Hiding Satisfying Assignments: 
*Two are Better than One*

Dimitris Achlioptas\(^1\), Haixia Jia\(^2\) and Cris Moore\(^2\)

\(^1\) Microsoft Research, \(^2\) University of New Mexico
Outline of the talk

- Background
Outline of the talk

- Background
- Our proposal
Outline of the talk

- Background
- Our proposal
- Space of solutions
Outline of the talk

- Background
- Our proposal
- Space of solutions
- Unit Clause heuristic and DPLL algorithms
Outline of the talk

- Background
- Our proposal
- Space of solutions
- Unit Clause heuristic and DPLL algorithms
- Experimental results
Outline of the talk

- Background
- Our proposal
- Space of solutions
- Unit Clause heuristic and DPLL algorithms
- Experimental results
- Conclusion and future work
Background:

Satisfiability (3-SAT) is the canonical NP-complete problem:
Background:

Satisfiability (3-SAT) is the canonical NP-complete problem:

Given a formula $\varphi$ with $m$ clauses $C_1, \ldots, C_m$ over $n$ variables.
**Background:**

Satisfiability (3-SAT) is the canonical NP-complete problem:

Given a formula $\varphi$ with $m$ clauses $C_1, \ldots C_m$ over $n$ variables.

Check if there exist TRUE/FALSE assignments to the variables that makes the formula satisfiable.
Background:

Satisfiability (3-SAT) is the canonical NP-complete problem:

Given a formula $\varphi$ with $m$ clauses $C_1, \ldots, C_m$ over $n$ variables.

Check if there exist TRUE/FALSE assignments to the variables that makes the formula satisfiable.

Example: $(v_1 \lor \overline{v_2} \lor v_3) \land (v_2 \lor v_3 \lor \overline{v_4})$
Random 3-SAT formulas:

A random 3-SAT formula $\varphi(n, m)$ can be constructed in the following:

1. Pick 3 variables $v_i, v_j, v_k$ from $n$ variables randomly.
2. Pick a sign (TRUE or FALSE) for each variable randomly.
3. Form a clause $\text{sign}(v_i) \lor \text{sign}(v_j) \lor \text{sign}(v_k)$.
4. Repeat above steps $m$ times to generate $m$ clauses.

We will call this 0-hidden formula.
Random 3-SAT formulas:

A random 3-SAT formula \( \varphi(n, m) \) can be constructed in the following:

- Pick 3 variables \( v_i, v_j, v_k \) from \( n \) variables randomly;

We will call this 0-hidden formula

Hiding Satisfying Assignments: – p.4/31
Random 3-SAT formulas:

A random 3-SAT formula $\varphi(n, m)$ can be constructed in the following:

- Pick 3 variables $v_i, v_j, v_k$ from $n$ variables randomly;
- Pick a sign (TRUE or FALSE) for each variable randomly;
Random 3-SAT formulas:

A random 3-SAT formula $\varphi(n, m)$ can be constructed in the following:

- Pick 3 variables $v_i, v_j, v_k$ from $n$ variables randomly;
- Pick a sign (TRUE or FALSE) for each variable randomly;
- Form a clause
  
  \[ (\text{sign}(v_i) \lor \text{sign}(v_j) \lor \text{sign}(v_k)) \]
Random 3-SAT formulas:

A random 3-SAT formula $\varphi(n, m)$ can be constructed in the following:

- Pick 3 variables $v_i, v_j, v_k$ from $n$ variables randomly;
- Pick a sign (TRUE or FALSE) for each variable randomly;
- Form a clause 
  \[(\text{sign}(v_i) \lor \text{sign}(v_j) \lor \text{sign}(v_k))\];

Repeat above steps m times to generate m clauses.
Random 3-SAT formulas:

A random 3-SAT formula $\varphi(n, m)$ can be constructed in the following:

- Pick 3 variables $v_i, v_j, v_k$ from $n$ variables randomly;
- Pick a sign (TRUE or FALSE) for each variable randomly;
- Form a clause
  $$(\text{sign}(v_i) \lor \text{sign}(v_j) \lor \text{sign}(v_k));$$

Repeat above steps $m$ times to generate $m$ clauses.

We will call this 0-hidden formula
When $r = m/n < 4.25$, the random instance almost always has a solution as $n \to \infty$. When $r > 4.25$, almost always no solution as $n \to \infty$. When $r = 4.25$, the formula seems to be hardest.
When $r = m/n < 4.25$, the random instance almost always has a solution as $n \to \infty$

When $r > 4.25$, almost always no solution as $n \to \infty$
When $r = m/n < 4.25$, the random instance almost always has a solution as $n \to \infty$

When $r > 4.25$, almost always no solution as $n \to \infty$

When $r \approx 4.25$, the formula seems to be hardest.
Why do we want to generate random hard satisfiable 3-SAT formulas?
Why do we want to generate random hard *satisfiable* 3-SAT formulas?

To develop and test *incomplete* search algorithms for SAT problems.
For incomplete search algorithms, such as WalkSAT, what if we don’t find a solution for a random 3-SAT formula?
For incomplete search algorithms, such as WalkSAT, what if we don’t find a solution for a random 3-SAT formula?

We could use a complete method to filter out the unsatisfiable cases...
For incomplete search algorithms, such as WalkSAT, what if we don’t find a solution for a random 3-SAT formula?

We could use a complete method to filter out the unsatisfiable cases...

...but this limits the size and difficulty of problems
For incomplete search algorithms, such as WalkSAT, what if we don’t find a solution for a random 3-SAT formula?

We could use a complete method to filter out the unsatisfiable cases...

...but this limits the size and difficulty of problems

Ideally, we want problem generators that generate satisfiable instances only
A naive attempt

\( \varphi(n, m) : \)
A naive attempt

\( \varphi(n, m) : \)

Choose a random truth assignment \( A \) of \( n \) variables;
A naive attempt

\( \varphi(n, m) : \)

Choose a random truth assignment \( A \) of \( n \) variables;

- Pick 3 variables \( v_i, v_j, v_k \) from \( n \) variables randomly;
A naive attempt

\[ \varphi(n, m) : \]
Choose a random truth assignment \( A \) of \( n \) variables;

- Pick 3 variables \( v_i, v_j, v_k \) from \( n \) variables randomly;
- choose a clause randomly from among the 7 clauses satisfied by \( A \);
A naive attempt

\( \varphi(n, m) \):

Choose a random truth assignment \( A \) of \( n \) variables;

- Pick 3 variables \( v_i, v_j, v_k \) from \( n \) variables randomly;
- choose a clause randomly from among the 7 clauses satisfied by \( A \);

Repeat above 2 steps \( m \) times to generate \( m \) clauses.
A naive attempt

\( \varphi(n, m) \):
Choose a random truth assignment \( A \) of \( n \) variables;

- Pick 3 variables \( v_i, v_j, v_k \) from \( n \) variables randomly;
- choose a clause randomly from among the 7 clauses satisfied by \( A \);

Repeat above 2 steps \( m \) times to generate \( m \) clauses.

The truth assignment \( A \) is the hidden solution;
A naive attempt

$\varphi(n, m)$:
Choose a random truth assignment $A$ of $n$ variables;

- Pick 3 variables $v_i, v_j, v_k$ from $n$ variables randomly;
- choose a clause randomly from among the 7 clauses satisfied by $A$;

Repeat above 2 steps $m$ times to generate $m$ clauses.

The truth assignment $A$ is the hidden solution;
We call this a 1-hidden formula.
Example

3 variables $v_1, v_2, v_3$, $A = \{v_1, v_2, v_3\}$. 8 possible ways to form a clause from these 3 variables:

- $(v_1, v_2, v_3)$
- $(v_1, v_2, \overline{v_3})$
- $(v_1, \overline{v_2}, v_3)$
- $(v_1, \overline{v_2}, \overline{v_3})$
- $(\overline{v_1}, v_2, v_3)$
- $(\overline{v_1}, v_2, \overline{v_3})$
- $(\overline{v_1}, \overline{v_2}, v_3)$
- $(\overline{v_1}, \overline{v_2}, \overline{v_3})$
Example

3 variables \( v_1, v_2, v_3 \), \( A = \{ v_1, v_2, v_3 \} \). 8 possible ways to form a clause from these 3 variables:

\[
\begin{align*}
(v_1, v_2, v_3) \\
(v_1, v_2, \overline{v_3}) \\
(v_1, \overline{v_2}, v_3) \\
(v_1, \overline{v_2}, \overline{v_3}) \\
(\overline{v_1}, v_2, v_3) \\
(\overline{v_1}, v_2, \overline{v_3}) \\
(\overline{v_1}, \overline{v_2}, v_3) \\
(\overline{v_1}, \overline{v_2}, \overline{v_3})
\end{align*}
\]

\((\overline{v_1}, \overline{v_2}, \overline{v_3}) \iff \text{not satisfied by } A\)
Problems:

The problems with this generator are:
Problems:

The problems with this generator are:
Too easy to solve
Problems:

The problems with this generator are:

Too easy to solve

Assignment $A$ acts as an attractor for algorithms like WalkSAT or DPLL (especially at high density)
Why 1-hidden assignments attract

3 variables $v_1, v_2, v_3$, $A = \{v_1, v_2, v_3\}$. 7 clauses satisfied by $A$:

$(v_1, v_2, v_3)$
$(v_1, v_2, \overline{v}_3)$
$(v_1, \overline{v}_2, v_3)$
$(v_1, \overline{v}_2, \overline{v}_3)$
$(\overline{v}_1, v_2, v_3)$
$(\overline{v}_1, v_2, \overline{v}_3)$
$(\overline{v}_1, \overline{v}_2, v_3)$
$(\overline{v}_1, \overline{v}_2, \overline{v}_3)$
$(\overline{v}_1, \overline{v}_2, \overline{v}_3)$
Why 1-hidden assignments attract

3 variables \( v_1, v_2, v_3 \), \( A = \{ v_1, v_2, v_3 \} \). 7 clauses satisfied by \( A \):

\[
\begin{align*}
(v_1, v_2, v_3) \\
(v_1, v_2, \overline{v_3}) \\
(v_1, \overline{v_2}, v_3) \\
(v_1, \overline{v_2}, \overline{v_3}) \\
(\overline{v_1}, v_2, v_3) \\
(\overline{v_1}, v_2, \overline{v_3}) \\
(\overline{v_1}, \overline{v_2}, v_3) \\
(\overline{v_1}, \overline{v_2}, \overline{v_3})
\end{align*}
\]
Why 1-hidden assignments attract

3 variables $v_1, v_2, v_3$, $A = \{v_1, v_2, v_3\}$. 7 clauses satisfied by $A$:

$(v_1, v_2, v_3)$
$(v_1, v_2, \overline{v_3})$
$(v_1, \overline{v_2}, v_3)$
$(v_1, \overline{v_2}, \overline{v_3})$
$(\overline{v_1}, v_2, v_3)$
$(\overline{v_1}, v_2, \overline{v_3})$
$(\overline{v_1}, \overline{v_2}, v_3)$
$(\overline{v_1}, \overline{v_2}, \overline{v_3})$
Why 1-hidden assignments attract

3 variables $v_1, v_2, v_3$, $A = \{v_1, v_2, v_3\}$. 7 clauses satisfied by $A$:

$(v_1, v_2, v_3)$
$(v_1, v_2, \overline{v_3})$
$(v_1, \overline{v_2}, v_3)$
$(v_1, \overline{v_2}, \overline{v_3})$
$(\overline{v_1}, v_2, v_3)$
$(\overline{v_1}, v_2, \overline{v_3})$
$(\overline{v_1}, \overline{v_2}, v_3)$
$(\overline{v_1}, \overline{v_2}, \overline{v_3})$
Why 1-hidden assignments attract

3 variables $v_1, v_2, v_3$, $A = \{v_1, v_2, v_3\}$. 7 clauses satisfied by $A$:

$(v_1, v_2, v_3)$
$(v_1, v_2, \overline{v_3})$
$(v_1, \overline{v_2}, v_3)$
$(v_1, \overline{v_2}, \overline{v_3})$
$(\overline{v_1}, v_2, v_3)$
$(\overline{v_1}, v_2, \overline{v_3})$
$(\overline{v_1}, \overline{v_2}, v_3)$
$(\overline{v_1}, \overline{v_2}, \overline{v_3})$

4/7 of the time, WalkSAT or the majority heuristic for DPLL point toward $A$
Our proposal:

Hide two assignments, $A$ and $\overline{A}$

Choose a random truth assignment $A$ of $n$ variables.
Our proposal:

Hide two assignments, $A$ and $\overline{A}$

Choose a random truth assignment $A$ of $n$ variables.

- Pick 3 variables $v_i, v_j, v_k$ from $n$ variables randomly;
Our proposal:

Hide two assignments, $A$ and $\overline{A}$

Choose a random truth assignment $A$ of $n$ variables.

- Pick 3 variables $v_i, v_j, v_k$ from $n$ variables randomly;
- choose a clause randomly from among the 6 clauses satisfied by both $A$ and $\overline{A}$;
Our proposal:

Hide two assignments, $A$ and $\overline{A}$

Choose a random truth assignment $A$ of $n$ variables.

- Pick 3 variables $v_i, v_j, v_k$ from $n$ variables randomly;
- choose a clause randomly from among the 6 clauses satisfied by both $A$ and $\overline{A}$;

Repeat above steps $m$ time to generate $m$ clauses.
Our proposal:

Hide two assignments, $A$ and $\overline{A}$

Choose a random truth assignment $A$ of $n$ variables.
- Pick 3 variables $v_i, v_j, v_k$ from $n$ variables randomly;
- choose a clause randomly from among the 6 clauses satisfied by both $A$ and $\overline{A}$;

Repeat above steps $m$ time to generate $m$ clauses.

We call this a 2-hidden formula.
Our proposal:

Hide two assignments, \( A \) and \( \overline{A} \)

Choose a random truth assignment \( A \) of \( n \) variables.

- Pick 3 variables \( v_i, v_j, v_k \) from \( n \) variables randomly;
- choose a clause randomly from among the 6 clauses satisfied by both \( A \) and \( \overline{A} \);

Repeat above steps \( m \) time to generate \( m \) clauses.

We call this a 2-hidden formula

We hope that the effects of the two attractors cancel out
Example

3 variables \( v_1, v_2, v_3 \), \( A = \{ v_1, v_2, v_3 \} \). 8 possible ways to form a clause from these 3 variables:

\[
\begin{align*}
( v_1, v_2, v_3 ) \\
( v_1, v_2, \overline{v_3} ) \\
( v_1, \overline{v_2}, v_3 ) \\
( v_1, \overline{v_2}, \overline{v_3} ) \\
( \overline{v_1}, v_2, v_3 ) \\
( \overline{v_1}, v_2, \overline{v_3} ) \\
( \overline{v_1}, \overline{v_2}, v_3 ) \\
( \overline{v_1}, \overline{v_2}, \overline{v_3} )
\end{align*}
\]
Example

3 variables $v_1, v_2, v_3, A = \{v_1, v_2, v_3\}$. 8 possible ways to form a clause from these 3 variables:

$(v_1, v_2, v_3) \iff$ not satisfied by $\overline{A}$

$(v_1, v_2, \overline{v_3})$

$(v_1, \overline{v_2}, v_3)$

$(v_1, \overline{v_2}, \overline{v_3})$

$(\overline{v_1}, v_2, v_3)$

$(\overline{v_1}, v_2, \overline{v_3})$

$(\overline{v_1}, \overline{v_2}, v_3)$

$(\overline{v_1}, \overline{v_2}, \overline{v_3}) \iff$ not satisfied by $A$
Example

3 variables \( v_1, v_2, v_3 \), \( A = \{ v_1, v_2, v_3 \} \). 6 clauses satisfied by both \( A \) and \( \overline{A} \):

\[
\begin{align*}
(v_1, v_2, \overline{v}_3) \\
(v_1, \overline{v}_2, v_3) \\
(v_1, \overline{v}_2, \overline{v}_3) \\
(\overline{v}_1, v_2, v_3) \\
(\overline{v}_1, v_2, \overline{v}_3) \\
(\overline{v}_1, \overline{v}_2, v_3) \\
(\overline{v}_1, \overline{v}_2, \overline{v}_3)
\end{align*}
\]
Example

3 variables $v_1, v_2, v_3$, $A = \{v_1, v_2, v_3\}$. 6 clauses satisfied by both $A$ and $\overline{A}$:

$$(v_1, v_2, \overline{v_3})$$
$$(v_1, \overline{v_2}, v_3)$$
$$(v_1, \overline{v_2}, \overline{v_3})$$
$$(\overline{v_1}, v_2, v_3)$$
$$(\overline{v_1}, v_2, \overline{v_3})$$
$$(\overline{v_1}, \overline{v_2}, v_3)$$
Example

3 variables $v_1, v_2, v_3$, $A = \{v_1, v_2, v_3\}$. 6 clauses satisfied by both $A$ and $\overline{A}$:

$(v_1, v_2, \overline{v_3})$
$(v_1, \overline{v_2}, v_3)$
$(v_1, \overline{v_2}, \overline{v_3})$
$(\overline{v_1}, v_2, v_3)$
$(\overline{v_1}, \overline{v_2}, \overline{v_3})$
$(\overline{v_1}, \overline{v_2}, v_3)$
Example

3 variables $v_1, v_2, v_3$, $A = \{v_1, v_2, v_3\}$. 6 clauses satisfied by both $A$ and $\overline{A}$:

$(v_1, v_2, \overline{v_3})$
$(v_1, \overline{v_2}, v_3)$
$(v_1, \overline{v_2}, \overline{v_3})$
$(\overline{v_1}, v_2, v_3)$
$(\overline{v_1}, v_2, \overline{v_3})$
$(\overline{v_1}, \overline{v_2}, v_3)$
$(\overline{v_1}, \overline{v_2}, \overline{v_3})$
Example

3 variables $v_1, v_2, v_3$, $A = \{v_1, v_2, v_3\}$. 6 clauses satisfied by both $A$ and $\overline{A}$:

$(v_1, v_2, \overline{v}_3)$
$(v_1, \overline{v}_2, v_3)$
$(v_1, \overline{v}_2, \overline{v}_3)$
$(\overline{v}_1, v_2, v_3)$
$(\overline{v}_1, v_2, \overline{v}_3)$
$(\overline{v}_1, \overline{v}_2, v_3)$

Equal probability toward both direction
Space of solutions

Let $X$ be the number of satisfying truth assignments in a random $k$-SAT formula.
Space of solutions

Let $X$ be the number of satisfying truth assignments in a random $k$-SAT formula

$$E[X] = \sum_{\sigma \in \{0,1\}^n} \Pr[\sigma \text{ is satisfying}]$$
Space of solutions

Let $X$ be the number of satisfying truth assignments in a random $k$-SAT formula

$$E[X] = \sum_{\sigma \in \{0,1\}^n} \Pr[\sigma \text{ is satisfying}]$$

$$= \sum_{z=0}^{n} \binom{n}{z} \Pr \left[ \text{a truth assignment with } z \text{ 1s satisfies a random clause} \right]^m$$
Space of solutions

Let $X$ be the number of satisfying truth assignments in a random $k$-SAT formula.

$$E[X] = \sum_{\sigma \in \{0,1\}^n} \Pr[\sigma \text{ is satisfying}]$$

$$= \sum_{z=0}^{n} \binom{n}{z} \Pr[\text{a truth assignment with } z \text{ 1s satisfies a random clause}]^m$$

$$= \sum_{z=0}^{n} \binom{n}{z} \left(1 - \sum_{j=1}^{k} \binom{k}{j} \frac{(1-z/n)^j (z/n)^{k-j}}{2^k - 1}\right)^m$$
Space of solutions

Let $X$ be the number of satisfying truth assignments in a random k-SAT formula

$$E[X] = \sum_{\sigma \in \{0,1\}^n} \Pr[\text{$\sigma$ is satisfying}]$$

$$= \sum_{z=0}^{n} \binom{n}{z} \Pr\left[\text{a truth assignment with $z$ 1s satisfies a random clause}\right]^m$$

$$= \sum_{z=0}^{n} \binom{n}{z} \left(1 - \sum_{j=1}^{k} \binom{k}{j} \frac{(1-z/n)^j (z/n)^{k-j}}{2^k-1}\right)^m$$

$$= \sum_{z=0}^{n} \binom{n}{z} \left(1 - \frac{1-(z/n)^k}{2^k-1}\right)^m$$
1-hidden formulas

Let $\alpha = \frac{z}{n}$, the fraction of variables that agree with $A$
1-hidden formulas

Let $\alpha = z/n$, the fraction of variables that agree with $A$

$$E[X] = \sum_{z=0}^{n} \binom{n}{z} \left( 1 - \frac{1-(z/n)^k}{2^k-1} \right)^m$$
1-hidden formulas

Let $\alpha = z/n$, the fraction of variables that agree with $A$

$$E[X] = \sum_{z=0}^{n} \binom{n}{z} \left(1 - \frac{1-(z/n)^k}{2^k-1}\right)^m$$

$$\sim \max_{\alpha \in [0,1]} \left[ \frac{1}{\alpha^\alpha (1-\alpha)^{1-\alpha}} \left(1 - \frac{1-\alpha^k}{2^k-1}\right)^r \right]^n$$
2-hidden formulas

\[ E[X] = \sum_{\sigma \in \{0,1\}^n} \Pr[\sigma \text{ is satisfying}] \]

\[ = \sum_{z=0}^n \binom{n}{z} \Pr \left[ \text{a truth assignment with } z \text{ 1s satisfies a random clause} \right] \]

\[ = \sum_{z=0}^n \binom{n}{z} \left( 1 - \sum_{j=1}^k \binom{k}{j} \frac{(1-z/n)^j (z/n)^{k-j}}{2^k - 2} \right)^m \]

\[ = \sum_{z=0}^n \binom{n}{z} \left( 1 - \frac{1-(z/n)^k}{2^k - 2} \right)^m \]

\[ \sim \max_{\alpha \in [0,1]} \left[ \frac{1}{\alpha^\alpha (1-\alpha)^{1-\alpha}} \left( 1 - \frac{1-\alpha^k-(1-\alpha)^k}{2^k - 2} \right)^m \right] \]
2-hidden formulas

\[ E[X] = \sum_{\sigma \in \{0,1\}^n} \Pr[\sigma \text{ is satisfying}] \]

\[ = \sum_{z=0}^{n} \binom{n}{z} \Pr \left[ \text{a truth assignment with } z \text{ 1s satisfies a random clause} \right] \]

\[ = \sum_{z=0}^{n} \binom{n}{z} \left(1 - \sum_{j=1}^{k} \binom{k}{j} \frac{(1-z/n)^j (z/n)^{k-j}}{2^k-2} \right)^m \]

\[ = \sum_{z=0}^{n} \binom{n}{z} \left(1 - \frac{1-(z/n)^k}{2^k-2} \right)^m \]

\[ \sim \max_{\alpha \in [0,1]} \left[ \frac{1}{\alpha^\alpha (1-\alpha)^{1-\alpha}} \left(1 - \frac{1-\alpha^k -(1-\alpha)^k}{2^k-2} \right)^r \right]^n \]

**Symmetric** if we flip \( \alpha \) around \( 1/2 \), exchanging \( \alpha \) and \( 1 - \alpha \)
Space of solutions

1-hidden($k = 5$):

![Graph showing space of solutions for 1-hidden with $k = 5$ and different values of $r$. The graph illustrates the relationship between $\alpha$ and $r$ with distinct curves for each value of $r$.](image-url)
Space of solutions

2-hidden($k = 5$):
Unit Clause

Unit Clause (UC) is a linear time heuristic:
Unit Clause

Unit Clause (UC) is a linear time heuristic:
if there are unit clauses, satisfy them;
Unit Clause

Unit Clause (UC) is a linear time heuristic: if there are unit clauses, satisfy them; else pick a random literal and satisfy it.
Facts:

We can think of UC as the first branch of a simple Davis-Putnam-Logemann-Loveland (DPLL) algorithm.
Facts:

We can think of UC as the first branch of a simple Davis-Putnam-Logemann-Loveland (DPLL) algorithm.

Chao and Franco showed that UC succeeds with constant probability on random 3-SAT formulas with $r < 8/3$, and fails w.h.p. for $r > 8/3$. 

Achlioptas, Beame and Molloy proved that exponential behavior of DPLL occurs for $r > 8/3$.

Physical calculations from Cocco and Monasson suggest that exponential behavior begins right at the density where UC begins to fail, i.e., $r = 8/3$. 

Hiding Satisfying Assignments: – p.21/31
Facts:

We can think of UC as the first branch of a simple Davis-Putnam-Logemann-Loveland (DPLL) algorithm.

Chao and Franco showed that UC succeeds with constant probability on random 3-SAT formulas with $r < 8/3$, and fails w.h.p. for $r > 8/3$.

Achlioptas, Beame and Molloy proved that exponential behavior of DPLL occurs for $r > 3.81$. 

Hiding Satisfying Assignments: – p.21/31
Facts:

We can think of UC as the first branch of a simple Davis-Putnam-Logemann-Loveland (DPLL) algorithm.

Chao and Franco showed that UC succeeds with constant probability on random 3-SAT formulas with $r < \frac{8}{3}$, and fails w.h.p. for $r > \frac{8}{3}$.

Achlioptas, Beame and Molloy proved that exponential behavior of DPLL occurs for $r > 3.81$.

Physical calculations from Cocco and Monasson suggest that exponential behavior begins right at the density where UC begins to fail. i.e., $r = \frac{8}{3}$. 
We believe:
For a given DPLL algorithm $A$, let $r_A$ be the largest value of $r$ for which a single branch of $A$ succeeds with constant probability. Then algorithm $A$ takes exponential time for $r > r_A$. 
Differential equation result

We analyze UC on random 1 and 2 hidden formulas by differential equation method.

\[
\frac{ds_{3,j}}{dx} = -\frac{3s_{3,j}}{1 - x}
\]

\[
\frac{ds_{2,j}}{dx} = -\frac{2s_{2,j}}{1 - x} + \frac{m_F(j + 1)s_{3,j+1} + m_T(3 - j)s_{3,j}}{(m_T + m_F)(1 - x)}
\]

More complicated than UC on 0-hidden formulas, but with the same symmetry...
Differential equation result

UC succeeds on 0-hidden formulas with constant probability iff $r < \frac{8}{3}$;
Differential equation result

UC succeeds on 0-hidden formulas with constant probability iff $r < 8/3$;

UC succeeds on 2-hidden formulas with constant probability iff $r < 8/3$;
Differential equation result

UC succeeds on 0-hidden formulas with constant probability iff \( r < \frac{8}{3}; \)

UC succeeds on 2-hidden formulas with constant probability iff \( r < \frac{8}{3}; \)

(UC succeeds on 1-hidden formulas with constant probability iff \( r < 2.679 \))
Differential equation result

UC succeeds on 0-hidden formulas with constant probability iff \( r < \frac{8}{3}; \)

UC succeeds on 2-hidden formulas with constant probability iff \( r < \frac{8}{3}; \)

(UC succeeds on 1-hidden formulas with constant probability iff \( r < 2.679 \))

We expect simple DPLL algorithms to start taking exponential time on 2-hidden formulas at the same density they do so for 0-hidden ones.
**zChaff result**

2-hidden formulas are almost as hard as 0-hidden ones.
Survey Propagation

2-hidden formulas are harder than 1-hidden ones
WalkSAT result \((n = 300)\)

2-hidden formulas are as hard as 0-hidden ones
WalkSAT result ($r = 4.25$)

2-hidden formulas are as hard as 0-hidden ones, 1-hidden are easier (both are polynomial [Barthel et al.])
Conclusions

We introduced an extremely simple new generator of random satisfiable 3-SAT instances;
Conclusions

We introduced an extremely simple new generator of random satisfiable 3-SAT instances; It is amenable to all the mathematical tools developed for the study of random 3-SAT instances;
Conclusions

We introduced an extremely simple new generator of random satisfiable 3-SAT instances;

It is amenable to all the mathematical tools developed for the study of random 3-SAT instances;

Experimentally, our generator appears to produce instances that are as hard as random 3-SAT instances, in sharp contrast to instances with a single hidden assignment.
Future works

Proving that the expected running time of natural Davis-Putnam algorithms on 2-hidden formulas is exponential in $n$. 
Future works

Proving that the expected running time of natural Davis-Putnam algorithms on 2-hidden formulas is exponential in \( n \).

Explaining the different threshold behaviors of Survey Propagation on 1-hidden and 2-hidden formulas.
Future works

Proving that the expected running time of natural Davis-Putnam algorithms on 2-hidden formulas is exponential in $n$.

Explaining the different threshold behaviors of Survey Propagation on 1-hidden and 2-hidden formulas.

Understanding how long WalkSAT takes at the midpoint between the two hidden assignments, before it becomes sufficiently unbalanced to converge to one of them.
Future works

Proving that the expected running time of natural Davis-Putnam algorithms on 2-hidden formulas is exponential in $n$.

Explaining the different threshold behaviors of Survey Propagation on 1-hidden and 2-hidden formulas.

Understanding how long WalkSAT takes at the midpoint between the two hidden assignments, before it becomes sufficiently unbalanced to converge to one of them.

Studying random 2-hidden formulas in the dense case where there are $\omega(n)$ clauses.
Acknowledgements

C.M. and H.J. are supported by NSF grant PHY-0200909.

H.J. is supported by an NSF Graduate Research Fellowship.