Overview

Introduction

Tractable cases

DPLL algorithms

CDCL solvers

Probabilistic algorithms

Applications

Proof Complexity

Random formulas
  Satisfiability threshold
  Application: the SAT filter
The random $k$-CNF model

A random $k$-CNF formula $F_k(n, m)$ is chosen as follow:
- $m$ times independently choose uniformly one of the $2^k \binom{n}{k}$ clauses.

Let $r$ denote the ratio of clauses to variables: $r := m/n$

**Empirical observation:** For $F = F_3(n, rn)$:
- if $r < 4.26$, then $F$ is likely satisfiable
- if $r > 4.26$, then $F$ is likely unsatisfiable
Conjecture: For every $k \geq 2$ there is a constant $r_k$ such that

$$\lim_{n \to \infty} \Pr[F_k(n, rn) \text{ is satisfiable}] = \begin{cases} 1 & \text{if } r < r_k \\ 0 & \text{if } r > r_k \end{cases}$$

Theorem

For $k = 2$, the conjecture holds with $r_2 = 1$. 
A weaker satisfiability threshold

**Theorem**

For every $k \geq 3$ there is a sequence $r_k(n)$ such that

$$\lim_{n \to \infty} \Pr[F_k(n, m) \text{ is satisfiable}] = \begin{cases} 
1 & \text{if } m \leq (r_k(n) - \epsilon)n \\
0 & \text{if } m \geq (r_k(n) + \epsilon)n
\end{cases}$$

Best known upper and lower bounds for $r_k$:

<table>
<thead>
<tr>
<th>$k$</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>7</th>
<th>10</th>
<th>20</th>
</tr>
</thead>
<tbody>
<tr>
<td>$r_k \leq$</td>
<td>4.51</td>
<td>10.23</td>
<td>21.33</td>
<td>87.88</td>
<td>708.94</td>
<td>726.817</td>
</tr>
<tr>
<td>$r_k \geq$</td>
<td>3.52</td>
<td>7.91</td>
<td>18.79</td>
<td>84.82</td>
<td>704.94</td>
<td>726.809</td>
</tr>
<tr>
<td>algo</td>
<td>3.52</td>
<td>5.54</td>
<td>9.63</td>
<td>33.23</td>
<td>172.65</td>
<td>95.263</td>
</tr>
</tbody>
</table>
An easy upper bound

**Theorem**

\[ r_3 \leq 5.19 \]

**Proof:** Let \( F = F_3(n, rn) \), and \( X := \left| \{ \alpha \in \{0, 1\}^n ; \alpha \models F \} \right| \).

We have:

- \( \Pr[X > 0] \leq E[X] = 2^n \Pr[\alpha \models F] \).
- \( \Pr[\alpha \models F] = \Pr[\alpha \models C]^m = (7/8)^m \)
- Thus \( \Pr[X > 0] \leq 2^n (7/8)^m = \left( 2(7/8)^r \right)^n \)
- Exponentially small if \( 2(7/8)^r < 1 \), i.e., if \( r > -\ln 2 / \ln(7/8) > 5.19 \)
Improvement by properties

Let $P(\alpha)$ be a property of assignments.

Define $X_P := |\{ \alpha \in \{0, 1\}^n ; \alpha \models F \text{ and } P(\alpha) \}|$

Obviously $X_P \leq X$, so if $X > 0$ implies $X_P > 0$, then:
$$\Pr[X > 0] \leq \Pr[X_P > 0] \leq E[X_P]$$

Now if $E[X_P] \ll E[X]$, we can obtain a better upper bound since $E[X_P] \to 0$ for smaller values of $r$. 
The single flip property

**Theorem**

\[ r_3 \leq 4.667 \]

For \( \alpha \models F \) and \( x \in V(F) \) with \( \alpha(x) = 0 \),
let \( \alpha_x = \alpha \) with \( [x := 1] \).

\( \alpha \models F \) has the single flip property \( SF \),
if \( \alpha_x \not\models F \) for all \( x \) with \( \alpha(x) = 0 \).

If \( F \) is satisfiable, then there is \( \alpha \models F \) with \( SF(\alpha) \)

Thus: \( \Pr[X > 0] \leq E[X_{SF}] \).
The expected number of single flip assignments

\[ E[X_{SF}] = (7/8)^n \sum_\alpha \Pr[SF(\alpha) | \alpha \models F] \]

Given \( \alpha \) and \( x \) with \( \alpha(x) = 0 \), \( \Pr[\alpha_x \not\models C] = \frac{n-1}{2} / 7 \binom{n}{3} = 3/(7n) \)

Thus: \( \Pr[\alpha_x \not\models F] = 1 - \left(1 - 3/(7n)\right)^n \)

Let \( n_0(\alpha) = |\{x; \alpha(x) = 0\}|. \)

\[ \Pr[SF(\alpha) | \alpha \models F] = \left(1 - (1 - 3/(7n))^n\right)^{n_0(\alpha)} \]

\[ = \left(1 - e^{-3r/7} + o(1)\right)^{n_0(\alpha)} \]

Thus \( E[X_{SF}] \leq (7/8)^n \left(2 - (1 - 3/(7n))^n\right)^n \)

\[ \leq (7/8)^n \left(2 - e^{-3r/7} + o(1)\right)^n \]

This term is exponentially small for \((7/8)^r(2 - e^{-3r/7}) < 1\), which holds for \( r \geq 4.667 \).
Algorithmic lower bound

Consider the following heuristic algorithm:

repeat \( N \) times

\[ \alpha := [] \]

while \( V(F\alpha) \neq \emptyset \) do

\[ \text{pick literal } a = H(F\alpha) \text{ in } F\alpha \]

\[ \alpha := \alpha \cup [a := 1] \]

if \( \alpha \models F \)

then return \( \alpha \)

Performance depends on heuristic \( H(F\alpha) \).
Pure literal heuristic

Pure literal heuristic $H(F\alpha)$:
- if $F\alpha$ contains a pure literal $a$, pick $a$
- otherwise pick a uniformly random from the literals in $F\alpha$

**Theorem**

For $r < 1.637$, the pure literal heuristic finds $\alpha \models F_3(n, rn)$ with high probability.

For $r \geq 1.7$, the pure literal heuristic fails on $F_3(n, rn)$ with high probability.
Generalized unit clause heuristic

Generalized unit clause heuristic $H(F\alpha)$:

- pick a uniformly random from the literals occurring in minimal width clauses in $F\alpha$

Theorem

For $r < 3.003$, the generalized unit clause heuristic finds $\alpha \models F_3(n, rn)$ with high probability.

For $r \geq 3.003$, the generalized unit clause heuristic fails on $F_3(n, rn)$ with high probability.
Balanced literal heuristic $H(F_\alpha)$:

- if $F_\alpha$ contains a pure literal $a$, pick $a$
- otherwise pick $a$ such that $p(a) - n(a)$ is maximal, where
  - $p(a)$: number of occurrences of $a$
  - $n(a)$: number of occurrences of $\bar{a}$

**Theorem**

For $r < 3.52$, the balanced literal heuristic finds $\alpha \models F_3(n, rn)$ with high probability.

In particular, $r_3 \geq 3.52$. 
Membership filters

A membership filter is a data structure that maintains a subset $Y \subset D$ of a large domain $D$.

It supports the operation $\text{query}(x)$ for $x \in D$ with the property:

- $\text{query}(x) = \text{No} \; \rightarrow \; x \notin Y$
- $\text{query}(x) = \text{Yes} \; \rightarrow \; x \in Y$ with high probability.

Applications: fast preliminary membership test
safety-critical test where false positives do not matter
Hash functions

Hash function $h : D \rightarrow [n]$, where $n \ll |D|$.

Assumption for analysis:

- $h(x)$ in uniformly random,
  i.e., $\Pr[h(x) = i] = 1/n$ for $x \in D$ and $i < n$,
- for $x \neq y \in D$, $h(x)$ and $h(y)$ are independent.

Hash function as membership filter: Fingerprinting

- Store the set $F := \{h(y) ; y \in Y\}$.
- To query $x$, test whether $h(x) \in F$. 
The Bloom filter

Let \( h_1, \ldots, h_k \) be hash functions \( h_i : D \to [n] \).

\( B \) boolean array of size \( n \)

\[
\text{buildBloom}(Y) \\
B := (0, \ldots, 0) \\
\text{for } x \in Y \text{ do} \\
\quad \text{for } i := 1 \text{ to } k \text{ do} \\
\quad \quad j := h_i(x) \\
\quad \quad B[j] := 1
\]

\[
\text{queryBloom}(x) \\
\quad \text{for } i := 1 \text{ to } k \text{ do} \\
\quad \quad j := h_i(x) \\
\quad \quad \text{if } B[j] = 0 \\
\quad \quad \quad \text{then return No} \\
\quad \quad \text{return Yes}
\]
Analysis of the Bloom filter

Let $m := |Y|$. 

Probability $\Pr[B[j] = 0] = (1 - 1/n)^{km} \approx e^{-km/n} =: p$

Probability of a false positive:

$$(1 - (1 - 1/n)^{km})^k \approx (1e^{-km/n})^k = (1 - p)^k =: f$$

Let $g := \ln f = k \ln(1 - e^{-km/n})$.

Minimize $g$ to find optimal number $k$ of hash functions.

$g$ is minimal for $k = n/m \cdot \ln 2$, where $f = 1/2^k = (0.6185n/m)$.

E.g. for $n = 8m$, we get $k = 6$ with prob. of false positives of about $2\%$. 
The SAT filter: clauses from elements

Let $h_1, \ldots, h_k$ be hash functions $h_i : D \times \mathbb{N} \rightarrow \{-n, \ldots, n\} \setminus \{0\}$.

Interpret value $i \leq n$ as $x_i$, and $-i$ as $\overline{x}_i$

makeClause($x$)

$n := 0$
repeat
  $n := n + 1$
  for $i := 1$ to $k$ do
    $a := h_i(x, n)$
    $C := C \lor a$
  until $w(C) = k$ and $C$ non-tautological
return $C$
The SAT filter: building and querying

\( \alpha \) assignment to variables \( x_1, \ldots, x_n \)

\begin{align*}
\text{buildSAT}(Y) & \\
& F := 1 \\
& \text{for } x \in Y \text{ do} \\
& \quad C := \text{makeClause}(x) \\
& \quad F := F \land C \\
& \alpha := \text{solve}(F)
\end{align*}

\begin{align*}
\text{querySAT}(x) & \\
& C := \text{makeClause}(x) \\
& \text{if } \alpha \models C \\
& \quad \text{then return Yes} \\
& \quad \text{else return No}
\end{align*}
Given \( m \), choose \( n \) and \( k \) so that \( F \) is satisfiable, e.g., so that \( m/n \leq 2^k \ln 2 - k \).

Probability of a false positive: \( p = (1 - 2^{-k}) \)

Efficiency: \[ \frac{-\log p}{n/m} = -(2^k \ln 2 - k) \log(1 - (2^{-k})) \]

- Bloom filter: optimal \( k \) with efficiency \( \ln 2 \)
- SAT filter: efficiency tends to 1 as \( k \to \infty \)