $C_4 = (x_1 \lor x_2 \lor \overline{x_4})$ be such a clause. Resolving $C_1$ and $C_4$, we get the critical clause $(x_1 \lor \overline{x_3} \lor \overline{x_4})$ for $x_1$ at $z$. 
Further Improvements — \( d \)-isolation

- We assume that \( z \) is \( d \)-isolated: no other satisfying solution within Hamming distance \( d \). We take \( d = \omega_n(1) \).
- If \( 001^{n-2} \) is not a satisfying solution, there must be another critical clause for \( x_1 \) at \( z \).
- There must be an unsatisfied clause at \( 001^{n-2} \) involving the literals \( x_1 \) or \( x_2 \). Let \( C_4 = (x_1 \lor x_2 \lor \bar{x}_4) \) be such a clause. Resolving \( C_1 \) and \( C_4 \), we get the critical clause \((x_1 \lor \bar{x}_3 \lor \bar{x}_4)\) for \( x_1 \) at \( z \).
- We also get another critical clause for \( x_1 \) by considering the nonsatisfying assignment \( 010 n^{n-3} \).
A resolvable pair of clauses $C_1$ and $C_2$ is $s$-bounded, if $|C_1|$, $|C_2| \leq s$ and $|\text{resolvent}(C_1, C_2)| \leq s$. 
A resolvable pair of clauses $C_1$ and $C_2$ is $s$-bounded, if $|C_1|$, $|C_2| \leq s$ and $|\text{resolvent}(C_1, C_2)| \leq s$.

$F_s$ denote the closure of the $k$-CNF under $s$-bounded resolution.
PPSZ Algorithm

- A resolvable pair of clauses \( C_1 \) and \( C_2 \) is \( s \)-bounded, if \(|C_1|, |C_2| \leq s\) and \(|\text{resolvent}(C_1, C_2)| \leq s\).
- \( F_s \) denote the closure of the \( k\text{-CNF} \) under \( s \)-bounded resolution.
- Improved \( k\text{-SAT} \) algorithm: Apply PPZ algorithm to \( F_s \).
For a $d$-isolated solution, we need to estimate the expected number of variables that appear last among the variables in one of its critical clauses according to a random permutation.
PPSZ Analysis

- For a $d$-isolated solution, we need to estimate the expected number of variables that appear last among the variables in one of its critical clauses according to a random permutation.
- Construct a critical clause tree for this calculation.
PPSZ Analysis

- For a $d$-isolated solution, we need to estimate the expected number of variables that appear last among the variables in one of its critical clauses according to a random permutation.
- Construct a critical clause tree for this calculation.
- Cuts of the critical clause tree correspond to critical clauses.
For a $d$-isolated solution, we need to estimate the expected number of variables that appear last among the variables in one of its critical clauses according to a random permutation.

- Construct a critical clause tree for this calculation.
- Cuts of the critical clause tree correspond to critical clauses.
- Calculate the probability that a variable occurs after a cut in its critical clause tree using a recurrence relation.
Lemma

Let $z$ be a $d$-isolated solution of a $k$-CNF and $s \geq k^d$. 

$$P(\text{PPSZ outputs } z) \geq 2^{-\left(1 - \frac{\mu_k}{k-1} + \epsilon(d,k)\right)n}.$$
PPSZ Results

Lemma

Let $z$ be a $d$-isolated solution of a $k$-CNF and $s \geq k^d$.  
$$\Pr(\text{PPSZ outputs } z) \geq 2^{-\left(1 - \frac{\mu_k}{k-1} + \epsilon(d,k)\right)n}.$$  

Notes:

1. $\epsilon$ goes to 0 as $d$ goes to infinity.
PPSZ Results

Lemma

Let $z$ be a $d$-isolated solution of a $k$-CNF and $s \geq k^d$. Then

$$\Pr(\text{PPSZ outputs } z) \geq 2\left(1 - \frac{\mu_k}{k-1} + \epsilon(d,k)\right)n.$$

Notes:

1. $\epsilon$ goes to 0 as $d$ goes to infinity.
2. $\mu_k$ increases with $k$ and $\mu_\infty = \frac{\pi^2}{6}$. 

The number of $d$-isolated solutions of a $k$-CNF is at most

$$2\left(1 - \frac{\mu_k}{k-1} + \epsilon(d,k)\right)n.$$
PPSZ Results

Lemma

Let $z$ be a $d$-isolated solution of a $k$-CNF and $s \geq k^d$. 

\[ P( \text{PPSZ outputs } z) \geq 2^{-(1 - \frac{\mu_k}{k-1} + \epsilon(d,k))n}. \]

Notes:

1. $\epsilon$ goes to 0 as $d$ goes to infinity.

2. $\mu_k = \sum_{j=1}^{\infty} \frac{1}{j(j + 1/k)}$

3. $\mu_k$ increases with $k$ and $\mu_\infty = \frac{\pi^2}{6} = 1.644 \cdots$
PPSZ Results

Lemma

Let $z$ be a $d$-isolated solution of a $k$-CNF and $s \geq k^d$. 
$P(PPSZ \text{ outputs } z) \geq 2^{-\left(1-\frac{\mu_k}{k-1}+\epsilon(d,k)\right)n}$.

Notes:

1. $\epsilon$ goes to 0 as $d$ goes to infinity.
2. 

$$\mu_k = \sum_{j=1}^{\infty} \frac{1}{j(j+1/k)}$$

3. $\mu_k$ increases with $k$ and $\mu_\infty = \frac{\pi^2}{6} = 1.644 \cdots$

4. The number of $d$-isolated solutions of a $k$-CNF is at most $2^{\left(1-\frac{\mu_k}{k-1}+\epsilon(d,k)\right)n}$.
Improved Lower Bounds for Depth-3 Circuits

**Theorem**

Let $E$ be an error-correcting code of minimum distance $d > \log n$ and at least $2^{n-n/\log n}$ code words. If $C$ is a $\Sigma \Pi \Sigma_k$ circuit computing the characteristic function of $E$, then $C$ has at least $2^{(\frac{\mu_k}{k-1} - o(1))n}$ gates.
Improved Lower Bounds for Depth-3 Circuits

**Theorem**

Let $E$ be an error-correcting code of minimum distance $d > \log n$ and at least $2^{n - n/\log n}$ code words. If $C$ is a $\Sigma \Pi \Sigma_k$ circuit computing the characteristic function of $E$, then $C$ has at least $2^{2^{(\frac{\mu_k}{k-1}-o(1))n}}$ gates.

**Theorem**

Let $E$ be an error-correcting code of minimum distance $d > \log n$ and at least $2^{n - \sqrt{n}/\log n}$ code words. If $C$ is a $\Sigma \Pi \Sigma$ circuit computing the characteristic function of $E$, then $C$ has at least $2^{1.282\sqrt{n}}$ gates.
If the $k$-CNF $F$ has a $d$-isolated solution for $d = \omega_n(1)$, then it can be found in time $2^{n \left( 1 - \frac{\mu_k}{k-1} - o(1) \right)}$ with constant success probability.
If the $k$-CNF $F$ has a $d$-isolated solution for $d = \omega_n(1)$, then it can be found in time $2^{n(1-\frac{\mu_k}{k-1}-o(1))}$ with constant success probability.

For the general case, PPSZ obtains the same bound for $k \geq 5$ and slightly weaker bounds for $k = 3$ and $k = 4$. The proof is involved.
If the $k$-CNF $F$ has a $d$-isolated solution for $d = \omega_n(1)$, then it can be found in time $2^{n(1 - \frac{\mu_k}{k-1} - o(1))}$ with constant success probability.

For the general case, PPSZ obtains the same bound for $k \geq 5$ and slightly weaker bounds for $k = 3$ and $k = 4$. The proof is involved.

Recently, T. Hertli presented a simpler and nicer proof to extend the PPSZ bound from the $d$-isolated case to the general case for all $k$. 
Let $C$ be a $\Sigma\Pi\Sigma_k$ circuit of size $s$ computing a balanced function $f$. Think of as $s = 2^{n-o(n)}$. 

Goal: to show that a 'low complexity' function $f$ requires large $s$. 

Let $F$ be a depth-2 subcircuit ($k$-cnf) such that $|F - 1(1)| = \Omega(2^{n/s}) = \Omega(2^{o(n)})$. 

Let $d$ be the VC-dimension of $F - 1$.
Let $C$ be a $\Sigma\Pi\Sigma_k$ circuit of size $s$ computing a balanced function $f$. Think of as $s = 2^{n-o(n)}$.

Goal: to show that a ‘low complexity’ function $f$ requires large $s$. 

Let $F$ be a depth-2 subcircuit ($k$-cnf) such that $|1^F - 1| = \Omega(2^{n/s}) = \Omega(2^{o(n)})$. Let $d$ be the VC-dimension of $F - 1$. 

Paturi

Properties of $k$-CNF
Let $C$ be a $\Sigma\Pi\Sigma_k$ circuit of size $s$ computing a balanced function $f$. Think of as $s = 2^{n-o(n)}$.

Goal: to show that a ‘low complexity’ function $f$ requires large $s$.

Let $F$ be a depth-2 subcircuit ($k$-$\text{CNF}$) such that $\left| F^{-1}(1) \right| = \Omega\left( 2^n/s \right) = \Omega(2^{o(n)})$. 
Let $C$ be a $\Sigma\Pi\Sigma_k$ circuit of size $s$ computing a balanced function $f$. Think of as $s = 2^{n-o(n)}$.

Goal: to show that a ‘low complexity’ function $f$ requires large $s$.

Let $F$ be a depth-2 subcircuit ($k$-CNF) such that $|F^{-1}(1)| = \Omega(2^n/s) = \Omega(2^{o(n)})$.

Let $d$ be the VC-dimension of $F^{-1}(1)$.
How to Prove Stronger Lower Bounds for Depth-3 Circuits

- \( d \geq \log(2^n/s)/\log n \). Without loss of generality, assume that the set \( \{1, 2, \cdots, d\} \) is shattered when you view the elements of \( F^{-1}(1) \) as subsets of \( \{1, 2, \cdots, n\} \).
How to Prove Stronger Lower Bounds for Depth-3 Circuits

- \( d \geq \log(2^n/s)/\log n \). Without loss of generality, assume that the set \( \{1, 2, \cdots, d\} \) is shattered when you view the elements of \( F^{-1}(1) \) as subsets of \( \{1, 2, \cdots, n\} \).

- Select \( 2^d \) inputs from \( F^{-1}(1) \) of the form \( yp_1(y)p_2(y)\cdots p_{(n-d)}(y) \) for each \( y \in \{0, 1\}^d \) for some degree \( d \) \( GF(2) \) polynomials \( p_i \) in \( d \) variables. Call this set \( D_F \).
How to Prove Stronger Lower Bounds for Depth-3 Circuits

- $d \geq \log(2^n/s)/\log n$. Without loss of generality, assume that the set \{1, 2, \cdots, d\} is shattered when you view the elements of $F^{-1}(1)$ as subsets of \{1, 2, \cdots, n\}.

- Select $2^d$ inputs from $F^{-1}(1)$ of the form $y p_1(y) p_2(y) \cdots p_{(n-d)}(y)$ for each $y \in \{0, 1\}^d$ for some degree $d$ GF(2) polynomials $p_i$ in $d$ variables. Call this set $D_F$.

- $F$ is constant on $D_F$. We argue that a random degree-2 GF(2) polynomial is constant on $D$ with probability at most $2^{-\Omega(d^2)}$. 
We then want to argue that there is at least one degree 2 polynomial that is not constant on every $D_F$. 
We then want to argue that there is at least one degree 2 polynomial that is not constant on every $D_F$.

The problem is that there are too many such sets $D_F$ (about $2^{O(n^k)}$).
Lemma (Sparsification Lemma, IPZ 1997)

∃ algorithm $A$ $\forall k \geq 2, \epsilon \in (0, 1], \phi \in k$-$\text{CNF}$ with $n$ variables, $A_{k,\epsilon}(\phi)$ outputs $\phi_1, \ldots, \phi_s \in k$-$\text{CNF}$ in $2^{\epsilon n}$ time such that

1. $s \leq 2^{\epsilon n}$; $\text{Sol}(\phi) = \bigcup_i \text{Sol}(\phi_i)$, where $\text{Sol}(\phi)$ is the set of satisfying assignments of $\phi$

2. $\forall i \in [s]$ each literal occurs $\leq O\left(\frac{k}{\epsilon}\right)^{3k}$ times in $\phi_i$. 
Theorem

Almost all degree 2 GF(2) polynomials require $\Omega(2^{n-o(n)})$ size $\Sigma\Pi\Sigma_k$ circuits for $k = o(\log n)$.

Proof Sketch:

1. Sparsify each of level-2 subcircuits to get an equivalent circuit which is an OR of linear size $k$-CNF’s. The size only goes up by a factor $2^{o(n)}$. 

Stronger Lower Bounds for Depth-3 Circuits

**Theorem**

*Almost all degree 2 GF(2) polynomials require \( \Omega(2^{n-o(n)}) \) size \( \Sigma \Pi \Sigma_k \) circuits for \( k = o(\log n) \).*

**Proof Sketch:**

1. Sparsify each of level-2 subcircuits to get an equivalent circuit which is an OR of linear size \( k\text{-CNF}'s \). The size only goes up by a factor \( 2^{o(n)} \).
2. There are only \( \binom{O(n^k)}{O(n)} \leq n^{O(n)} \) many linear size \( k\text{-CNFs} \).
Stronger Lower Bounds for Depth-3 Circuits

**Theorem**

Almost all degree 2 GF(2) polynomials require \( \Omega(2^{n-o(n)}) \) size \( \Sigma \Pi \Sigma_k \) circuits for \( k = o(\log n) \).

**Proof Sketch:**

1. Sparsify each of level-2 subcircuits to get an equivalent circuit which is an OR of linear size \( k \)-CNF’s. The size only goes up by a factor \( 2^{o(n)} \).
2. There are only \( (O(n^k)) \leq n^{O(n)} \) many linear size \( k \)-CNFs.
3. We can now complete the previous counting argument.
Switching Lemma

**Lemma (Håstad’s Switching Lemma)**

Let $F$ be a $k$-CNF and $\rho$ be a random restriction with $pn$ unset variables. Then

$$P(\text{Decision tree height of } F \mid \rho > t) \leq (5pk)^t$$
Switching Lemma

Lemma (Håstad’s Switching Lemma)

Let $F$ be a $k$-CNF and $\rho$ be a random restriction with $pn$ unset variables. Then

$$\Pr(\text{Decision tree height of } F | \rho > t) \leq (5pk)^t$$

Notes:

1. A restriction $\rho$ is a mapping from $\{1, 2, \ldots, n\} \rightarrow \{0, 1, *\}$. If $\rho(i) = *$, then we say that variable $i$ is unset.
Switching Lemma

Lemma (Håstad’s Switching Lemma)

Let $F$ be a $k$-CNF and $\rho$ be a random restriction with $pn$ unset variables. Then

$$P(\text{Decision tree height of } F \upharpoonright \rho > t) \leq (5pk)^t$$

Notes:

1. A restriction $\rho$ is a mapping from $\{1, 2, \ldots, n\} \rightarrow \{0, 1, \ast\}$. If $\rho(i) = \ast$, then we say that variable $i$ is unset.
2. $F \upharpoonright \rho$ is the $k$-CNF obtained by restricting $F$ to $\rho$. 


Switching Lemma

Lemma (Håstad’s Switching Lemma)

Let $F$ be a $k$-CNF and $\rho$ be a random restriction with $pn$ unset variables. Then

$$\Pr(\text{Decision tree height of } F \upharpoonright \rho > t) \leq (5pk)^t$$

Notes:

1. A restriction $\rho$ is a mapping from $\{1, 2, \ldots, n\} \rightarrow \{0, 1, \ast\}$. If $\rho(i) = \ast$, then we say that variable $i$ is unset.
2. $F \upharpoonright \rho$ is the $k$-CNF obtained by restricting $F$ to $\rho$.
3. Switching Lemma $\longrightarrow$ strong correlation bounds for approximating parity function by small depth circuits.
Switching Lemma

Lemma (Håstad’s Switching Lemma)

Let $F$ be a $k$-CNF and $\rho$ be a random restriction with $pn$ unset variables. Then

$$\mathbb{P}(\text{Decision tree height of } F \upharpoonright \rho > t) \leq (5pk)^t$$

Notes:

1. A restriction $\rho$ is a mapping from $\{1, 2, \ldots, n\} \rightarrow \{0, 1, *\}$. If $\rho(i) = *$, then we say that variable $i$ is unset.
2. $F \upharpoonright \rho$ is the $k$-CNF obtained by restricting $F$ to $\rho$.
3. Switching Lemma $\rightarrow$ strong correlation bounds for approximating parity function by small depth circuits.
4. Switching Lemma $\rightarrow$ a satisfiability algorithm for small depth circuits.
Switching Lemma

Lemma (Håstad’s Switching Lemma)

Let $F$ be a $k$-CNF and $\rho$ be a random restriction with $pn$ unset variables. Then

$$P(\text{Decision tree height of } F \upharpoonright \rho > t) \leq (5pk)^t$$

Notes:

1. A restriction $\rho$ is a mapping from $\{1, 2, \ldots, n\} \rightarrow \{0, 1, *\}$. If $\rho(i) = *$, then we say that variable $i$ is unset.

2. $F \upharpoonright \rho$ is the $k$-CNF obtained by restricting $F$ to $\rho$.

3. Switching Lemma $\rightarrow$ strong correlation bounds for approximating parity function by small depth circuits.

4. Switching Lemma $\rightarrow$ a satisfiability algorithm for small depth circuits.

5. Requires a nontrivial extension of the Switching Lemma.
Small Depth Circuits and Satisfiability Algorithm

- An \((n, m, d)\)-circuit is a Boolean circuit on \(n\) variables with \(d\) alternating layers of AND/OR gates where each layer has at most \(m = cn\) gates.
Small Depth Circuits and Satisfiability Algorithm

- An \((n, m, d)\)-circuit is a Boolean circuit on \(n\) variables with \(d\) alternating layers of AND/OR gates where each layer has at most \(m = cn\) gates.
- An \((n, m, d, k)\)-circuit is an \((n, m, d)\)-circuit where each gate at level \(d\) (bottom level) has fan-in bounded by \(k\) (instead of limiting the number of gates at level \(d\)).
Small Depth Circuits and Satisfiability Algorithm

- An \((n, m, d)\)-circuit is a Boolean circuit on \(n\) variables with \(d\) alternating layers of AND/OR gates where each layer has at most \(m = cn\) gates.

- An \((n, m, d, k)\)-circuit is an \((n, m, d)\)-circuit where each gate at level \(d\) (bottom level) has fan-in bounded by \(k\) (instead of limiting the number of gates at level \(d\)).

**Theorem (Satisfiability Algorithm for Small Depth Circuits)**

There is a Las Vegas algorithm for deciding the satisfiability of an \((n, cn, d)\)-circuit \(C\) with expected time at most \(\text{poly}(n) |C| 2^{n(1 - \mu_{c,d})}\), where the savings

\[
\mu_{c,d} \geq \frac{1}{(O(\log c + d \log d))^{d-1}}
\]
Correlation

- Let $f$ and $g$ be two Boolean functions on $n$ variables. Let $q = \mathbb{P}_{x \in \{0,1\}^n}(f(x) = g(x))$
Correlation

- Let $f$ and $g$ be two Boolean functions on $n$ variables. Let 
  \[ q = \mathbb{P}_{x \in \{0,1\}^n}(f(x) = g(x)) \]
- The correlation between $f$ and $g$ is defined as 
  \[ \text{Cor}(f, g) = 2q - 1. \]
Correlation

- Let $f$ and $g$ be two Boolean functions on $n$ variables. Let $q = P_{x \in \{0,1\}^n}(f(x) = g(x))$
- The correlation between $f$ and $g$ is defined as $Cor(f, g) = 2q - 1$.
- If $\mathcal{F}$ is a class of Boolean functions, we define the correlation between $f$ and $\mathcal{F}$ as $Cor(f, \mathcal{F}) = \text{maximum correlation between } f \text{ and some function } g \in \mathcal{F}$. 
Let $f$ and $g$ be two Boolean functions on $n$ variables. Let
\[ q = \mathbb{P}_{x \in \{0,1\}^n}(f(x) = g(x)) \]

The correlation between $f$ and $g$ is defined as
\[ \text{Cor}(f, g) = 2q - 1. \]

If $\mathcal{F}$ is a class of Boolean functions, we define the correlation between $f$ and $\mathcal{F}$ as $\text{Cor}(f, \mathcal{F}) = \text{maximum correlation between } f \text{ and some function } g \in \mathcal{F}.$

If $\mathcal{F}$ is closed under complementation, then
\[ 0 \leq \text{Cor}(f, \mathcal{F}) \leq 1. \]
Correlation Bounds for Small Depth Circuits

Theorem

The correlation of parity with any \((n, m, d)\)-circuit is at most

\[ 2^{-\mu_{c,d}n} = 2^{-n/(O(\log c + d \log d))^{d-1}} \]
The correlation of parity with any \((n, m, d)\)-circuit is at most

\[2^{-\mu_{c,d} n} = 2^{-n/(O(\log c+d \log d))^{d-1}}\]

For linear size circuits where \(c\) and \(d\) are constants, the savings \(\mu\) is constant and the correlation bound \(2^{-\Theta(n)}\) is strongly exponential.
Correlation Bounds for Small Depth Circuits

Theorem

The correlation of parity with any \((n, m, d)\)-circuit is at most

\[ 2^{-\mu_{c,d} n} = 2^{-n/(O(\log c + d \log d))^{d-1}} \]

1. For linear size circuits where \(c\) and \(d\) are constants, the savings \(\mu\) is constant and the correlation bound \(2^{-\Theta(n)}\) is strongly exponential.

2. Nontrivial savings and correlation bounds for circuit of size up to \(2^{O(n^{1/(d-1)})}\).
Further Improvements could be Hard

- If the satisfiability of an \((n, m, d)\)-circuit can be decided in time \(2^{n(1-\frac{1}{O(\log m) o(d)})}\), then \(\text{NEXP} \subsetneq \text{NC}^1\).
Further Improvements could be Hard

If the satisfiability of an \((n, m, d)\)-circuit can be decided in time \(2^{n \left(1 - \frac{1}{O(\log m)^{o(d)}}\right)}\), then \(\text{NEXP} \subset \text{NC}^1\).
Partitions

- A set of functions $g_1, \ldots, g_l : \{0, 1\}^n \to \{0, 1\}$ partitions $\{0, 1\}^n$ if $(g_i^{-1}(1))_{1 \leq i \leq l}$ is a partition of $\{0, 1\}^n$. 
Partitions

- A set of functions $g_1, \ldots, g_l : \{0, 1\}^n \rightarrow \{0, 1\}$ partitions $\{0, 1\}^n$ if $(g_i^{-1}(1))_{1 \leq i \leq l}$ is a partition of $\{0, 1\}^n$.
- The $i$'th region of the partition is $g_i^{-1}(1)$. We identify the region with the function $g_i$. 

Paturi

Properties of $k$-CNF
Partitions

- A set of functions $g_1, \ldots, g_l : \{0, 1\}^n \rightarrow \{0, 1\}$ partitions $\{0, 1\}^n$ if $(g_i^{-1}(1))_{1 \leq i \leq l}$ is a partition of $\{0, 1\}^n$.
- The $i$'th region of the partition is $g_i^{-1}(1)$. We identify the region with the function $g_i$.
- $g_i$ are of the form $G \land \rho$, where $G$ is $k$-CNF and $\rho$ a restriction. We denote the region $\mathcal{R}$ by $(G, \rho)$.
Partitions

- A set of functions $g_1, \ldots, g_l : \{0, 1\}^n \rightarrow \{0, 1\}$ partitions $\{0, 1\}^n$ if $(g_i^{-1}(1))_{1 \leq i \leq l}$ is a partition of $\{0, 1\}^n$.

- The $i$'th region of the partition is $g_i^{-1}(1)$. We identify the region with the function $g_i$.

- $g_i$ are of the form $G \land \rho$, where $G$ is $k$-CNF and $\rho$ a restriction. We denote the region $\mathcal{R}$ by $(G, \rho)$.

- Two circuits are equivalent in a region $\mathcal{R}$ if $\mathcal{R} \implies (C \equiv D)$. 

Partitions

- A set of functions $g_1, \ldots, g_l : \{0, 1\}^n \to \{0, 1\}$ partitions $\{0, 1\}^n$ if $(g_i^{-1}(1))_{1 \leq i \leq l}$ is a partition of $\{0, 1\}^n$.
- The $i$'th region of the partition is $g_i^{-1}(1)$. We identify the region with the function $g_i$.
- $g_i$ are of the form $G \land \rho$, where $G$ is $k$-CNF and $\rho$ a restriction. We denote the region $R$ by $(G, \rho)$.
- Two circuits are equivalent in a region $R$ if $R \implies (C \equiv D)$.
- A set $\mathcal{P} = \{(R_i = (G_i, \rho_i), C_i)\}$ is a **partitioning** for a circuit $C$ if $R_i$ partition $\{0, 1\}^n$ and $C_i$ is equivalent to $C$ in region $R_i$ for all $i$. 

Paturi
Properties of $k$-CNF
Canonical Decision Tree

- \textit{height}(T) of a decision tree \( T \) is the length of the longest path.


**Canonical Decision Tree**

- \( \text{height}(T) \) of a decision tree \( T \) is the length of the longest path.
- **Canonical decision tree** \( \text{tree}(F) \) for a \( \text{CNF} \) \( F \) is as follows:
Canonical Decision Tree

- $\text{height}(T)$ of a decision tree $T$ is the length of the longest path.
- **Canonical decision tree** $\text{tree}(F)$ for a $\text{CNF}$ $F$ is as follows:
  1. Fix an ordering of clauses in $F$
**Canonical Decision Tree**

- *height(T)* of a decision tree *T* is the length of the longest path.
- **Canonical decision tree** *tree(F)* for a CNF *F* is as follows:
  1. Fix an ordering of clauses in *F*
  2. If a clause is empty, return 0
Canonical Decision Tree

- $height(T)$ of a decision tree $T$ is the length of the longest path.
- **Canonical decision tree** $tree(F)$ for a CNF $F$ is as follows:
  1. Fix an ordering of clauses in $F$
  2. If a clause is empty, return 0
  3. If there are no clauses, return 1
height(T) of a decision tree T is the length of the longest path.

Canonical decision tree tree(F) for a CNF F is as follows:
1. Fix an ordering of clauses in F
2. If a clause is empty, return 0
3. If there are no clauses, return 1
4. Let C be the first clause. Query the variables in C in order
Canonical Decision Tree

- $\text{height}(T)$ of a decision tree $T$ is the length of the longest path.
- **Canonical decision tree** $\text{tree}(F)$ for a CNF $F$ is as follows:
  1. Fix an ordering of clauses in $F$
  2. If a clause is empty, return 0
  3. If there are no clauses, return 1
  4. Let $C$ be the first clause. Query the variables in $C$ in order
  5. Restrict $F$ based on the query results and recurse.
**Canonical Decision Tree**

- $height(T)$ of a decision tree $T$ is the length of the longest path.
- **Canonical decision tree** $tree(F)$ for a CNF $F$ is as follows:
  1. Fix an ordering of clauses in $F$
  2. If a clause is empty, return 0
  3. If there are no clauses, return 1
  4. Let $C$ be the first clause. Query the variables in $C$ in order
  5. Restrict $F$ based on the query results and recurse.
Canonical decision tree $\text{tree}(\Phi)$ for a sequence of \((F_1, \ldots, F_l)\) of CNF’s is as follows:
Canonical Decision Tree for a Sequence of Formulas

- Canonical decision tree \( \text{tree}(\Phi) \) for a sequence of \((F_1, \ldots, F_l)\) of \(\text{CNF}’\)s is as follows:
  1. First construct the canonical decision tree for \(F_1\).
Canonical Decision Tree for a Sequence of Formulas

- Canonical decision tree $\text{tree} (\Phi)$ for a sequence of $(F_1, \ldots, F_l)$ of CNF’s is as follows:
  1. First construct the canonical decision tree for $F_1$.
  2. Along each path, restrict $F_2, \ldots, F_l$ by the results of queries and recurse.
Canonical Decision Tree for a Sequence of Formulas

- Canonical decision tree $\text{tree}(\Phi)$ for a sequence of $(F_1, \ldots, F_l)$ of CNF’s is as follows:
  1. First construct the canonical decision tree for $F_1$.
  2. Along each path, restrict $F_2, \ldots, F_l$ by the results of queries and recurse.
  3. Label the leaves with the tuples of the leaves from each of the trees.
Canonical Decision Tree for a Sequence of Formulas

- **Canonical decision tree** $\text{tree}(\Phi)$ for a sequence of $(F_1, \ldots, F_l)$ of $\text{CNF}$’s is as follows:
  1. First construct the canonical decision tree for $F_1$.
  2. Along each path, restrict $F_2, \ldots, F_l$ by the results of queries and recurse.
  3. Label the leaves with the tuples of the leaves from each of the trees.

- We say that a clause **contributes** variables to a path if any variable in the clause are queried when the clause gets its turn.
Extended Switching Lemma

Lemma (Extended Switching Lemma)

Let $\Phi = (F_1, \ldots, F_m)$ be a sequence of $k$-CNF’s (or $k$-DNF’s) on $n$ variables. For $p \leq 1/13$, let $\rho$ be a random restriction that leaves $pn$ variables unset. The probability that the decision tree for $\Phi$ has a path of length $> t$ where each $F_i$ contributes at least one node to the path is at most $(13pk)^t$. 
Lemma (Switching Algorithm)

Let $\Phi = (F_1, \ldots, F_m)$ be a sequence of $k$-DNF’s on $n$ variables. There exits a randomized algorithm which takes $\Phi$ as input and outputs a partitioning $\mathcal{P} = \{(R_i, C_i)\}_{1 \leq i \leq s}$ for $\Phi$ such that $C_i$ are $k$-CNF’s in at most $n/100k$ variables, and with high probability the algorithm runs in time at most $\text{poly}(n) \cdot \text{size}(\Phi)^s$. 

Paturi 
Properties of $k$-CNF
Lemma (Switching Algorithm)

Let $\Phi = (F_1, \ldots, F_m)$ be a sequence of $k$-DNF’s on $n$ variables. There exits a randomized algorithm which takes $\Phi$ as input and outputs a partitioning $\mathcal{P} = \{(R_i, C_i)\}_{1 \leq i \leq s}$ for $\Phi$ such that $C_i$ are $k$-CNF’s in at most $n/100k$ variables, and with high probability

$$s \leq \frac{2n}{100k} 2^{n - \frac{n}{100k}} + 3^{-k} m$$
Lemma (Switching Algorithm)

Let $\Phi = (F_1, \ldots, F_m)$ be a sequence of $k$-DNF’s on $n$ variables. There exists a randomized algorithm which takes $\Phi$ as input and outputs a partitioning $\mathcal{P} = \{(R_i, C_i)\}_{1 \leq i \leq s}$ for $\Phi$ such that $C_i$ are $k$-CNF’s in at most $\frac{n}{100k}$ variables, and with high probability

1. $s \leq \frac{2n}{100k} 2^n - \frac{n}{100k} + 3^{-k} m$
2. the algorithm runs in time at most $\text{poly}(n) \text{size}(\Phi)s$. 
Algorithm for Depth-3 Circuits

- Satisfiability Algorithm for \((n, m = cn, 3)\)-circuits (AND-OR-AND) running in time \(2^{n \left(1 - \frac{1}{O(\log c)^2}\right)}\).
Algorithm for Depth-3 Circuits

- Satisfiability Algorithm for \((n, m = cn, 3)\)-circuits (AND-OR-AND) running in time \(2^{n(1 - \frac{1}{O(\log c)^2})}\).
- Reduce the \((n, m, 3)\)-circuit to a small family of \((n, m, 3, k)\)-circuits \(C\) where \(k = O(\log c)\). Overhead is minimal.
Algorithm for Depth-3 Circuits

- Satisfiability Algorithm for \((n, m = cn, 3)\)-circuits (AND-OR-AND) running in time \(2^{n(1 - \frac{1}{O(\log c)^2})}\).
- Reduce the \((n, m, 3)\)-circuit to a small family of \((n, m, 3, k)\)-circuits \(C\) where \(k = O(\log c)\). Overhead is minimal.
- Apply the Switching Algorithm to the family of \(\Phi = (F_1, \ldots, F_m)\) \(k\)-DNF’s to obtain a partitioning into about \(2^n(1 - \frac{1}{100k})\) regions where \(\Phi\) is equivalent to a sequence of \(k\)-CNF’s in at most \(n/100k\) variables and each region is defined by a \(k\)-CNF in the same set of variables.
Algorithm for Depth-3 Circuits

- Satisfiability Algorithm for \((n, m = cn, 3)\)-circuits (AND-OR-AND) running in time \(2^{n \left(1 - \frac{1}{O(\log c)^2}\right)}\).
- Reduce the \((n, m, 3)\)-circuit to a small family of \((n, m, 3, k)\)-circuits \(\mathcal{C}\) where \(k = O(\log c)\). Overhead is minimal.
- Apply the Switching Algorithm to the family of \(\Phi = (F_1, \ldots, F_m)\) \(k\)-DNF’s to obtain a partitioning into about \(2^n(1 - \frac{1}{100k})\) regions where \(\Phi\) is equivalent to a sequence of \(k\)-CNF’s in at most \(n/100k\) variables and each region is defined by a \(k\)-CNF in the same set of variables.
- For each region, collapse the levels to obtain a single \(k\)-CNF and take the conjunction with the defining \(k\)-CNF of the region.
Algorithm for Depth-3 Circuits

- Satisfiability Algorithm for \((n, m = cn, 3)\)-circuits (AND-OR-AND) running in time \(2^{n(1 - \frac{1}{O(\log c)^2})}\).
- Reduce the \((n, m, 3)\)-circuit to a small family of \((n, m, 3, k)\)-circuits \(C\) where \(k = O(\log c)\). Overhead is minimal.
- Apply the Switching Algorithm to the family of \(\Phi = (F_1, \ldots, F_m)\) \(k\)-DNF’s to obtain a partitioning into about \(2^{n(1 - \frac{1}{100k})}\) regions where \(\Phi\) is equivalent to a sequence of \(k\)-CNF’s in at most \(n/100k\) variables and each region is defined by a \(k\)-CNF in the same set of variables.
- For each region, collapse the levels to obtain a single \(k\)-CNF and take the conjunction with the defining \(k\)-CNF of the region.
- Apply a \(k\)-SAT algorithm to each \(k\)-CNF.
Thank You