A better algorithm for random k-SAT

Amin Coja-Oghlan

University of Edinburgh

The k-SAT problem

The k-SAT problem

- Given: a Boolean formula Φ in *conjunctive normal form*.
- The clauses have *length k*.
- Task: decide whether there is a satisfying assignment.
- This problem is well known to be *NP-hard*.

k-SAT is hard

Worst-case running time

- Suppose Φ is a k-SAT formula with n variables.
- There are 2^n possible assignments.
- We could solve Φ by *trying all of them* (in principle).
- But if $n = 1,000$, then this is *infeasible*.
- However, no better algorithm is known to solve all inputs!

k-SAT is hard

Worst-case running time

- Suppose Φ is a k-SAT formula with n variables.
- There are 2^n possible assignments.
- We could solve Φ by *trying all of them* (in principle).
- But if $n = 1,000$, then this is *infeasible*.
- However, no better algorithm is known to solve all inputs!

Complexity Theory

- provides methods for *classifying* how hard a problems is. . .
- . . . *relative* to other problems.
- NP-complete $=$ as hard as k-SAT (with $k \geq 3$).
- No *absolute* measure of hardness.

k-SAT is hard

Worst-case running time

- Suppose Φ is a k-SAT formula with n variables.
- There are 2^n possible assignments.
- We could solve Φ by *trying all of them* (in principle).
- But if $n = 1,000$, then this is *infeasible*.
- However, no better algorithm is known to solve all inputs!

Questions

- What makes a k-SAT formula *hard* ?
- What types of inputs are *easy*?

Random k-SAT

- Aim: *contrive* hard (but satisfiable) forumlas.
- Let's try the *simplest* random model.

Random k-SAT

- Aim: *contrive* hard (but satisfiable) forumlas.
- Let's try the *simplest* random model.

Uniformly random k-SAT

- n variables $x_1, \ldots, x_n$.
- $F_k(n, m)$ has m *random* clauses.
- Let $m = r \cdot n$ with $r = \Theta(1)$.
- "With high probability" $=$ with probability $1 - o(1)$ as $n \to \infty$.

Random k-SAT

- Aim: *contrive* hard (but satisfiable) forumlas.
- Let's try the *simplest* random model.

The statistical physics perspective

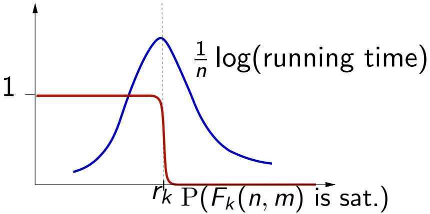
- *Spin glasses*.
- Gibbs measure at zero temperature.
- Rigorous vs. non-rigorous methods.

The k-SAT threshold

Theorem (Friedgut 1999)

For each k there is a *threshold* $r_k = r_k(n)$ so that w.h.p.

- $F_k(n, m)$ is *satisfiable* if $r < r_k - \epsilon$,
- $F_k(n, m)$ is *unsatisfiable* if $r > r_k + \epsilon$.



Running time of "worst-case" algorithms is *exponential* and *peaks at* r_k.

The k-SAT threshold

Theorem (Friedgut 1999)

For each k there is a *threshold* $r_k = r_k(n)$ so that w.h.p.

- $F_k(n, m)$ is *satisfiable* if $r < r_k - \epsilon$,
- $F_k(n, m)$ is *unsatisfiable* if $r > r_k + \epsilon$.

Theorem (Achlioptas, Peres 2004)

$r_k \sim 2^k \ln 2$.

Proof

2nd moment method (non-algorithmic).

Algorithms for random k-SAT

Question

- The *threshold* is $r_k \sim 2^k \ln 2$.
- For what m/n can *compute* satisfying assignments *efficiently*?

Algorithms for random k-SAT

Question

- The *threshold* is $r_k \sim 2^k \ln 2$.
- For what m/n can *compute* satisfying assignments *efficiently*?

Algorithm	*Density* $m/n < \cdots$	
Pure Literal	$o(1)$ as $k \to \infty$	Kim 2006
Walksat, rigorous	$\frac{1}{6} \cdot 2^k/k^2$	CFFKV 2009
Walksat, non-rigorous	$2^k/k$	Monasson 2003
Shortest Clause	$\frac{e^2}{8} \cdot 2^k/k$	Chvatal, Reed 1992
Unit Clause	$\frac{e}{2} \cdot 2^k/k$	Chao, Franco 1990
SC+backtracking	$1.817 \cdot 2^k/k$	Frieze, Suen 1996
BP+decimation (non-rigorous)	$e \cdot 2^k/k$	Montanari 2007

Algorithms for random k-SAT

Question

- The *threshold* is $r_k \sim 2^k \ln 2$.
- For what m/n can *compute* satisfying assignments *efficiently*?

In summary,

...*efficient algorithms* are known to succeed up to $m/n = c \cdot 2^k/k$.

Algorithms for random k-SAT

Question

- The *threshold* is $r_k \sim 2^k \ln 2$.
- For what m/n can *compute* satisfying assignments *efficiently*?

In summary,

...*efficient algorithms* are known to succeed up to $m/n = c \cdot 2^k/k$,

Problem (Chvatal, Reed 1992)

Devise an algorithm that succeeds up to $m/n = 2^k \omega(k)/k$, $\omega(k) \to \infty$.

Frozen variables

Replica symmetry breaking

- The *k-SAT threshold* is $r \sim 2^k \ln 2$.
- But there occurs another *phase transition* at $r \sim 2^k \ln k/k$...
- ...that affects the computational difficulty.

Frozen variables

Replica symmetry breaking

- The *k-SAT threshold* is $r \sim 2^k \ln 2$.
- But there occurs another *phase transition* at $r \sim 2^k \ln k/k \ldots$
- ...that affects the computational difficulty.

Loose vs. frozen variables

Let Φ be a k-CNF, σ a *satisfying assignment*, and x a variable.

- x is loose if there is a satisfying assignment τ such that

$$\sigma(x) \neq \tau(x) \text{ and } \mathrm{dist}(\sigma, \tau) \leq \ln(n).$$

Frozen variables

Replica symmetry breaking

- The *k-SAT threshold* is $r \sim 2^k \ln 2$.
- But there occurs another *phase transition* at $r \sim 2^k \ln k/k \ldots$
- ...that affects the computational difficulty.

Loose vs. frozen variables

Let Φ be a k-CNF, σ a *satisfying assignment*, and x a variable.

- x is loose if there is a satisfying assignment τ such that

$$\sigma(x) \neq \tau(x) \text{ and } \mathrm{dist}(\sigma, \tau) \leq \ln(n).$$

- x is frozen if for any satisfying assignment τ

$$\sigma(x) \neq \tau(x) \Rightarrow \mathrm{dist}(\sigma, \tau) = \Omega(n).$$

Frozen variables (ctd.)

Question

Why are things so much "harder" for $r > 2^k \ln k/k$?

Theorem (Achlioptas, ACO 2008)

For a *random* satisfying assignment of $F_k(n, m)$:

1. if $r < (1 - \varepsilon_k)2^k \ln k/k$, then almost all variables are *loose* w.h.p.
2. if $r > (1 + \varepsilon_k)2^k \ln k/k$, then almost all variables are *frozen* w.h.p.

Frozen variables (ctd.)

Question

Why are things so much "harder" for $r > 2^k \ln k/k$?

Theorem (Achlioptas, ACO 2008)

For a *random* satisfying assignment of $F_k(n, m)$:

1. if $r < (1 - \varepsilon_k)2^k \ln k/k$, then almost all variables are *loose* w.h.p.
2. if $r > (1 + \varepsilon_k)2^k \ln k/k$, then almost all variables are *frozen* w.h.p.

In other words. . .

- first correlations between variables are purely *local*.
- but then *long-range correlations* occur.

A new algorithm

But if RSB occurs at $r \sim 2^k \ln(k)/k$...

- ...*local search* algorithms ought to succeed up to that density.
- Yet none has been known to succeed beyond $\mathrm{const} \times 2^k/k$.

A new algorithm

But if RSB occurs at $r \sim 2^k \ln(k)/k \ldots$

- ...*local search* algorithms ought to succeed up to that density.
- Yet none has been known to succeed beyond $\mathrm{const} \times 2^k/k$.

Theorem (ACO 2009)

$\texttt{Fix}(F_k(n, m))$ succeeds up to $r = (1 - \varepsilon_k)2^k \ln k/k$.

A new algorithm

But if RSB occurs at $r \sim 2^k \ln(k)/k$...

- ... *local search* algorithms ought to succeed up to that density.
- Yet none has been known to succeed beyond $\mathrm{const} \times 2^k/k$.

Theorem (ACO 2009)

$\texttt{Fix}(F_k(n,m))$ succeeds up to $r = (1-\varepsilon_k)2^k \ln k/k$.

The algorithm `Fix`

1. Start with the *all-true* assignment.
2. For any *all-negative* clause
3. flip one of its variables w/out generating *new* unsat clauses (if possible).
4. *Clean-up step:* satisfy the remaining unsat clauses.

Analyzing `Fix`

The algorithm `Fix`

1. Start with the *all-true* assignment.
2. For any *all-negative* clause
3. flip one of its variables w/out generating *new* unsat clauses (if possible).
4. Compute a set Z' of variables such that *any clause*
   - either is satisfied by a variable in $V \setminus Z'$,
   - or contains *at least three* variables from Z'.

   Find a matching from the *unsat* clauses to Z'.

Analyzing `Fix`

The algorithm `Fix`

1. Start with the *all-true* assignment.
2. For any *all-negative* clause
3. flip one of its variables w/out generating *new* unsat clauses (if possible).
4. Compute a set Z' of variables such that *any clause*
   - either is satisfied by a variable in $V \setminus Z'$,
   - or contains *at least three* variables from Z'.

   Find a matching from the *unsat* clauses to Z'.

- Let $\varepsilon > 0$ and suppose $k > k_0(\varepsilon)$.
- Let $\Phi = F_k(n, m)$ with $m/n = (1 - \varepsilon) \cdot 2^k \ln(k)/k$.
- There are $2^{-k} m = n \cdot (1 - \varepsilon) \ln(k)/k$ *all-negative* clauses.

Analyzing `Fix`

The algorithm `Fix`

1. Start with the *all-true* assignment.
2. For any *all-negative* clause
3. flip one of its variables w/out generating *new* unsat clauses (if possible).
4. Compute a set Z' of variables such that *any clause*
   - either is satisfied by a variable in $V \setminus Z'$,
   - or contains *at least three* variables from Z'.

   Find a matching from the *unsat* clauses to Z'.

Key Lemma

W.h.p. #unsat clauses after Steps 1–3 is $\leq n \exp(-k^{\varepsilon})$.

Remember: initially there were $n \cdot (1-\varepsilon)\ln(k)/k$ of them.

Analyzing Fix (ctd.)

Fixing the first unsat clause

- Consider the *first* all-neg clause, say, $\bar{x}_1 \vee \cdots \vee \bar{x}_k$.

Analyzing Fix (ctd.)

Fixing the first unsat clause

- Consider the *first* all-neg clause, say, $\bar{x}_1 \vee \cdots \vee \bar{x}_k$.
- Need to flip one x_i to *false*.

Analyzing `Fix` (ctd.)

Fixing the first unsat clause

- Consider the *first* all-neg clause, say, $\bar{x}_1 \vee \cdots \vee \bar{x}_k$.
- Need to flip one x_i to *false*.
- This generates a *new* unsat clause iff Φ contains

$$x_i \vee \bar{y}_2 \vee \cdots \vee \bar{y}_k.$$

Analyzing `Fix` (ctd.)

Fixing the first unsat clause

- Consider the *first* all-neg clause, say, $\bar{x}_1 \vee \cdots \vee \bar{x}_k$.
- Need to flip one x_i to *false*.
- This generates a *new* unsat clause iff Φ contains

$$x_i \vee \bar{y}_2 \vee \cdots \vee \bar{y}_k.$$

- *Total number* of uniquely pos clauses is

$$k2^{-k} \cdot m = (1-\varepsilon)\ln k \cdot n.$$

Analyzing `Fix` (ctd.)

Fixing the first unsat clause

- Consider the *first* all-neg clause, say, $\bar{x}_1 \vee \cdots \vee \bar{x}_k$.
- Need to flip one x_i to *false*.
- This generates a *new* unsat clause iff Φ contains

$$x_i \vee \bar{y}_2 \vee \cdots \vee \bar{y}_k.$$

- *Total number* of uniquely pos clauses is

$$k2^{-k} \cdot m = (1-\varepsilon) \ln k \cdot n.$$

- $\Rightarrow$ for each x_i we *expect* $(1-\varepsilon) \ln k$.

Fixing the first unsat clause

- Consider the *first* all-neg clause, say, $\bar{x}_1 \vee \cdots \vee \bar{x}_k$.
- Need to flip one x_i to *false*.
- This generates a *new* unsat clause iff Φ contains

$$x_i \vee \bar{y}_2 \vee \cdots \vee \bar{y}_k.$$

- *Total number* of uniquely pos clauses is

$$k2^{-k} \cdot m = (1-\varepsilon)\ln k \cdot n.$$

- $\Rightarrow$ for each x_i we *expect* $(1-\varepsilon)\ln k$.
- In fact, for each x_i the number is $\sim \mathrm{Po}((1-\varepsilon)\ln k)$.

Analyzing `Fix` (ctd.)

Fixing the first unsat clause (ctd.)

- Consider the *first* all-neg clause, say, $\bar{x}_1 \vee \cdots \vee \bar{x}_k$.
- Need to flip one x_i to *false*.
- For each x_i the number of

$$x_i \vee \bar{y}_2 \vee \cdots \vee \bar{y}_k.$$

is $\mathrm{Po}((1-\varepsilon)\ln k)$.

Analyzing Fix (ctd.)

Fixing the first unsat clause (ctd.)

- Consider the *first* all-neg clause, say, $\bar{x}_1 \vee \cdots \vee \bar{x}_k$.
- Need to flip one x_i to *false*.
- For each x_i the number of

$$x_i \vee \bar{y}_2 \vee \cdots \vee \bar{y}_k.$$

  is $\mathrm{Po}((1-\varepsilon)\ln k)$.
- $\Rightarrow$ we are *free to flip* x_i with probability

$$\exp(-(1-\varepsilon)\ln k) = k^{\varepsilon-1}.$$

Analyzing Fix (ctd.)

Fixing the first unsat clause (ctd.)

- Consider the *first* all-neg clause, say, $\bar{x}_1 \vee \cdots \vee \bar{x}_k$.
- Need to flip one x_i to *false*.
- For each x_i the number of

$$x_i \vee \bar{y}_2 \vee \cdots \vee \bar{y}_k.$$

  is $\mathrm{Po}((1-\varepsilon)\ln k)$.
- $\Rightarrow$ we are *free to flip* x_i with probability

$$\exp(-(1-\varepsilon)\ln k) = k^{\varepsilon-1}.$$

- $\Rightarrow$ the *expected* number of free x_is is $k \cdot k^{\varepsilon-1} = k^{\varepsilon}$.

Analyzing Fix (ctd.)

Fixing the first unsat clause (ctd.)

- Consider the *first* all-neg clause, say, $\bar{x}_1 \vee \cdots \vee \bar{x}_k$.
- Need to flip one x_i to *false*.
- For each x_i the number of

$$x_i \vee \bar{y}_2 \vee \cdots \vee \bar{y}_k.$$

  is $\mathrm{Po}((1-\varepsilon)\ln k)$.
- $\Rightarrow$ we are *free to flip* x_i with probability

$$\exp(-(1-\varepsilon)\ln k) = k^{\varepsilon-1}.$$

- $\Rightarrow$ the *expected* number of free x_is is $k \cdot k^{\varepsilon-1} = k^{\varepsilon}$.
- $\Rightarrow$ we can *fix* the clause with prob $1-\exp(-k^{\varepsilon})$.

Analyzing `Fix` (ctd.)

How to proceed

This calculation only applies to the *first* all-neg clause.

To proceed, we need to take into account:

- setting vars to *false* may *increase* the # of uniquely pos clauses. . .

Analyzing `Fix` (ctd.)

How to proceed

This calculation only applies to the *first* all-neg clause.

To proceed, we need to take into account:

- setting vars to *false* may *increase* the # of uniquely pos clauses. . .
- . . . or actually *decrease* it.

Analyzing `Fix` (ctd.)

How to proceed

This calculation only applies to the *first* all-neg clause.

To proceed, we need to take into account:

- setting vars to *false* may *increase* the # of uniquely pos clauses. . .
- . . . or actually *decrease* it.
- Setting a var to false may fix *several* all-neg clauses.

Analyzing Fix (ctd.)

How to proceed

This calculation only applies to the *first* all-neg clause.

To proceed, we need to take into account:

- setting vars to *false* may *increase* the # of uniquely pos clauses. . .
- . . . or actually *decrease* it.
- Setting a var to false may fix *several* all-neg clauses.

Method of deferred decisions

- *Only* reveal the information needed for the *next step*,
- so that everything else remains *random*.

Analyzing Fix (ctd.)

Let Φ_i =(random) clause i; Φ_{ij} = jth literal in Φ_i.

A 'card game'

- Track Steps 1–3 by *maps* $\pi_t : [m] \times [k] \to \{-1, 1\} \cup \{\text{literals}\}$.

Analyzing Fix (ctd.)

Let Φ_i =(random) clause i; $\Phi_{ij} = j$th literal in Φ_i.

A 'card game'

- Track Steps 1–3 by *maps* $\pi_t : [m] \times [k] \to \{-1, 1\} \cup \{\text{literals}\}$.
- $\pi_t(i, j) \in \{+1, -1\} \Rightarrow$ we've only revealed the sign of Φ_{ij}.

Analyzing `Fix` (ctd.)

Let Φ_i =(random) clause i; $\Phi_{ij} = j$th literal in Φ_i.

A 'card game'

- Track Steps 1–3 by *maps* $\pi_t : [m] \times [k] \to \{-1, 1\} \cup \{\text{literals}\}$.
- $\pi_t(i, j) \in \{+1, -1\} \Rightarrow$ we've only revealed the sign of Φ_{ij}.
- $\pi_t(i, j) \in \{\text{literals}\} \Rightarrow$ we've revealed the actual literal Φ_{ij}.

Analyzing Fix (ctd.)

Let Φ_i =(random) clause i; $\Phi_{ij} = j$th literal in Φ_i.

A 'card game'

- Track Steps 1–3 by *maps* $\pi_t : [m] \times [k] \to \{-1, 1\} \cup \{\text{literals}\}$.
- $\pi_t(i, j) \in \{+1, -1\} \Rightarrow$ we've only revealed the sign of Φ_{ij}.
- $\pi_t(i, j) \in \{\text{literals}\} \Rightarrow$ we've revealed the actual literal Φ_{ij}.
- The set Z of *vars set to flase* corresponds to a *sequence* $(Z_t)_t$.

Analyzing Fix (ctd.)

Let Φ_i =(random) clause i; $\Phi_{ij} = j$th literal in Φ_i.

A 'card game'

- Track Steps 1–3 by *maps* $\pi_t : [m] \times [k] \to \{-1, 1\} \cup \{\text{literals}\}$.
- $\pi_t(i, j) \in \{+1, -1\} \Rightarrow$ we've only revealed the sign of Φ_{ij}.
- $\pi_t(i, j) \in \{\text{literals}\} \Rightarrow$ we've revealed the actual literal Φ_{ij}.
- The set Z of *vars set to flase* corresponds to a *sequence* $(Z_t)_t$.
- U_t = crtitical clauses.

Analyzing `Fix` (ctd.)

Let Φ_i =(random) clause i; $\Phi_{ij} = j$th literal in Φ_i.

A 'card game'

- Track Steps 1–3 by *maps* $\pi_t : [m] \times [k] \to \{-1, 1\} \cup \{\text{literals}\}$.
- $\pi_t(i, j) \in \{+1, -1\} \Rightarrow$ we've only revealed the sign of Φ_{ij}.
- $\pi_t(i, j) \in \{\text{literals}\} \Rightarrow$ we've revealed the actual literal Φ_{ij}.
- The set Z of *vars set to flase* corresponds to a *sequence* $(Z_t)_t$.
- U_t = crtitical clauses.
- and $U_t(x) = \#$ critical clauses *'supported'* by x.

Analyzing `Fix` (ctd.)

Initialization: what is π_0?

- Let $\pi_0(i,j) =$*sign of* Φ_{ij} ...
- unless Φ_{ij} is the *only* positive literal in $\Phi_i \rightsquigarrow \pi_0(i,j) = \Phi_{ij}$.
- Let $Z_0 = \emptyset$.
- Let $U_0 =$all clauses with *exactly* one pos literal.

Analyzing Fix (ctd.)

Defining π_t for $t \geq 1$

PI1
- Let $\phi_t = \min_{i \in [m]} \{\Phi_i$ is *all-negative* w/out var. from $Z_{t-1}\}$.
- no such $i \Rightarrow$ stop.

PI2
- Let $j = \min_{l \leq k} \{U_{t-1}(|\Phi_{\phi_t j}|) = 0\}$.
- no such $l \Rightarrow$ let $j = 1$.
- Let $Z_t = Z_{t-1} \cup \{\Phi_{\phi_t j}\}$.

PI3
- $U_t = \{i : \Phi_i$ has *ex. one* pos lit $\notin Z_t$ and no neg lit $\in \bar{Z}_t\}$.
- $U_t(x) =$those where x is the unique pos literal.

PI4

$$\pi_t(i,j) = \begin{cases} \Phi_{ij} & \text{if } i = \phi_t \vee |\Phi_{ij}| \in Z_t \vee (i \in U_t \wedge \pi_0(i,j) = 1), \\ \pi_{t-1}(i,j) & \text{otherwise.} \end{cases}$$

Analyzing Fix (ctd.)

$$\begin{array}{ccccccccc} - & - & - & + & + & + & + & + & + \\ - & - & - & - & - & - & + & - & + \\ - & - & - & - & - & - & - & - & + \\ - & - & - & - & - & - & - & + & - \\ - & - & - & - & - & - & - & - & - \end{array}$$

The card game: example

- The initial *sign pattern* ($k = 5$).
- Φ_1, Φ_2, Φ_3 are all-negative, the next three clauses
- Φ_4, Φ_5, Φ_6 have exactly one positive literal, etc.
- The variables underlying the $\pm$s are still *uniformly random*.

Analyzing Fix (ctd.)

$$\pi_0 = \begin{array}{ccccccccc} - & - & - & {\color{red}x_5} & {\color{red}x_2} & {\color{red}x_3} & + & + & + \\ - & - & - & - & - & - & + & - & + \\ - & - & - & - & - & - & - & - & + \\ - & - & - & - & - & - & - & + & - \\ - & - & - & - & - & - & - & - & - \end{array}$$

The card game: example

- The *supporting* variables revealed.
- $U_0(x_2) = U_0(x_3) = U_0(x_5) = 1$.
- $U_0 = \{4, 5, 6\}$.

Analyzing Fix (ctd.)

$\bar{x}_2$	$-$	$-$	x_5	x_2	x_3	$+$	$+$	$+$
$\bar{x}_3$	$-$	$-$	$-$	$-$	$-$	$+$	$-$	$+$
$\bar{x}_1$	$-$	$-$	$-$	$-$	$-$	$-$	$-$	$+$
$-$	$-$	$-$	$-$	$-$	$-$	$-$	$+$	$-$
$-$	$-$	$-$	$-$	$-$	$-$	$-$	$-$	$-$

The card game: example

- Reveal the *first* all-negative clause.

Analyzing `Fix` (ctd.)

$\bar{x}_2$	$-$	$\bar{x}_1$	x_5	x_2	x_3	$+$	$+$	$+$
$\bar{x}_3$	$-$	$-$	$-$	$-$	$-$	$+$	$-$	$+$
$\bar{x}_1$	$-$	$-$	$-$	$-$	$-$	$-$	$-$	x_1
$-$	$-$	$-$	$-$	$-$	$-$	$-$	x_1	$-$
$-$	$-$	$-$	$\bar{x}_1$	$-$	$-$	$-$	$-$	$-$

The card game: example

- $U_0(x_2) = U_0(x_3) = 1$ but $U_0(x_1) = 0$.
- Thus, $Z_1 = \{x_1\}$.
- Reveal all occurrences of x_1.

Analyzing Fix (ctd.)

$$\pi_1 = \begin{array}{ccccccccc} \bar{x}_2 & - & \bar{x}_1 & x_5 & x_2 & x_3 & + & \color{red}{x_4} & + \\ \bar{x}_3 & - & - & - & - & - & + & - & + \\ \bar{x}_1 & - & - & - & - & - & - & - & x_1 \\ - & - & - & - & - & - & - & x_1 & - \\ - & - & - & \bar{x}_1 & - & - & - & - & - \end{array}$$

The card game: example

- There is one new 'critical' clause, namely Φ_8.
- Reveal its *supporting variable* x_4.
- Φ_4 contains $\bar{x}_1 \Rightarrow$ not critical anymore.
- At this point the vars underlying the $\pm$ are *uniform* over $V \setminus \{x_1\}$.

Analyzing Fix (ctd.)

$$\begin{array}{ccccccccc}
\bar{x}_2 & \color{red}{\bar{x}_5} & \bar{x}_1 & x_5 & x_2 & x_3 & + & x_4 & + \\
\bar{x}_3 & - & - & - & - & - & + & - & + \\
\bar{x}_1 & - & - & - & - & - & - & - & x_1 \\
- & - & - & - & - & - & - & x_1 & - \\
- & - & - & \bar{x}_1 & - & - & - & - & -
\end{array}$$

The card game: example

- Reveal the next all-minus clause.
- Flip x_5 as it does not support any clauses, i.e., $Z_2 = \{x_1, x_5\}$.

Analyzing Fix (ctd.)

$\bar{x}_2$	$\bar{x}_5$	$\bar{x}_1$	x_5	x_2	x_3	$+$	x_4	$+$
$\bar{x}_3$	$-$	$-$	$-$	$-$	$-$	$+$	$-$	x_5
$\bar{x}_1$	$-$	$-$	$-$	$-$	$-$	$-$	$-$	x_1
$-$	$-$	$-$	$-$	$-$	$-$	$-$	x_1	$-$
$\bar{x}_5$	$-$	$-$	$\bar{x}_1$	$-$	$-$	$-$	$-$	$-$

The card game: example

- Reveal all occurrences of x_5.

Analyzing Fix (ctd.)

$$\pi_2 = \begin{array}{ccccccccc} \bar{x}_2 & \bar{x}_5 & \bar{x}_1 & x_5 & x_2 & x_3 & + & x_4 & \color{red}{x_4} \\ \bar{x}_3 & - & - & - & - & - & + & - & x_5 \\ \bar{x}_1 & - & - & - & - & - & - & - & x_1 \\ - & - & - & - & - & - & - & x_1 & - \\ \bar{x}_5 & - & - & \bar{x}_1 & - & - & - & - & - \end{array}$$

The card game: example

- x_5 occurs in the last clause, which becomes critical.
- Thus, we have to reveal the var underlying the $+$.
- At this point the vars underlying the $\pm$ are *uniform* over $V \setminus \{x_1, x_5\}$.
- No all-minus columns left $\Rightarrow$ halt.

Analyzing Fix (ctd.)

Let T =stopping time.

Lemma

For all $t \le \min\{T, n\}$ there are $\le 2^{1-k} m \exp(-kt/n)$ *all-minus columns* w.h.p.

Analyzing Fix (ctd.)

Let T =stopping time.

Lemma

For all $t \leq \min\{T, n\}$ there are $\leq 2^{1-k} m \exp(-kt/n)$ *all-minus columns* w.h.p.

Proof

- For any $1 \leq i \leq m$ we have $\mathrm{P}\left[\pi_0(i,\cdot) = \text{all-minus}\right] = 2^{-k}$.

Analyzing Fix (ctd.)

Let T =stopping time.

Lemma

For all $t \leq \min\{T, n\}$ there are $\leq 2^{1-k} m \exp(-kt/n)$ *all-minus columns* w.h.p.

Proof

- For any $1 \leq i \leq m$ we have $\mathrm{P}\left[\pi_0(i, \cdot) = \text{all-minus}\right] = 2^{-k}$.
- At each time s we flip a variable $z_s \in V \setminus Z_{s-1}$.

Analyzing Fix (ctd.)

Let T =stopping time.

Lemma

For all $t \leq \min\{T, n\}$ there are $\leq 2^{1-k} m \exp(-kt/n)$ *all-minus columns* w.h.p.

Proof

- For any $1 \leq i \leq m$ we have $\mathrm{P}[\pi_0(i, \cdot) = \text{all-minus}] = 2^{-k}$.
- At each time s we flip a variable $z_s \in V \setminus Z_{s-1}$.
- If $\pi_{s-1}(i, j) = -1$, then $\Phi_{ij} \in V \setminus Z_{s-1}$ is uniformly distributed.

Analyzing `Fix` (ctd.)

Let T =stopping time.

Lemma

For all $t \leq \min\{T, n\}$ there are $\leq 2^{1-k} m \exp(-kt/n)$ *all-minus columns* w.h.p.

Proof

- For any $1 \leq i \leq m$ we have $\mathrm{P}\left[\pi_0(i,\cdot) = \text{all-minus}\right] = 2^{-k}$.
- At each time s we flip a variable $z_s \in V \setminus Z_{s-1}$.
- If $\pi_{s-1}(i,j) = -1$, then $\Phi_{ij} \in V \setminus Z_{s-1}$ is uniformly distributed.
- Hence, $\mathrm{P}\left[|\Phi_{ij}| = z_s | \mathcal{F}_{s-1}\right] \geq 1/(n - s + 1)$.

Analyzing Fix (ctd.)

Let T =stopping time.

Lemma

For all $t \leq \min\{T, n\}$ there are $\leq 2^{1-k} m \exp(-kt/n)$ *all-minus columns* w.h.p.

Proof

- For any $1 \leq i \leq m$ we have $\mathrm{P}\left[\pi_0(i,\cdot) = \text{all-minus}\right] = 2^{-k}$.
- At each time s we flip a variable $z_s \in V \setminus Z_{s-1}$.
- If $\pi_{s-1}(i,j) = -1$, then $\Phi_{ij} \in V \setminus Z_{s-1}$ is uniformly distributed.
- Hence, $\mathrm{P}\left[|\Phi_{ij}| = z_s | \mathcal{F}_{s-1}\right] \geq 1/(n-s+1)$.
- Consequently,

$$\mathrm{P}\left[\pi_t(i,\cdot) = \text{all-minus}\right] \leq 2^{-k} \prod_{s \leq t} 1 - \frac{1}{n-s+1} \leq 2^{-k} \exp(-kt/n).$$

Analyzing `Fix` (ctd.)

Let T =stopping time.

Lemma

For all $t \leq \min\{T, n\}$ there are $\leq 2^{1-k} m \exp(-kt/n)$ *all-minus columns* w.h.p.

Analyzing `Fix` (ctd.)

Let T =stopping time.

Lemma

For all $t \leq \min\{T, n\}$ there are $\leq 2^{1-k} m \exp(-kt/n)$ *all-minus columns* w.h.p.

Corollary

$T \leq 4n \ln \ln k/k$ w.h.p.

Analyzing `Fix` (ctd.)

Lemma

For all $t \leq T$ we have $|U_t| \leq (1 - \varepsilon/2)\ln(k)/n$ w.h.p.

Analyzing Fix (ctd.)

Lemma

For all $t \leq T$ we have $|U_t| \leq (1-\varepsilon/2)\ln(k)/n$ w.h.p.

Wrapping up: phase 1

- We know $|Z_T| = T \leq 4n \ln\ln k/k$ w.h.p.

Analyzing `Fix` (ctd.)

Lemma

For all $t \leq T$ we have $|U_t| \leq (1-\varepsilon/2)\ln(k)/n$ w.h.p.

Wrapping up: phase 1

- We know $|Z_T| = T \leq 4n \ln\ln k/k$ w.h.p.
- W.h.p. $|U_t| \leq (1-\varepsilon/2)\ln(k)/n$ for all $t \leq T$.

Analyzing `Fix` (ctd.)

Lemma

For all $t \leq T$ we have $|U_t| \leq (1-\varepsilon/2)\ln(k)/n$ w.h.p.

Wrapping up: phase 1

- We know $|Z_T| = T \leq 4n \ln\ln k/k$ w.h.p.
- W.h.p. $|U_t| \leq (1-\varepsilon/2)\ln(k)/n$ for all $t \leq T$.
- Thus, w.h.p. there are $\geq nk^{\varepsilon/2-1}$ vars x with $U_t(x) = 0$ for all $t \leq T$.

Analyzing `Fix` (ctd.)

Lemma

For all $t \leq T$ we have $|U_t| \leq (1-\varepsilon/2)\ln(k)/n$ w.h.p.

Wrapping up: phase 1

- We know $|Z_T| = T \leq 4n\ln\ln k/k$ w.h.p.
- W.h.p. $|U_t| \leq (1-\varepsilon/2)\ln(k)/n$ for all $t \leq T$.
- Thus, w.h.p. there are $\geq nk^{\varepsilon/2-1}$ vars x with $U_t(x) = 0$ for all $t \leq T$.
- Therefore, the argument used for the *first clause* (i.e., $t = 1$)...
- ...actually applies for all t.

The factor graph

- $\Phi = k$-CNF formula.
- The *factor graph* is a bipartite auxiliary graph.
- Its vertices are the *variables* and the *clauses* of Φ.
- Each clause is adjacent to the variables that occur in it.

The factor graph

- $\Phi = k$-CNF formula.
- The *factor graph* is a bipartite auxiliary graph.
- Its vertices are the *variables* and the *clauses* of Φ.
- Each clause is adjacent to the variables that occur in it.

Unit Clause and the factor graph

To *set* the next variable, the algorithm inspects

- ...for each *clause* the variables that it contains,
- ...and their assigned *values*.

The factor graph

- $\Phi = k$-CNF formula.
- The *factor graph* is a bipartite auxiliary graph.
- Its vertices are the *variables* and the *clauses* of Φ.
- Each clause is adjacent to the variables that occur in it.

Unit Clause and the factor graph

To *set* the next variable, the algorithm inspects

- ...for each *clause* the variables that it contains,
- ...and their assigned *values*.

Thus, it inspects the *factor graph* up to depth one.

The factor graph (ctd.)

- $\Phi = k$-CNF formula.
- The *factor graph* is a bipartite auxiliary graph.
- Its vertices are the *variables* and the *clauses* of Φ.
- Each clause is adjacent to the variables that occur in it.

The factor graph (ctd.)

- $\Phi = k$-CNF formula.
- The *factor graph* is a bipartite auxiliary graph.
- Its vertices are the *variables* and the *clauses* of Φ.
- Each clause is adjacent to the variables that occur in it.

`Fix`: depth three

In the first phase the algorithm

- ...inspects each *clauses* and its *variables*,
- ...the *clauses* in which these vars occur,
- ...and the values of all *other* variables in those clauses.

The factor graph (ctd.)

Belief Propagation: a depth ω algorithm.

- Check out the *ω-neighborhood* of each variable.

The factor graph (ctd.)

Belief Propagation: a depth ω algorithm.

- Check out the *ω-neighborhood* of each variable.
- Most likely, this is a *tree*.

The factor graph (ctd.)

Belief Propagation: a depth ω algorithm.

- Check out the *ω-neighborhood* of each variable.
- Most likely, this is a *tree*.
- Compute the *marginals* at the *root* variable,
- . . . given all previous decisions.

The factor graph (ctd.)

Belief Propagation: a depth ω algorithm.

- Check out the *ω-neighborhood* of each variable.
- Most likely, this is a *tree*.
- Compute the *marginals* at the *root* variable,
- . . . given all previous decisions.
- Decimation: assign a variable based on these marginals.

The factor graph (ctd.)

Belief Propagation: a depth ω algorithm.

- Check out the *ω-neighborhood* of each variable.
- Most likely, this is a *tree*.
- Compute the *marginals* at the *root* variable,
- ... given all previous decisions.
- Decimation: assign a variable based on these marginals.

Surprise

To reach the dRSB point, *depth three* is sufficient.

Conclusion

- `Fix` works up to the dRSB point, at least asymptotically for *large* k.
- Is $F_k(n, m)$ *'hard'* beyond the dRSB point?
- Better algorithm for *small* k?