Algorithms for 3-SAT
An Exposition

N.R.Aravind

Indian Institute of Technology Hyderabad

Outline

0. SAT: History of algorithms
1. Branching
2. Local Search
3. Random Walk
4. Resolutions and randomness

0. SAT: history of algorithms

CNF SAT

Input: $C_1, \ldots, C_m$: Disjunctive clauses over $x_1, \ldots, x_n$

Output: $\{x_1, \ldots, x_n\} \rightarrow \{T, F\}$ that satisfies every clause

CNF SAT

Input: $C_1, \ldots, C_m$: Disjunctive clauses over $x_1, \ldots, x_n$

Output: $\{x_1, \ldots, x_n\} \rightarrow \{T, F\}$ that satisfies every clause

Eg 1: $(x \vee y \vee z)$, $(\neg x \vee \neg y \vee \neg z)$

Eg 2: $(x \vee y)$, $(x \vee \neg y)$, $(\neg x)$

k-SAT: Each clause has at most k literals.

Polynomial-time algorithm when $k = 2$; NP-hard for $k \geq 3$.

Algorithms for 3-SAT

$O(1.61^n)$	1987	Monien, Speckenmeyer
$O(1.38^n)$	1998	PPSZ
$O(1.33^n)$	1999	Schöning
$O(1.308^n)$	2011	PPSZ, Hertli
$O(1.47^n)$	2002,'04	DGHKPRS, Brueggemann, Kern
$O(1.308^n)$	2019	Hansen, Kaplan, Zamir, Zwick

Algorithms for SAT

k-SAT	$O((2-2/k)^n)$	Schöning
SAT	$O^*(2^{n(1-\frac{1}{1+\log m/n})})$	Schuler
SAT $(m = cn)$	$\to O\left((2-\varepsilon)^n\right)$	Arvind,Schuler
SAT	$O^*(2^{n-c\sqrt{n}})$	Pudlak

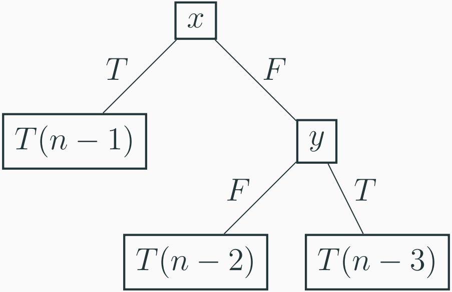
Exponential Time Hypothesis

- For every k, k-SAT needs $\Omega({c_k}^n)$ time, $c_k > 1$
- SAT cannot be solved in time $O(2^{o(n)})$.

1. Branching

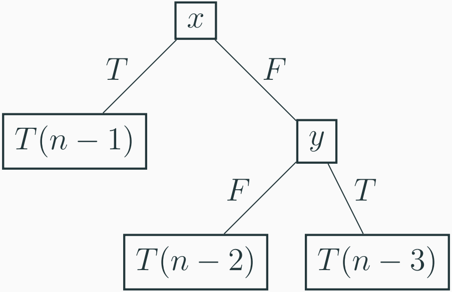
Branching

Consider a clause $(x \vee \neg y \vee z)$



Branching

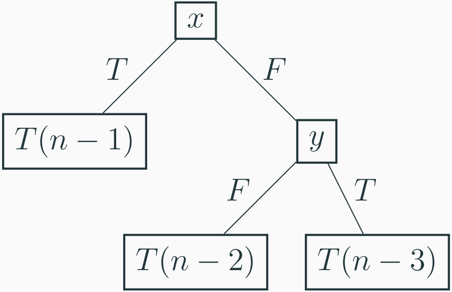
Consider a clause $(x \vee \neg y \vee z)$



$$T(n) = T(n-1) + T(n-2) + T(n-3) \to 1.83^n$$

Branching

Consider a clause $(x \vee \neg y \vee z)$



$T(n) = T(n-1) + T(n-2) + T(n-3) \to 1.83^n$

Improve to: $T(n) = T(n-1) + T(n-2) \to 1.61^n$

2. Local Search

Local Search

1. Cover $\{0,1\}^n$ with Hamming balls $B(a,r) = \{x : H(x,a) = r\}$
2. Search each ball in time

Local Search

1. Cover $\{0,1\}^n$ with Hamming balls $B(a,r)=\{x : H(x,a)=r\}$
2. Search each ball in time $O(3^r)$.

Local Search

1. Cover $\{0,1\}^n$ with Hamming balls $B(a,r) = \{x : H(x,a) = r\}$
2. Search each ball in time $O(3^r)$.
3. While there is a false clause $(l_1 \vee l_2 \vee l_3)$, change one of them.

Local Search

1. Cover $\{0,1\}^n$ with Hamming balls $B(a,r) = \{x : H(x,a) = r\}$
2. Search each ball in time $O(3^r)$.
3. While there is a false clause $(l_1 \vee l_2 \vee l_3)$, change one of them.
4. $O(1.732^n)$ algorithm for 3-SAT

Local Search

- $r \sim \varepsilon n \to Vol(B) \sim 2^{H(\varepsilon)n}$
- Number of balls: $O^*(2^{n(1-H(\varepsilon))})$

Local Search

- $r \sim \varepsilon n \to Vol(B) \sim 2^{H(\varepsilon)n}$
- Number of balls: $O^*(2^{n(1-H(\varepsilon))})$
- Minimize $k^{\varepsilon n} 2^{n(1-H(\varepsilon))}$.
- $O^*(\left(\frac{2k}{k+1}\right)^n)$
- $O(1.5^n)$ for 3-SAT

Local Search for 3-SAT

How efficiently can we search $B(a, r)$?

- $\sim (2.79)^r \to (1.473)^n$

Local Search for 3-SAT

How efficiently can we search $B(a, r)$?

- $\sim (2.79)^r \rightarrow (1.473)^n$
- Need $\sim (1.9)^r$ to beat $(1.308)^n$

3. Random walks

Sch´oning's Algorithm

- Pick a random assignment.

Sch´oning's Algorithm

- Pick a random assignment.
- If C is a false clause, randomly flip a literal of C.

Sch´oning's Algorithm

- Pick a random assignment.
- If C is a false clause, randomly flip a literal of C.
- If no solution in $3n$ steps, restart.

Sch´oning's Algorithm

- Pick a random assignment.
- If C is a false clause, randomly flip a literal of C.
- If no solution in $3n$ steps, restart.
- Pr[Reaching a satisfying assignment] $\geq \left(\frac{3}{4}\right)^n$.

Schöning's Algorithm

Pr[Success]: $\sum_{j=0}^{n} \binom{n}{j} \frac{1}{2^n} q_j$

q_j: Pr[reaching n within $3n$ steps from $n-j$].

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

4. Resolutions and randomness

Unique 3-SAT

- $\varphi = \{(x \vee y), (x \vee \neg y), (\neg y \vee \neg x)\}$

Unique 3-SAT

- $\varphi = \{(x \vee y),\ (x \vee \neg y),\ (\neg y \vee \neg x)\}$
- $(C_1 \vee C_2) \to x$

Unique 3-SAT

- $\varphi = \{(x \vee y), (x \vee \neg y), (\neg y \vee \neg x)\}$
- $(C_1 \vee C_2) \to x$
- $(C_2 \vee C_3) \to \neg y$

Unique 3-SAT

- $\varphi = \{(x \vee y), (x \vee \neg y), (\neg y \vee \neg x)\}$
- $(C_1 \vee C_2) \to x$
- $(C_2 \vee C_3) \to \neg y$
- $x = T, y = F$.

$\varphi \in$ Unique-3-SAT if it has exactly one satisfying assignment: $(a_1, a_2, \ldots, a_n)$.

Unique 3-SAT and Implications

- $\varphi = \{(x \vee y \vee z), (x \vee y \vee \neg z), (\neg x \vee y), (\neg y \vee \neg z), (\neg x \vee z)\}$

Unique 3-SAT and Implications

- $\varphi = \{(x \vee y \vee z), (x \vee y \vee \neg z), (\neg x \vee y), (\neg y \vee \neg z), (\neg x \vee z)\}$
- $(C_1 \vee C_2 \vee C_3) \rightarrow y$

Unique 3-SAT and Implications

- $\varphi = \{(x \vee y \vee z), (x \vee y \vee \neg z), (\neg x \vee y), (\neg y \vee \neg z), (\neg x \vee z)\}$
- $(C_1 \vee C_2 \vee C_3) \rightarrow y$
- $\varphi \xrightarrow{3} y$

Unique 3-SAT and Implications

- $\varphi = \{(x \vee y \vee z), (x \vee y \vee \neg z), (\neg x \vee y), (\neg y \vee \neg z), (\neg x \vee z)\}$
- $(C_1 \vee C_2 \vee C_3) \rightarrow y$
- $\varphi \xrightarrow{3} y$
- $\varphi[y = T] \xrightarrow{1} \neg z$

Unique 3-SAT and Implications

- $\varphi = \{(x \vee y \vee z),\ (x \vee y \vee \neg z),\ (\neg x \vee y),\ (\neg y \vee \neg z),\ (\neg x \vee z)\}$
- $(C_1 \vee C_2 \vee C_3) \rightarrow y$
- $\varphi \xrightarrow{3} y$
- $\varphi[y = T] \xrightarrow{1} \neg z$
- $\varphi[y = T, z = F] \xrightarrow{1} \neg x$

Unique 3-SAT and Implications

- $\varphi = \{(x \vee y \vee z), (x \vee y \vee \neg z), (\neg x \vee y), (\neg y \vee \neg z), (\neg x \vee z)\}$
- $(C_1 \vee C_2 \vee C_3) \to y$
- $\varphi \xrightarrow{3} y$
- $\varphi[y = T] \xrightarrow{1} \neg z$
- $\varphi[y = T, z = F] \xrightarrow{1} \neg x$
- $\varphi[y = T] \xrightarrow{2} \neg x$

t-Implication

$$\varphi \overset{\mathsf{t}}{\Rightarrow} x$$

if

$$\varphi \overset{\mathsf{t}}{\rightarrow} x \text{ or } \varphi \overset{\mathsf{t}}{\rightarrow} \neg x$$

if

$$\varphi \text{ contains } t \text{ clauses that imply } x \text{ or } \neg x$$

t-implication

The t-implication subroutine checks whether there exists **a set of t clauses that force some variable**.
Time: $O(m)^t = n^{O(t)}$.

$PPSZ(\varphi, t)$

1. Random ordering of variables: $x_1, \ldots, x_n$
2. For $i = 1$ to n:
   - If $\varphi[x_1 = b_1, \ldots, x_{i-1} = b_i] \overset{t}{\Rightarrow} x_i = T$, set $x_i = T$.
   - Else If $\varphi[x_1 = b_1, \ldots, x_{i-1} = b_i] \overset{t}{\Rightarrow} x_i = F$, set $x_i = F$.
   - Else set $x_i \in_U \{T, F\}$.

$PPSZ(\varphi, b, t)$

1. Formula φ, random assignment $b : (b_1, \ldots, b_n)$.
2. Random ordering of variables: $x_1, x_2, \ldots, x_n$
3. For $i = 1$ to n:
   - If $\varphi[x_1 = b_1, \ldots, x_{i-1} = b_i] \overset{t}{\Rightarrow} x_i = T$, set $x_i = T$.
   - Else If $\varphi[x_1 = b_1, \ldots, x_{i-1} = b_i] \overset{t}{\Rightarrow} x_i = F$, set $x_i = F$.
   - Else set $x_i = b_i$.

PPSZ success

Theorem

If $\varphi \in$ *Unique-3-SAT, then* $PPSZ(\varphi, t)$ *finds the unique solution with probability*

$$2^{-0.39n} \sim 1.308^{-n}.$$

PPSZ Example

$$(x \vee y),\ (x \vee \neg y),\ (\neg y \vee \neg x)$$

Prob[$PPSZ(\varphi, 1)$ succeeds] is $\dfrac{1}{2}$.

Prob[$PPSZ(\varphi, 2)$ succeeds] is 1.

Guessed vs Forced variables

π: Permutation of the variables

$Forced(\pi) = \{x_i | \varphi(x_1 = a_1, \ldots, x_{i-1} = a_i) \overset{t}{\Rightarrow} x_i = a_i\}$.

$Guessed(\pi) = \{x_1, \ldots, x_n\} \setminus Forced(\pi)$.

For a fixed π, the probability that PPSZ correctly outputs a is:

$$\left(\frac{1}{2}\right)^{|Guessed(\pi)|}$$

Probability of success

$$
\begin{aligned}
Pr[\text{PPSZ succeeds}] &= E_\pi[\left(\frac{1}{2}\right)^{|Guessed(\pi)|}] \\
&\geq \left(\frac{1}{2}\right)^{E_\pi[Guessed(\pi)]} \\
&= \left(\frac{1}{2}\right)^{\sum_i p_i} \geq 2^{-pn}
\end{aligned}
$$

where $p_i = Pr_\pi[x_i \in Guessed(\pi)] \leq p < 1$.

Probability of success

$$Pr[\text{PPSZ succeeds}] = E_\pi[\left(\frac{1}{2}\right)^{|Guessed(\pi)|}]$$

$$= \left(\frac{1}{2}\right)^{\sum_i p_i} \geq 2^{-pn}$$

where $p_i = Pr_\pi[x_i \in Guessed(\pi)] \leq p < 1$.

Probability of success

$$
\begin{aligned}
Pr[\text{PPSZ succeeds}] &= E_\pi[\left(\frac{1}{2}\right)^{|Guessed(\pi)|}] \\
&\geq \left(\frac{1}{2}\right)^{E_\pi[Guessed(\pi)]} \\
&= \left(\frac{1}{2}\right)^{\sum_i p_i} \geq 2^{-pn}
\end{aligned}
$$

where $p_i = Pr_\pi[x_i \in Guessed(\pi)] \leq p < 1$.

PPSZ Bounds for Unique-3-SAT

Theorem

For a Unique 3-SAT instance, $p \leq 2\log 2 - 1 \sim 0.386$

Forcing clauses

Unique satisfying assignment: $x = y = z = T$.

Then for every variable x, there is a clause of the form:

$$(x \vee \bar{y} \vee \bar{z})$$

$Pr[x \text{ is forced}] \geq$

Forcing clauses

Unique satisfying assignment: $x = y = z = T$.

Then for every variable x, there is a clause of the form:

$$(x \vee \bar{y} \vee \bar{z})$$

$Pr[x \text{ is forced}] \geq \frac{1}{3}$.

Implication in Partial Assignments

Unique sat assigment: $x_1 = T, x_2 = T, \ldots, x_n = T$.

$\varphi[(x_1 = F, x_2 = F, \ldots, x_k = F)]$ has a clause of the form:

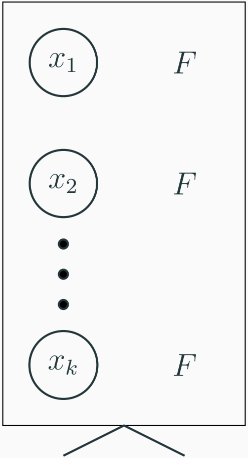
$$\neg x_{k+1}$$

OR

$$\neg x_{k+1} \vee \neg x_{k+2}$$

Otherwise: $(x_1 = \ldots = x_k = F), (x_{k+1} = \ldots = x_n = T)$ satisfies φ.

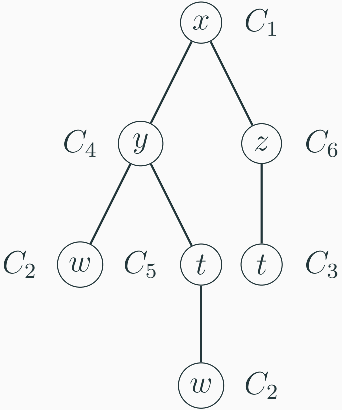
Implication in Partial Assignments



$x_{k+1} = F$ OR $x_{k+2} = F$

Clause Tree

$C_1 : (x, \neg y, \neg z)$
$C_2 : (x, y, w)$
$C_3 : (x, z, t)$
$C_4 : (y, \neg w, \neg t)$
$C_5 : (t, \neg w)$
$C_6 : (z, \neg t)$



Clause Tree

$C_1 : (x, \neg y, \neg z)$

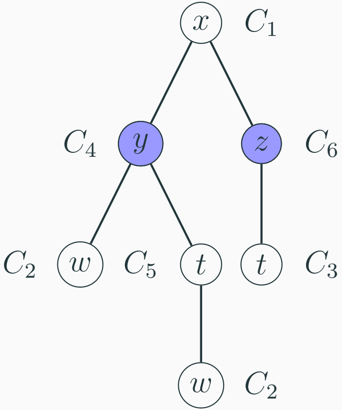
$C_2 : (x, y, w)$

$C_3 : (x, z, t)$

$C_4 : (y, \neg w, \neg t)$

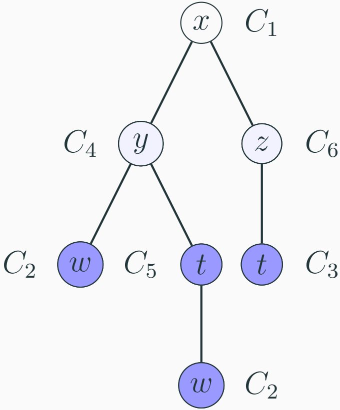
$C_5 : (t, \neg w)$

$C_6 : (z, \neg t)$

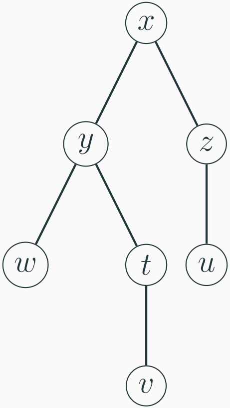


Clause Tree

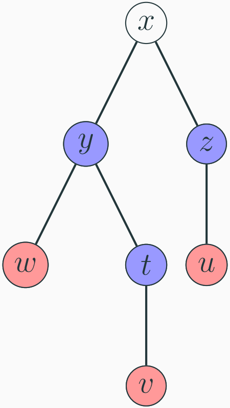
$C_1 : (x, \neg y, \neg z)$
$C_2 : (x, y, w)$
$C_3 : (x, z, t)$
$C_4 : (y, \neg w, \neg t)$
$C_5 : (t, \neg w)$
$C_6 : (z, \neg t)$



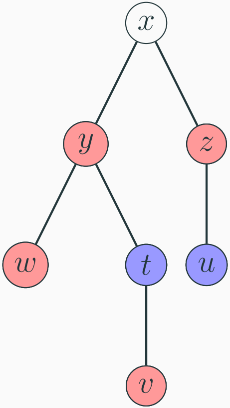
Clause Tree



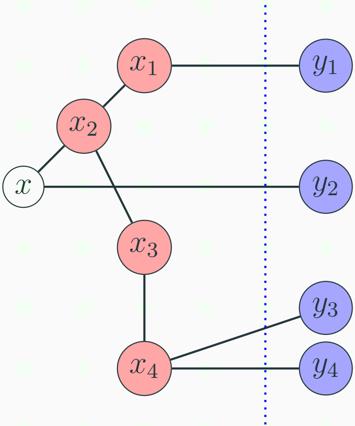
Clause Tree



Clause Tree



Clause Tree



$y_1 = \ldots = y_4 = T$
5-implies
$x_1 = T$.

Clause Tree Lemma

- $R_x = \{v | v \text{ is on a } >_\pi \text{ path from } x\}$.
- If $|R_x| \leq t$, then $x \in Forced(\pi)$ if t-implication is used.

Clause Tree Lemma

- $R_x = \{v | v$ is on a $>_\pi$ path from $x\}$.
- If $|R_x| \leq t$, then $x \in Forced(\pi)$ if t-implication is used.

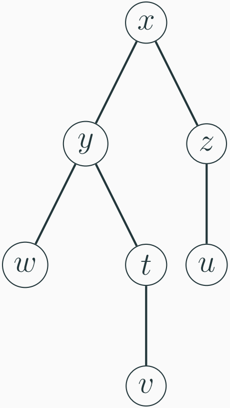
Lemma

If all variables $>_\pi x$ are at distance at most d, then x is 2^d-implied.

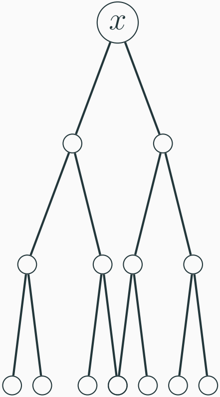
Probability lower bound

Pr[x is forced] $\geq$

Pr[$y >_\pi x \in B(x,d)$]



$\geq$ Pr[$y > x \in B(x,d)$]



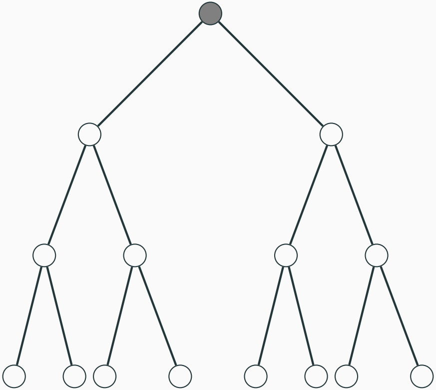
Probability lower bound

Pr[$y > x$ at distance at most d

$\geq$ Pr[$y > x$ at finite distance]$-\varepsilon$

~ 0.61

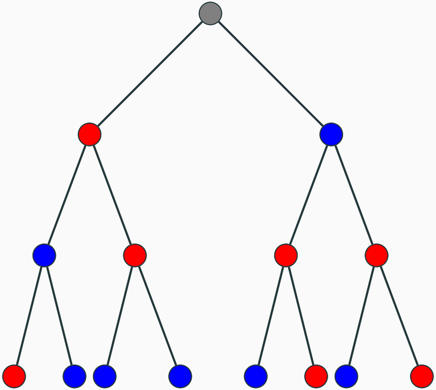
Probability of infinite paths



$$f : V \to [0, 1]$$

$$Pr[root \to x_1 \to x_2 \to \ldots | f(x_1), f(x_2), f(x_3) \ldots > root]$$

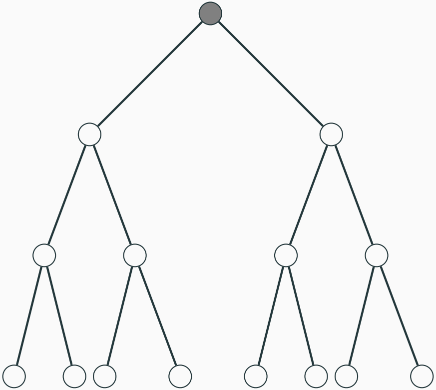
Probability of infinite paths



$$f : V \to [0, 1]$$

$$Pr[root \to x_1 \to x_2 \to \ldots | \\ f(x_1), f(x_2), f(x_3) \ldots > root]$$

Probability of infinite paths



$$P(x) = Pr[\text{Infinite path } \geq x]$$

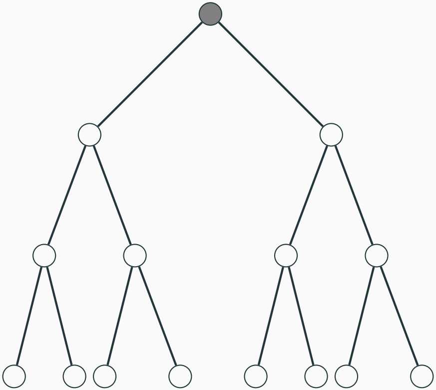
$$P(x) = (1-x)[1-(1-P(x))^2]$$

$$P(x) = \frac{(1-2x)}{(1-x)}$$

Ans: $$\int_0^{1/2} \frac{(1-2x)}{(1-x)^2} dx$$

$$= 2\log 2 - 1 \sim 0.386$$

Probability of infinite paths



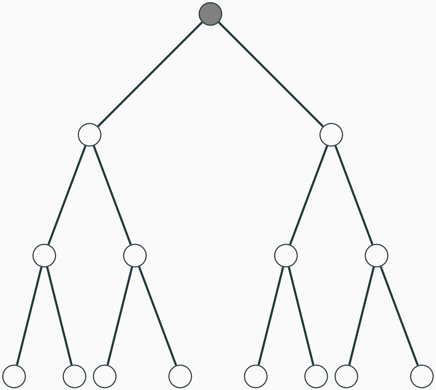
$$P(x) = Pr[\text{Infinite path } \geq x]$$

$$P(x) = (1-x)[1-(1-P(x))^2]$$

Ans: $\int_0^{1/2} \frac{(1-2x)}{(1-x)^2} dx$

$= 2\log 2 - 1 \sim 0.386$

Probability of infinite paths



$$P(x) = Pr[\text{Infinite path } \geq x]$$

$$P(x) = (1-x)[1-(1-P(x))^2]$$

$$P(x) = \frac{(1-2x)}{(1-x)}$$

Ans: $\int_0^{1/2} \frac{(1-2x)}{(1-x)^2} dx$

$$= 2\log 2 - 1 \sim 0.386$$

Not in this talk...

- Parameterized algorithms
- Backdoors to 2-SAT, q-Horn etc
- Random SAT formulas
- Resolution-type exponential algorithms for hard problems