Satisfiability and the Giant Component in Online Variants of the Classical Random Models

David Kravitz

kravitz@cmu.edu

Department of Mathematical Sciences

Carnegie Mellon University

General Ground Rules

General Ground Rules

- Vertex or variable set is $[n] = \{1, 2, \ldots, n\}$.

General Ground Rules

- Vertex or variable set is $[n] = \{1, 2, \ldots, n\}$.
- We look at what happens when $n \to \infty$.

General Ground Rules

- Vertex or variable set is $[n] = \{1, 2, \ldots, n\}$.
- We look at what happens when $n \to \infty$.
- Something happens *with high probability*, or **whp**, if it happens with probability $1 - o_n(1)$.

General Ground Rules

- Vertex or variable set is $[n] = \{1, 2, \ldots, n\}$.
- We look at what happens when $n \to \infty$.
- Something happens *with high probability*, or **whp**, if it happens with probability $1 - o_n(1)$.
- For any $k \geq 1$, a random k-set is chosen uniformly at random from $\binom{[n]}{k}$.

General Ground Rules

- Vertex or variable set is $[n] = \{1, 2, \ldots, n\}$.
- We look at what happens when $n \to \infty$.
- Something happens *with high probability*, or **whp**, if it happens with probability $1 - o_n(1)$.
- For any $k \geq 1$, a random k-set is chosen uniformly at random from $\binom{[n]}{k}$.
- Random edges in graphs are k-sets when $k = 2$.

General Ground Rules

- Vertex or variable set is $[n] = \{1, 2, \ldots, n\}$.
- We look at what happens when $n \to \infty$.
- Something happens *with high probability*, or **whp**, if it happens with probability $1 - o_n(1)$.
- For any $k \geq 1$, a random k-set is chosen uniformly at random from $\binom{[n]}{k}$.
- Random edges in graphs are k-sets when $k = 2$.
- Duplications within k-sets won't change any of our results so we ignore them.

A famous result

A famous result

Let $G_{n,cn}$ be a random graph with n vertices and cn random edges for some constant c.

A famous result

Let $G_{n,cn}$ be a random graph with n vertices and cn random edges for some constant c.

Theorem: (Erdős, Rényi, 1960)
Let C_1 be the size of the largest component of $G_{n,cn}$.

A famous result

Let $G_{n,cn}$ be a random graph with n vertices and cn random edges for some constant c.

Theorem: (Erdős, Rényi, 1960)
Let C_1 be the size of the largest component of $G_{n,cn}$.

- If $c < \frac{1}{2}$ then **whp** $C_1 = O(\log n)$.

A famous result

Let $G_{n,cn}$ be a random graph with n vertices and cn random edges for some constant c.

Theorem: (Erdős, Rényi, 1960)
Let C_1 be the size of the largest component of $G_{n,cn}$.

- If $c < \frac{1}{2}$ then **whp** $C_1 = O(\log n)$.
- If $c > \frac{1}{2}$ then **whp** $C_1 = \Omega(n)$.

A famous result

Let $G_{n,cn}$ be a random graph with n vertices and cn random edges for some constant c.

Theorem: (Erdős, Rényi, 1960)
Let C_1 be the size of the largest component of $G_{n,cn}$.

- If $c < \frac{1}{2}$ then **whp** $C_1 = O(\log n)$.
- If $c > \frac{1}{2}$ then **whp** $C_1 = \Omega(n)$.
- Furthermore, if $c > \frac{1}{2}$ then **whp** $C_2 = O(\log n)$.

A famous result

Let $G_{n,cn}$ be a random graph with n vertices and cn random edges for some constant c.

Theorem: (Erdős, Rényi, 1960)
Let C_1 be the size of the largest component of $G_{n,cn}$.

- If $c < \frac{1}{2}$ then **whp** $C_1 = O(\log n)$.
- If $c > \frac{1}{2}$ then **whp** $C_1 = \Omega(n)$.
- Furthermore, if $c > \frac{1}{2}$ then **whp** $C_2 = O(\log n)$.

The jump at $c = \frac{1}{2}$ is called a phase transition .

A famous result

Let $G_{n,cn}$ be a random graph with n vertices and cn random edges for some constant c.

Theorem: (Erdős, Rényi, 1960)
Let C_1 be the size of the largest component of $G_{n,cn}$.

- If $c < \frac{1}{2}$ then **whp** $C_1 = O(\log n)$.
- If $c > \frac{1}{2}$ then **whp** $C_1 = \Omega(n)$.
- Furthermore, if $c > \frac{1}{2}$ then **whp** $C_2 = O(\log n)$.

The jump at $c = \frac{1}{2}$ is called a phase transition .

A component of size $\Omega(n)$ is called a giant component .

Susceptibility

Susceptibility

Let G be a graph with connected components $C_1, \ldots, C_r$.

Susceptibility

Let G be a graph with connected components $C_1, \ldots, C_r$. We define the susceptibility of G to be

$$X(G) \quad = \quad \frac{1}{n} \sum_{i=1}^{r} |C_i|^2.$$

Susceptibility

Let G be a graph with connected components $C_1, \ldots, C_r$.
We define the susceptibility of G to be

$$X(G) \quad = \quad \frac{1}{n} \sum_{i=1}^{r} |C_i|^2.$$

Note: $X(G)$ is the expected component size of a vertex chosen uniformly at random.

Susceptibility

Let G be a graph with connected components $C_1, \ldots, C_r$.
We define the susceptibility of G to be

$$X(G) \quad = \quad \frac{1}{n} \sum_{i=1}^{r} |C_i|^2.$$

Note: $X(G)$ is the expected component size of a vertex chosen uniformly at random.
Let e be a random edge in G.

Susceptibility

Let G be a graph with connected components $C_1, \ldots, C_r$.
We define the susceptibility of G to be

$$X(G) \;=\; \frac{1}{n}\sum_{i=1}^{r} |C_i|^2.$$

Note: $X(G)$ is the expected component size of a vertex chosen uniformly at random.
Let e be a random edge in G.
The probability that e joins C_i and C_j is $\frac{|C_i|\;|C_j|}{n^2}$.

Susceptibility

Let G be a graph with connected components $C_1, \ldots, C_r$.
We define the susceptibility of G to be

$$X(G) \;=\; \frac{1}{n}\sum_{i=1}^{r}|C_i|^2.$$

Note: $X(G)$ is the expected component size of a vertex chosen uniformly at random.
Let e be a random edge in G.
The probability that e joins C_i and C_j is $\frac{|C_i|\;|C_j|}{n^2}$.

If $i \neq j$ then

$$X(G+e) - X(G) \;=\; \frac{1}{n}\left(|C_i| + |C_j|\right)^2 - \frac{1}{n}|C_i|^2 - \frac{1}{n}|C_j|^2$$

Susceptibility

Let G be a graph with connected components $C_1, \ldots, C_r$.
We define the susceptibility of G to be

$$X(G) \;=\; \frac{1}{n}\sum_{i=1}^{r} |C_i|^2.$$

Note: $X(G)$ is the expected component size of a vertex chosen uniformly at random.
Let e be a random edge in G.
The probability that e joins C_i and C_j is $\frac{|C_i|\;|C_j|}{n^2}$.

If $i \neq j$ then

$$X(G+e)-X(G) \;=\; \tfrac{1}{n}\left(|C_i|+|C_j|\right)^2 - \tfrac{1}{n}|C_i|^2 - \tfrac{1}{n}|C_j|^2 \;=\; \tfrac{2}{n}|C_i||C_j|.$$

Susceptibility, continued

Susceptibility, continued

$$E[X(G+e) - X(G)] \quad = \quad \sum_{i \neq j} \frac{|C_i||C_j|}{n^2} \quad \frac{2}{n}|C_i||C_j|$$

Susceptibility, continued

$$E[X(G+e)-X(G)] = \sum_{i \neq j} \frac{|C_i||C_j|}{n^2} \; \frac{2}{n}|C_i||C_j|$$

$$E[X(G+e)-X(G)] = \frac{2}{n}\left(\sum_{i=1}^{r} \frac{|C_i|^2}{n}\right)^2 - 2\sum_{i=1}^{r} \frac{|C_i|^4}{n^3}$$

Susceptibility, continued

$$E[X(G+e)-X(G)] = \sum_{i\neq j} \frac{|C_i||C_j|}{n^2} \quad \frac{2}{n}|C_i||C_j|$$

$$E[X(G+e)-X(G)] = \frac{2}{n}\left(\sum_{i=1}^{r} \frac{|C_i|^2}{n}\right)^2 - 2\sum_{i=1}^{r} \frac{|C_i|^4}{n^3}$$

$$E[X(G+e)-X(G)] = \tfrac{2}{n}X^2(G) - o(1)$$

Susceptibility, continued

$$E[X(G+e)-X(G)] = \sum_{i \neq j} \frac{|C_i||C_j|}{n^2} \quad \frac{2}{n}|C_i||C_j|$$

$$E[X(G+e)-X(G)] = \frac{2}{n}\left(\sum_{i=1}^{r} \frac{|C_i|^2}{n}\right)^2 - 2\sum_{i=1}^{r} \frac{|C_i|^4}{n^3}$$

$$E[X(G+e)-X(G)] = \tfrac{2}{n} X^2(G)$$

Susceptibility, continued

$$E[X(G+e)-X(G)] \;=\; \sum_{i\neq j}\frac{|C_i||C_j|}{n^2} \;\; \frac{2}{n}|C_i||C_j|$$

$$E[X(G+e)-X(G)] \;=\; \frac{2}{n}\left(\sum_{i=1}^{r}\frac{|C_i|^2}{n}\right)^2 \;-\; 2\sum_{i=1}^{r}\frac{|C_i|^4}{n^3}$$

$$E[X(G+e)-X(G)] \;=\; \tfrac{2}{n}X^2(G) \quad\Rightarrow\quad f' = 2f^2$$

Susceptibility, continued

$$E[X(G+e)-X(G)] \quad = \quad \sum_{i\neq j} \frac{|C_i||C_j|}{n^2} \quad \frac{2}{n}|C_i||C_j|$$

$$E[X(G+e)-X(G)] \quad = \quad \frac{2}{n}\left(\sum_{i=1}^{r}\frac{|C_i|^2}{n}\right)^2 \quad - \quad 2\sum_{i=1}^{r}\frac{|C_i|^4}{n^3}$$

$$E[X(G+e)-X(G)] \quad = \quad \tfrac{2}{n}X^2(G) \quad \Rightarrow \quad f' = 2f^2$$

Solving $f' = 2f^2$ and $f(0) = 1$

Susceptibility, continued

$$E[X(G+e) - X(G)] \quad = \quad \sum_{i \neq j} \frac{|C_i||C_j|}{n^2} \quad \frac{2}{n}|C_i||C_j|$$

$$E[X(G+e) - X(G)] \quad = \quad \frac{2}{n}\left(\sum_{i=1}^{r} \frac{|C_i|^2}{n}\right)^2 \quad - \quad 2\sum_{i=1}^{r} \frac{|C_i|^4}{n^3}$$

$$E[X(G+e) - X(G)] \quad = \quad \tfrac{2}{n}X^2(G) \quad \Rightarrow \quad f' = 2f^2$$

Solving $f' = 2f^2$ and $f(0) = 1$ gives $f(x) \;=\; \frac{1}{1-2x}$,

Susceptibility, continued

$$E[X(G+e) - X(G)] \quad = \quad \sum_{i \neq j} \frac{|C_i||C_j|}{n^2} \quad \frac{2}{n}|C_i||C_j|$$

$$E[X(G+e) - X(G)] \quad = \quad \frac{2}{n}\left(\sum_{i=1}^{r} \frac{|C_i|^2}{n}\right)^2 \quad - \quad 2\sum_{i=1}^{r} \frac{|C_i|^4}{n^3}$$

$$E[X(G+e) - X(G)] \quad = \quad \tfrac{2}{n} X^2(G) \quad \Rightarrow \quad f' = 2f^2$$

Solving $f' = 2f^2$ and $f(0) = 1$ gives $f(x) \;=\; \frac{1}{1-2x}$, which blows up at $x = \frac{1}{2}$.

A variation on the theme

A variation on the theme

Let c be a constant and let $(e_1, f_1),\ (e_2, f_2),\ \ldots,\ (e_{cn}, f_{cn})$ be a sequence of cn pairs of random edges on $[n]$.

A variation on the theme

Let c be a constant and let $(e_1, f_1),\ (e_2, f_2),\ \ldots,\ (e_{cn}, f_{cn})$ be a sequence of cn pairs of random edges on $[n]$.

We are going to examine two types of algorithms which choose one edge from each pair:

A variation on the theme

Let c be a constant and let $(e_1, f_1),\ (e_2, f_2),\ \ldots,\ (e_{cn}, f_{cn})$ be a sequence of cn pairs of random edges on $[n]$.

We are going to examine two types of algorithms which choose one edge from each pair:

- Offline Algorithms - All cn pairs are presented and then the cn choices are made.

A variation on the theme

Let c be a constant and let $(e_1, f_1),\ (e_2, f_2),\ \ldots,\ (e_{cn}, f_{cn})$ be a sequence of cn pairs of random edges on $[n]$.

We are going to examine two types of algorithms which choose one edge from each pair:

- Offline Algorithms - All cn pairs are presented and then the cn choices are made.
- Online Algorithms - Pairs appear sequentially.

A variation on the theme

Let c be a constant and let $(e_1, f_1),\ (e_2, f_2),\ \ldots,\ (e_{cn}, f_{cn})$ be a sequence of cn pairs of random edges on $[n]$.

We are going to examine two types of algorithms which choose one edge from each pair:

- Offline Algorithms - All cn pairs are presented and then the cn choices are made.
- Online Algorithms - Pairs appear sequentially.
  - The choice between e_i, f_i is made upon presentation, without knowledge of future edges.

A variation on the theme

Let c be a constant and let $(e_1, f_1),\ (e_2, f_2),\ \ldots,\ (e_{cn}, f_{cn})$ be a sequence of cn pairs of random edges on $[n]$.

We are going to examine two types of algorithms which choose one edge from each pair:

- Offline Algorithms - All cn pairs are presented and then the cn choices are made.
- Online Algorithms - Pairs appear sequentially.
  - The choice between e_i, f_i is made upon presentation, without knowledge of future edges.
  - This is called an Achlioptas Process, named after Dimitris Achlioptas who first posed the question of online avoidance of a giant component.

Avoiding a Giant Component

Avoiding a Giant Component

The interesting case for online avoidance is $c > \frac{1}{2}$.

Avoiding a Giant Component

The interesting case for online avoidance is $c > \frac{1}{2}$.

- (2001) Bohman and Frieze gave an online algorithm which avoids a giant **whp** for $c < 0.535$.

Avoiding a Giant Component

The interesting case for online avoidance is $c > \frac{1}{2}$.

- (2001) Bohman and Frieze gave an online algorithm which avoids a giant **whp** for $c < 0.535$.
- Spencer and Wormald claim they can achieve $c < 0.89$.

Avoiding a Giant Component

The interesting case for online avoidance is $c > \frac{1}{2}$.

- (2001) Bohman and Frieze gave an online algorithm which avoids a giant **whp** for $c < 0.535$.
- Spencer and Wormald claim they can achieve $c < 0.89$.
- (2006) Bohman and Kim showed that the offline version has a threshold $c_{off} \approx 0.976$:

Avoiding a Giant Component

The interesting case for online avoidance is $c > \frac{1}{2}$.

- (2001) Bohman and Frieze gave an online algorithm which avoids a giant **whp** for $c < 0.535$.
- Spencer and Wormald claim they can achieve $c < 0.89$.
- (2006) Bohman and Kim showed that the offline version has a threshold $c_{off} \approx 0.976$:
  - If $c < c_{off}$ then **whp** one can avoid a giant.
  - If $c > c_{off}$ then **whp** one can not avoid a giant.

Avoiding a Giant Component

The interesting case for online avoidance is $c > \frac{1}{2}$.

- (2001) Bohman and Frieze gave an online algorithm which avoids a giant **whp** for $c < 0.535$.
- Spencer and Wormald claim they can achieve $c < 0.89$.
- (2006) Bohman and Kim showed that the offline version has a threshold $c_{off} \approx 0.976$:
  - If $c < c_{off}$ then **whp** one can avoid a giant.
  - If $c > c_{off}$ then **whp** one can not avoid a giant.
- (Bohman, Frieze, Wormald, 2004)
  A giant can not be avoided online for $c < 0.9668 < c_{off}$.

Avoiding a Giant Component

The interesting case for online avoidance is $c > \frac{1}{2}$.

- (2001) Bohman and Frieze gave an online algorithm which avoids a giant **whp** for $c < 0.535$.
- Spencer and Wormald claim they can achieve $c < 0.89$.
- (2006) Bohman and Kim showed that the offline version has a threshold $c_{off} \approx 0.976$:
  - If $c < c_{off}$ then **whp** one can avoid a giant.
  - If $c > c_{off}$ then **whp** one can not avoid a giant.
- (Bohman, Frieze, Wormald, 2004)
  A giant can not be avoided online for $c < 0.9668 < c_{off}$.
  This separates online and offline.

General Online Processes

General Online Processes

- Let A be an online algorithm for the choice of one edge from each presented pair (e_i, f_i) without "future knowledge".

General Online Processes

- Let A be an online algorithm for the choice of one edge from each presented pair (e_i, f_i) without "future knowledge".
- Let $G_A(i)$ be the graph this algorithm creates after i choices.

General Online Processes

- Let A be an online algorithm for the choice of one edge from each presented pair (e_i, f_i) without "future knowledge".
- Let $G_A(i)$ be the graph this algorithm creates after i choices.
- This produces a random graph process $G_A(1), G_A(2), \ldots, G_A(cn)$.

General Online Processes

- Let A be an online algorithm for the choice of one edge from each presented pair (e_i, f_i) without "future knowledge".
- Let $G_A(i)$ be the graph this algorithm creates after i choices.
- This produces a random graph process $G_A(1), G_A(2), \ldots, G_A(cn)$.
- A size algorithm A makes the choices between edges e_{i+1} and f_{i+1} based on the sizes of the components in $G_A(i)$.

General Online Processes

- Let A be an online algorithm for the choice of one edge from each presented pair (e_i, f_i) without "future knowledge".
- Let $G_A(i)$ be the graph this algorithm creates after i choices.
- This produces a random graph process $G_A(1), G_A(2), \ldots, G_A(cn)$.
- A size algorithm A makes the choices between edges e_{i+1} and f_{i+1} based on the sizes of the components in $G_A(i)$.
- A bounded size algorithm is a size algorithm that makes no distinction between components larger than some fixed constant m.

General Online Processes

- Let A be an online algorithm for the choice of one edge from each presented pair (e_i, f_i) without "future knowledge".
- Let $G_A(i)$ be the graph this algorithm creates after i choices.
- This produces a random graph process $G_A(1), G_A(2), \ldots, G_A(cn)$.
- A size algorithm A makes the choices between edges e_{i+1} and f_{i+1} based on the sizes of the components in $G_A(i)$.
- A bounded size algorithm is a size algorithm that makes no distinction between components larger than some fixed constant m.
- A bounded first-edge algorithm is a bounded size algorithm that chooses between e_{i+1} and f_{i+1} only by looking at the sizes of the components in $G_A(i)$ connected by e_i.

Conjectures by Spencer (2001)

Conjectures by Spencer (2001)

Conjecture 1:

Any size algorithm A has a critical value t_0 such that :

Conjectures by Spencer (2001)

Conjecture 1:
Any size algorithm A has a critical value t_0 such that :

- If $c = t_0 - \epsilon$ then **whp** the largest component of $G_A(cn)$ has $O(\log n)$ vertices.

Conjectures by Spencer (2001)

Conjecture 1:
Any size algorithm A has a critical value t_0 such that :

- If $c = t_0 - \epsilon$ then **whp** the largest component of $G_A(cn)$ has $O(\log n)$ vertices.
- If $c = t_0 + \epsilon$ then **whp** the largest component of $G_A(cn)$ has $\Omega(n)$ vertices, and all other components are of size $O(\log n)$.

Conjectures by Spencer, continued

We can model the susceptibility for any bounded size algorithm A using a differential equation, and it has a blow-up point c_A.

Conjectures by Spencer, continued

We can model the susceptibility for any bounded size algorithm A using a differential equation, and it has a blow-up point c_A. ($c_A = \frac{1}{2}$ in the Erdős and Rényi model)

Conjectures by Spencer, continued

We can model the susceptibility for any bounded size algorithm A using a differential equation, and it has a blow-up point c_A. ($c_A = \frac{1}{2}$ in the Erdős and Rényi model)

Conjecture 2: Let A_1 be the bounded size algorithm that takes e_{i+1} if and only if it joins two isolated vertices in $G_{A_1}(t)$.

Conjectures by Spencer, continued

We can model the susceptibility for any bounded size algorithm A using a differential equation, and it has a blow-up point c_A. ($c_A = \frac{1}{2}$ in the Erdős and Rényi model)

Conjecture 2: Let A_1 be the bounded size algorithm that takes e_{i+1} if and only if it joins two isolated vertices in $G_{A_1}(t)$.

- If $c > c_{A_1}$ then **whp** $G_{A_1}(cn)$ has a component of size $\Omega(n)$.

Conjectures by Spencer, continued

We can model the susceptibility for any bounded size algorithm A using a differential equation, and it has a blow-up point c_A. ($c_A = \frac{1}{2}$ in the Erdős and Rényi model)

Conjecture 2: Let A_1 be the bounded size algorithm that takes e_{i+1} if and only if it joins two isolated vertices in $G_{A_1}(t)$.

- If $c > c_{A_1}$ then **whp** $G_{A_1}(cn)$ has a component of size $\Omega(n)$.
- If $c < c_{A_1}$ then **whp** all components of $G_{A_1}(cn)$ have size $O(\log n)$.

Conjectures by Spencer, continued

We can model the susceptibility for any bounded size algorithm A using a differential equation, and it has a blow-up point c_A. ($c_A = \frac{1}{2}$ in the Erdős and Rényi model)

Conjecture 3:

Conjectures by Spencer, continued

We can model the susceptibility for any bounded size algorithm A using a differential equation, and it has a blow-up point c_A. ($c_A = \frac{1}{2}$ in the Erdős and Rényi model)

Conjecture 3: Let A be the *any* bounded size algorithm .

Conjectures by Spencer, continued

We can model the susceptibility for any bounded size algorithm A using a differential equation, and it has a blow-up point c_A. ($c_A = \frac{1}{2}$ in the Erdős and Rényi model)

Conjecture 3: Let A be the *any* bounded size algorithm .

- If $c > c_A$ then **whp** $G_A(cn)$ has a component of size $\Omega(n)$.
- If $c < c_A$ then **whp** all components of $G_A(cn)$ have size $O(\log n)$.

Theorem 1

Theorem 1

If A is a bounded first-edge algorithm, c_A exists such that

Theorem 1

If A is a bounded first-edge algorithm, c_A exists such that

1. If $c < c_A$ then **whp** the largest component in the graph $G_A(cn)$ has $O\left(n^{12/13} \log n\right)$ vertices.

Theorem 1

If A is a bounded first-edge algorithm, c_A exists such that

1. If $c < c_A$ then **whp** the largest component in the graph $G_A(cn)$ has $O\left(n^{12/13} \log n\right)$ vertices.
2. If $c > c_A$ then **whp** $G_A(cn)$ has a component of size $\Omega(n)$.

Theorem 1

If A is a bounded first-edge algorithm, c_A exists such that

1. If $c < c_A$ then **whp** the largest component in the graph $G_A(cn)$ has $O\left(n^{12/13} \log n\right)$ vertices.
2. If $c > c_A$ then **whp** $G_A(cn)$ has a component of size $\Omega(n)$.

Notes:

Theorem 1

If A is a bounded first-edge algorithm, c_A exists such that

1. If $c < c_A$ then **whp** the largest component in the graph $G_A(cn)$ has $O\left(n^{12/13} \log n\right)$ vertices.
2. If $c > c_A$ then **whp** $G_A(cn)$ has a component of size $\Omega(n)$.

Notes:

- Spencer and Wormald independently proved Theorem 1, allowing for A to be any *bounded size algorithm* and showed that the largest component in (1.) is $O(\log n)$ **whp**.

Theorem 1

If A is a bounded first-edge algorithm, c_A exists such that

1. If $c < c_A$ then **whp** the largest component in the graph $G_A(cn)$ has $O\left(n^{12/13} \log n\right)$ vertices.
2. If $c > c_A$ then **whp** $G_A(cn)$ has a component of size $\Omega(n)$.

Notes:

- Spencer and Wormald independently proved Theorem 1, allowing for A to be any *bounded size algorithm* and showed that the largest component in (1.) is $O(\log n)$ **whp**.
- The main tool in our proof is the differential equations method.

Theorem 1

If A is a bounded first-edge algorithm, c_A exists such that

1. If $c < c_A$ then **whp** the largest component in the graph $G_A(cn)$ has $O\left(n^{12/13}\log n\right)$ vertices.
2. If $c > c_A$ then **whp** $G_A(cn)$ has a component of size $\Omega(n)$.

Notes:

- Spencer and Wormald independently proved Theorem 1, allowing for A to be any *bounded size algorithm* and showed that the largest component in (1.) is $O(\log n)$ **whp**.
- The main tool in our proof is the differential equations method.
- The critical value c_A is the given by the blow-up point in the differential equation for the susceptibility.

Another Algorithm

Another Algorithm

Note: If trying to *create* a giant as fast as possible, then the interesting case is $c \in (\frac{1}{4}, \frac{1}{2})$.

Another Algorithm

Note: If trying to *create* a giant as fast as possible, then the interesting case is $c \in (\frac{1}{4}, \frac{1}{2})$.

- If $c < \frac{1}{4}$ then selecting all $2cn$ edges won't make a giant **whp**.

Another Algorithm

Note: If trying to *create* a giant as fast as possible, then the interesting case is $c \in (\frac{1}{4}, \frac{1}{2})$.

- If $c < \frac{1}{4}$ then selecting all $2cn$ edges won't make a giant **whp**.
- If $c > \frac{1}{2}$ then selecting e_i every time makes a giant **whp**.

Another Algorithm

Note: If trying to *create* a giant as fast as possible, then the interesting case is $c \in (\frac{1}{4}, \frac{1}{2})$.

- If $c < \frac{1}{4}$ then selecting all $2cn$ edges won't make a giant **whp**.
- If $c > \frac{1}{2}$ then selecting e_i every time makes a giant **whp**.

Given pair (e_i, f_i), accept $e_i = \{u_i, v_i\}$ if and only if neither u_i nor v_i is an isolated vertex.

Another Algorithm

Note: If trying to *create* a giant as fast as possible, then the interesting case is $c \in (\frac{1}{4}, \frac{1}{2})$.

- If $c < \frac{1}{4}$ then selecting all $2cn$ edges won't make a giant **whp**.
- If $c > \frac{1}{2}$ then selecting e_i every time makes a giant **whp**.

Given pair (e_i, f_i), accept $e_i = \{u_i, v_i\}$ if and only if neither u_i nor v_i is an isolated vertex.

This algorithm is designed to create a giant component relatively quickly, and indeed it does:

Another Algorithm

Note: If trying to *create* a giant as fast as possible, then the interesting case is $c \in (\frac{1}{4}, \frac{1}{2})$.

- If $c < \frac{1}{4}$ then selecting all $2cn$ edges won't make a giant **whp**.
- If $c > \frac{1}{2}$ then selecting e_i every time makes a giant **whp**.

Given pair (e_i, f_i), accept $e_i = \{u_i, v_i\}$ if and only if neither u_i nor v_i is an isolated vertex.

This algorithm is designed to create a giant component relatively quickly, and indeed it does:

Theorem 2: If $c > 0.385$ then **whp** this algorithm will create a graph with a component of size $\Omega(n)$.

A Difference Between Online and Offline

A Difference Between Online and Offline

Theorem 3: If $c < 0.2544$ then for any Achlioptas process, **whp** all of the components of the graph created in cn steps will be of size $O(\log n)$.

A Difference Between Online and Offline

Theorem 3: If $c < 0.2544$ then for any Achlioptas process, **whp** all of the components of the graph created in cn steps will be of size $O(\log n)$.

Theorem 4: If $c > 0.25$ then **whp** there is a way to choose one edge from each pair and create a graph with a component of size $\Omega(n)$.

A Difference Between Online and Offline

Theorem 3: If $c < 0.2544$ then for any Achlioptas process, **whp** all of the components of the graph created in cn steps will be of size $O(\log n)$.

Theorem 4: If $c > 0.25$ then **whp** there is a way to choose one edge from each pair and create a graph with a component of size $\Omega(n)$.

Theorem 4: If $c > 0.25$ then **whp** there is a way to choose one edge from each $pair$ and create a graph with a component of size $\Omega(n)$.

A Difference Between Online and Offline

Theorem 3: If $c < 0.2544$ then for any Achlioptas process, **whp** all of the components of the graph created in cn steps will be of size $O(\log n)$.

Theorem 4: If $c > 0.25$ then **whp** there is a way to choose one edge from each pair and create a graph with a component of size $\Omega(n)$.

Theorem 4: If $c > \frac{1}{2*2}$ then **whp** there is a way to choose one edge from each $2 - tuple$ and create a graph with a component of size $\Omega(n)$.

A Difference Between Online and Offline

Theorem 3: If $c < 0.2544$ then for any Achlioptas process, **whp** all of the components of the graph created in cn steps will be of size $O(\log n)$.

Theorem 4: If $c > 0.25$ then **whp** there is a way to choose one edge from each pair and create a graph with a component of size $\Omega(n)$.

Theorem 4b: If $c > \frac{1}{2k}$ then **whp** there is a way to choose one edge from each $k - tuple$ and create a graph with a component of size $\Omega(n)$.

k-SAT Formulas

k-SAT Formulas

For some fixed n, we take Boolean variables

$$\{x_1, x_2, \ldots, x_n\}.$$

k-SAT Formulas

For some fixed n, we take Boolean variables

$$\{x_1, x_2, \ldots, x_n\}.$$

Then, k-clauses are chosen from the literals

$$\{x_1, x_2, \ldots, x_n, \overline{x}_1, \overline{x}_2, \ldots, \overline{x}_n\}$$

k-SAT Formulas

For some fixed n, we take Boolean variables

$$\{x_1, x_2, \ldots, x_n\}.$$

Then, k-clauses are chosen from the literals

$$\{x_1, x_2, \ldots, x_n, \overline{x}_1, \overline{x}_2, \ldots, \overline{x}_n\}$$

An assignment must exist to the Boolean variables which satisfies every clause.

Example:

Example:

$n = 4$, $k = 2$, our clauses are:

Example:

$n = 4$, $k = 2$, our clauses are:

$$\{x_1, x_2\} \quad \{\overline{x}_2, x_4\} \quad \{\overline{x}_1, x_3\} \quad \{\overline{x}_2, x_3\} \quad \{x_2, \overline{x}_4\} \quad \{\overline{x}_3, \overline{x}_4\}$$

Example:

$n = 4$, $k = 2$, our clauses are:

$$\{x_1, x_2\} \quad \{\overline{x}_2, x_4\} \quad \{\overline{x}_1, x_3\} \quad \{\overline{x}_2, x_3\} \quad \{x_2, \overline{x}_4\} \quad \{\overline{x}_3, \overline{x}_4\}$$

x_1	x_2	x_3	x_4
$\overline{x}_1$	$\overline{x}_2$	$\overline{x}_3$	$\overline{x}_4$

Example:

$n = 4$, $k = 2$, our clauses are:

$$\{x_1, x_2\} \quad \{\overline{x}_2, x_4\} \quad \{\overline{x}_1, x_3\} \quad \{\overline{x}_2, x_3\} \quad \{x_2, \overline{x}_4\} \quad \{\overline{x}_3, \overline{x}_4\}$$

x_1	x_2	x_3	x_4
$\overline{x}_1$	$\overline{x}_2$	$\overline{x}_3$	$\overline{x}_4$

Example:

$n = 4$, $k = 2$, our clauses are:

$$\{x_1, x_2\} \quad \{\overline{x}_2, x_4\} \quad \{\overline{x}_1, x_3\} \quad \{\overline{x}_2, x_3\} \quad \{x_2, \overline{x}_4\} \quad \underline{\{\overline{x}_3, \overline{x}_4\}}$$

x_1	x_2	x_3	x_4
$\overline{x}_1$	$\overline{x}_2$	$\overline{x}_3$	$\overline{x}_4$

Example:

$n = 4$, $k = 2$, our clauses are:

$$\{x_1, x_2\} \quad \underline{\{\overline{x}_2, x_4\}} \quad \{\overline{x}_1, x_3\} \quad \{\overline{x}_2, x_3\} \quad \{x_2, \overline{x}_4\} \quad \{\overline{x}_3, \overline{x}_4\}$$

x_1	x_2	x_3	x_4
$\overline{x}_1$	$\overline{x}_2$	$\overline{x}_3$	$\overline{x}_4$

Example:

$n = 4$, $k = 2$, our clauses are:

$$\{x_1, x_2\} \quad \{\overline{x}_2, x_4\} \quad \{\overline{x}_1, x_3\} \quad \{\overline{x}_2, x_3\} \quad \{x_2, \overline{x}_4\} \quad \{\overline{x}_3, \overline{x}_4\}$$

x_1	x_2	x_3	x_4
$\overline{x}_1$	$\overline{x}_2$	$\overline{x}_3$	$\overline{x}_4$

Random k-SAT

Random k-SAT

- A *randomly generated* k*-clause* is one whose k literals are chosen uniformly at random from all $2n$ possibilities.

Random k-SAT

- A *randomly generated k-clause* is one whose k literals are chosen uniformly at random from all $2n$ possibilities.
- For any constant c, let $F_k(cn)$ be a family of cn randomly generated k-clauses.

Random k-SAT

- A *randomly generated k-clause* is one whose k literals are chosen uniformly at random from all $2n$ possibilities.
- For any constant c, let $F_k(cn)$ be a family of cn randomly generated k-clauses.

Theorem: (Chvátal, Reed, 1992) As $n \to \infty$,

- If $c < 1$ then $\Pr[F_2(cn)$ is satisfiable $] \to 1$.
- If $c > 1$ then $\Pr[F_2(cn)$ is satisfiable $] \to 0$.

Random k-SAT

- A *randomly generated k-clause* is one whose k literals are chosen uniformly at random from all $2n$ possibilities.
- For any constant c, let $F_k(cn)$ be a family of cn randomly generated k-clauses.

Theorem: (Chvátal, Reed, 1992) As $n \to \infty$,

- If $c < 1$ then $\Pr[F_2(cn)$ is satisfiable $] \to 1$.
- If $c > 1$ then $\Pr[F_2(cn)$ is satisfiable $] \to 0$.

$c = 1$ is called the threshold density for $k = 2$.

Threshold Density

Threshold Density

Satisfiability Threshold Conjecture

For each $k > 2$ there exists a threshold density c_k such that:

- If $c < c_k$ then $F_k(cn)$ is satisfiable with high probability.
- If $c > c_k$ then $F_k(cn)$ is not satisfiable with high probability.

Threshold Density

Satisfiability Threshold Conjecture

For each $k > 2$ there exists a threshold density c_k such that:

- If $c < c_k$ then $F_k(cn)$ is satisfiable with high probability.
- If $c > c_k$ then $F_k(cn)$ is not satisfiable with high probability.

Theorem: (Friedgut, 1999)

For each k and n there exists a threshold density function $c_k(n)$.

Threshold Density

Satisfiability Threshold Conjecture

For each $k > 2$ there exists a threshold density c_k such that:

- If $c < c_k$ then $F_k(cn)$ is satisfiable with high probability.
- If $c > c_k$ then $F_k(cn)$ is not satisfiable with high probability.

Theorem: (Friedgut, 1999)

For each k and n there exists a threshold density function $c_k(n)$.

- For a given k, each n may have *its own* threshold $c_k(n)$.

Threshold Density

Satisfiability Threshold Conjecture

For each $k > 2$ there exists a threshold density c_k such that:

- If $c < c_k$ then $F_k(cn)$ is satisfiable with high probability.
- If $c > c_k$ then $F_k(cn)$ is not satisfiable with high probability.

Theorem: (Friedgut, 1999)

For each k and n there exists a threshold density function $c_k(n)$.

- For a given k, each n may have *its own* threshold $c_k(n)$.
- These thresholds may not converge to a limit as $n \to \infty$.

Threshold Density

Satisfiability Threshold Conjecture

For each $k > 2$ there exists a threshold density c_k such that:

- If $c < c_k$ then $F_k(cn)$ is satisfiable with high probability.
- If $c > c_k$ then $F_k(cn)$ is not satisfiable with high probability.

Theorem: (Friedgut, 1999)

For each k and n there exists a threshold density function $c_k(n)$.

- For a given k, each n may have *its own* threshold $c_k(n)$.
- These thresholds may not converge to a limit as $n \to \infty$.

Threshold Density

Satisfiability Threshold Conjecture

For each $k > 2$ there exists a threshold density c_k such that:

- If $c < c_k$ then $F_k(cn)$ is satisfiable with high probability.
- If $c > c_k$ then $F_k(cn)$ is not satisfiable with high probability.

Theorem: (Friedgut, 1999)

For each k and n there exists a threshold density function $c_k(n)$.

- For a given k, each n may have *its own* threshold $c_k(n)$.
- These thresholds may not converge to a limit as $n \to \infty$.
- Friedgut used Fourier Analysis in his proof of this theorem.

Things we do know

Things we do know

Theorem: (Achlioptas, Naor, Peres, 2003)
For $k \geq 1$, there exist constants α and β such that

- If $c < 2^k \ln 2 - \alpha k$ then $F_k(cn)$ is satisfiable **whp**.
- If $c > 2^k \ln 2 - \beta k$ then $F_k(cn)$ is not satisfiable **whp**.

Things we do know

Theorem: (Achlioptas, Naor, Peres, 2003)
For $k \geq 1$, there exist constants α and β such that

- If $c < 2^k \ln 2 - \alpha k$ then $F_k(cn)$ is satisfiable **whp**.
- If $c > 2^k \ln 2 - \beta k$ then $F_k(cn)$ is not satisfiable **whp**.

Theorem: (Kaporis, Kirousis, Lalas, 2002)

$F_3(3.42n)$ is satisfiable with high probability.

Things we do know

Theorem: (Achlioptas, Naor, Peres, 2003)
For $k \geq 1$, there exist constants α and β such that

- If $c < 2^k \ln 2 - \alpha k$ then $F_k(cn)$ is satisfiable **whp**.
- If $c > 2^k \ln 2 - \beta k$ then $F_k(cn)$ is not satisfiable **whp**.

Theorem: (Kaporis, Kirousis, Lalas, 2002)
$F_3(3.42n)$ is satisfiable with high probability.

Theorem: (Dubois, Boufkhad, Mandler, 2000)
$F_3(4.6n)$ is not satisfiable with high probability.

Online Version

Online Version

- We are given cn *randomly chosen* k-clauses, one at a time.

Online Version

- We are given cn *randomly chosen* k-clauses, one at a time.
- Accept or reject each clause as it is presented with no knowledge of what is coming.

Online Version

- We are given cn *randomly chosen* k-clauses, one at a time.
- Accept or reject each clause as it is presented with no knowledge of what is coming.
- ALL clauses taken must be satisfied.

Online Version

- We are given cn *randomly chosen* k-clauses, one at a time.
- Accept or reject each clause as it is presented with no knowledge of what is coming.
- ALL clauses taken must be satisfied.

Question: What is the maximum expected number of clauses that an algorithm can accept?

Online Version

- We are given cn *randomly chosen* k-clauses, one at a time.
- Accept or reject each clause as it is presented with no knowledge of what is coming.
- ALL clauses taken must be satisfied.

Question: What is the maximum expected number of clauses that an algorithm can accept?

(either c is fixed or $c \to \infty$)

Easy

There is an online algorithm which accepts an expected $(1 - \frac{1}{2^k})cn$ clauses.

Easy

There is an online algorithm which accepts an expected $(1 - \frac{1}{2^k})cn$ clauses.

Begin by setting all variables to true, then accept any clause which doesn't have everything false.

Easy

There is an online algorithm which accepts an expected $(1 - \frac{1}{2^k})cn$ clauses.

Begin by setting all variables to true, then accept any clause which doesn't have everything false.

So, if $k = 2$ then accept $\{{\color{green}\bullet}, {\color{green}\bullet}\}$, $\{{\color{green}\bullet}, {\color{red}\bullet}\}$, $\{{\color{red}\bullet}, {\color{green}\bullet}\}$, and reject $\{{\color{red}\bullet}, {\color{red}\bullet}\}$. This accepts an expected $\frac{3}{4}cn$ clauses.

Online-Lazy

Online-Lazy

Given: $(k = 2)$	Accept?	Set to:
{●(yellow), ●(yellow)} , {●(yellow), ●(red)} , {●(red), ●(yellow)}	Yes	{●(green), ●(yellow)}, {●(green), ●(red)} , {●(red), ●(green)}
{●(red), ●(red)}	No	
{●(red), ●(green)}, {●(green), ●(red)}, {●(green), ●(yellow)}, {●(yellow), ●(green)}, {●(green), ●(green)}	Yes	

Online-Lazy

Given: ($k = 2$)	**Accept?**	**Set to:**
{🟡, 🟡} ,{🟡, 🔴} , {🔴, 🟡}	Yes	{🟢, 🟡}, {🟢, 🔴} , {🔴, 🟢}
{🔴, 🔴}	No	
{🔴, 🟢},{🟢, 🔴}, {🟢, 🟡}, {🟡, 🟢}, {🟢, 🟢}	Yes	

This accepts an expected $\frac{3}{4}cn + \frac{3}{8}n$ clauses as $c \to \infty$.

Online-Lazy

Given: ($k=2$)	Accept?	Set to:
{🟡, 🟡} ,{🟡, 🔴} , {🔴, 🟡}	Yes	{🟢, 🟡}, {🟢, 🔴} , {🔴, 🟢}
{🔴, 🔴}	No	
{🔴, 🟢},{🟢, 🔴}, {🟢, 🟡}, {🟡, 🟢}, {🟢, 🟢}	Yes	

This accepts an expected $(1-\frac{1}{2^k})cn + a_k n$ clauses as $c \to \infty$.

k	1	**2**	3	4	5	10
a_k	0.5	**0.375**	$0.2842\ldots$	$0.2209\ldots$	$0.1765\ldots$	$0.0809\ldots$

Offline Version

Offline Version

Here we look at all cn clauses and take as many as possible.

Offline Version

Here we look at all cn clauses and take as many as possible.

For the equivalent offline problem, we expect to take $(1 - \frac{1}{2^k})cn + \Theta(\sqrt{c})n$ out of cn clauses.
(Coppersmith, Gamarnik, Hajiaghayi, Sorkin, 2004)

Offline Version

Here we look at all cn clauses and take as many as possible.

For the equivalent offline problem, we expect to take $(1 - \frac{1}{2^k})cn + \Theta(\sqrt{c})n$ out of cn clauses.
(Coppersmith, Gamarnik, Hajiaghayi, Sorkin, 2004)

Therefore, an optimal online algorithm is somewhere between $(1 - \frac{1}{2^k})cn + a_k n$ and $(1 - \frac{1}{2^k})cn + \Theta(\sqrt{c})n$.

Offline Version

Here we look at all cn clauses and take as many as possible.

For the equivalent offline problem, we expect to take $(1 - \frac{1}{2^k})cn + \Theta(\sqrt{c})n$ out of cn clauses.
(Coppersmith, Gamarnik, Hajiaghayi, Sorkin, 2004)

Therefore, an optimal online algorithm is somewhere between $(1 - \frac{1}{2^k})cn + a_k n$ and $(1 - \frac{1}{2^k})cn + \Theta(\sqrt{c})n$.

Theorem: (K,05)
Any online algorithm accepts less than $(1 - \frac{1}{2^k})cn + \ln 2n$ clauses with high probability.

A naive algorithm

A naive algorithm

1. Begin by accepting every possible clause for as long as possible.

A naive algorithm

1. Begin by accepting every possible clause for as long as possible.
2. When given the first clause that can't be accepted, reject it and set all the variables.

A naive algorithm

1. Begin by accepting every possible clause for as long as possible.
2. When given the first clause that can't be accepted, reject it and set all the variables.
3. Accept all remaining clauses if and only if they are satisfied by our assignment.

A naive algorithm

1. Begin by accepting every possible clause for as long as possible.
2. When given the first clause that can't be accepted, reject it and set all the variables.
3. Accept all remaining clauses if and only if they are satisfied by our assignment.

Claim: With high probability this will accept

$$\left(1 - \frac{1}{2^k}\right) cn + (\ln 2 - o_k(1))n$$

out of cn clauses.

A naive algorithm

1. Begin by accepting every possible clause for as long as possible.
2. When given the first clause that can't be accepted, reject it and set all the variables.
3. Accept all remaining clauses if and only if they are satisfied by our assignment.

Claim: With high probability this will accept

$$\left(1 - \frac{1}{2^k}\right) cn + (\ln 2 - o_k(1))n$$

out of cn clauses.

Corollary: The naive algorithm is asymptotically optimal.

Proof of Claim:

Proof of Claim:

Let $\lambda = 2^k \ln 2 - \alpha k$.
Both of the following are true with high probability:

Proof of Claim:

Let $\lambda = 2^k \ln 2 - \alpha k$.
Both of the following are true with high probability:

- The first λn clauses will be accepted.

Proof of Claim:

Let $\lambda = 2^k \ln 2 - \alpha k$.
Both of the following are true with high probability:

- The first λn clauses will be accepted. Theorem, ANP, 03

Proof of Claim:

Let $\lambda = 2^k \ln 2 - \alpha k$.
Both of the following are true with high probability:

- The first λn clauses will be accepted. Theorem, ANP, 03
- $(1 - \frac{1}{2^k})(c - \lambda)n - O(\sqrt{n})$ of the remaining clauses will be accepted.

Proof of Claim:

Let $\lambda = 2^k \ln 2 - \alpha k$.
Both of the following are true with high probability:

- The first λn clauses will be accepted. Theorem, ANP, 03
- $(1 - \frac{1}{2^k})(c - \lambda)n - O(\sqrt{n})$ of the remaining clauses will be accepted. each of $(c - \lambda)n$ clauses accepted with probability $1 - \frac{1}{2^k}$

Proof of Claim:

Let $\lambda = 2^k \ln 2 - \alpha k$.
Both of the following are true with high probability:

- The first λn clauses will be accepted. Theorem, ANP, 03
- $(1 - \frac{1}{2^k})(c - \lambda)n - O(\sqrt{n})$ of the remaining clauses will be accepted. each of $(c - \lambda)n$ clauses accepted with probability $1 - \frac{1}{2^k}$

Total number of clauses accepted by naive algorithm:
$= \lambda n + \left(1 - \frac{1}{2^k}\right)(c - \lambda)n - O(\sqrt{n})$

Proof of Claim:

Let $\lambda = 2^k \ln 2 - \alpha k$.
Both of the following are true with high probability:

- The first λn clauses will be accepted. Theorem, ANP, 03
- $(1 - \frac{1}{2^k})(c - \lambda)n - O(\sqrt{n})$ of the remaining clauses will be accepted. each of $(c - \lambda)n$ clauses accepted with probability $1 - \frac{1}{2^k}$

Total number of clauses accepted by naive algorithm:

$$= \lambda n + \left(1 - \frac{1}{2^k}\right)(c - \lambda)n - O(\sqrt{n})$$

$$= \left(1 - \frac{1}{2^k}\right) cn + \left(\frac{1}{2^k}\lambda - O(\frac{1}{\sqrt{n}})\right) n$$

Proof of Claim:

Let $\lambda = 2^k \ln 2 - \alpha k$.
Both of the following are true with high probability:

- The first λn clauses will be accepted. Theorem, ANP, 03
- $(1 - \frac{1}{2^k})(c - \lambda)n - O(\sqrt{n})$ of the remaining clauses will be accepted. each of $(c - \lambda)n$ clauses accepted with probability $1 - \frac{1}{2^k}$

Total number of clauses accepted by naive algorithm:

$$= \lambda n + \left(1 - \frac{1}{2^k}\right)(c - \lambda)n - O(\sqrt{n})$$

$$= \left(1 - \frac{1}{2^k}\right) cn + \left(\frac{1}{2^k}\lambda - O(\frac{1}{\sqrt{n}})\right) n$$

$$= \left(1 - \frac{1}{2^k}\right) cn + \left(\ln 2 - \frac{\alpha k}{2^k} - O(\frac{1}{\sqrt{n}})\right) n$$

Proof of Claim:

Let $\lambda = 2^k \ln 2 - \alpha k$.
Both of the following are true with high probability:

- The first λn clauses will be accepted. Theorem, ANP, 03
- $(1 - \frac{1}{2^k})(c - \lambda)n - O(\sqrt{n})$ of the remaining clauses will be accepted. each of $(c-\lambda)n$ clauses accepted with probability $1 - \frac{1}{2^k}$

Total number of clauses accepted by naive algorithm:

$$= \lambda n + \left(1 - \frac{1}{2^k}\right)(c - \lambda)n - O(\sqrt{n})$$

$$= \left(1 - \frac{1}{2^k}\right)cn + \left(\frac{1}{2^k}\lambda - O(\frac{1}{\sqrt{n}})\right)n$$

$$= \left(1 - \frac{1}{2^k}\right)cn + \left(\ln 2 - \frac{\alpha k}{2^k} - O(\frac{1}{\sqrt{n}})\right)n$$

$$= \left(1 - \frac{1}{2^k}\right)cn + (\ln 2)n - o_k(1)n$$

Proof of Theorem Sketch   Fix $k = 3$

Proof of Theorem Sketch Fix $k = 3$

Theorem: Any online algorithm cannot accept

$$\frac{7}{8}cn + (\ln 2)n + o(n)$$

clauses with high probability.

Proof of Theorem Sketch Fix $k = 3$

Theorem: Any online algorithm cannot accept

$$\frac{7}{8}cn + (\ln 2)n + o(n)$$

clauses with high probability.

Proof: Let S_i be the set of valid assignments after i clauses.

Proof of Theorem Sketch   Fix $k = 3$

Theorem: Any online algorithm cannot accept

$$\frac{7}{8}cn + (\ln 2)n + o(n)$$

clauses with high probability.

Proof: Let S_i be the set of valid assignments after i clauses.

So, $|S_0| = 2^n$

Proof of Theorem Sketch Fix $k = 3$

Theorem: Any online algorithm cannot accept

$$\frac{7}{8}cn + (\ln 2)n + o(n)$$

clauses with high probability.

Proof: Let S_i be the set of valid assignments after i clauses.

So, $|S_0| = 2^n$ and $S_i \subseteq S_{i-1}$, with equality if clause i is rejected.

Proof of Theorem Sketch   Fix $k = 3$

Theorem: Any online algorithm cannot accept

$$\frac{7}{8}cn + (\ln 2)n + o(n)$$

clauses with high probability.

Proof: Let S_i be the set of valid assignments after i clauses.

So, $|S_0| = 2^n$ and $S_i \subseteq S_{i-1}$, with equality if clause i is rejected.

Let B_i be the number of clauses accepted minus $\frac{7}{8}cn$, i.e. the number accepted “beyond" what we can get easily.

Proof of Theorem Sketch Fix $k = 3$

Theorem: Any online algorithm cannot accept

$$\frac{7}{8}cn + (\ln 2)n + o(n)$$

clauses with high probability.

Proof: Let S_i be the set of valid assignments after i clauses.

So, $|S_0| = 2^n$ and $S_i \subseteq S_{i-1}$, with equality if clause i is rejected.

Let B_i be the number of clauses accepted minus $\frac{7}{8}cn$, i.e. the number accepted “beyond" what we can get easily.
Note that $B_{i+1} - B_i \leq \frac{1}{8}$.

Proof of Theorem Sketch Fix $k = 3$

Theorem: Any online algorithm cannot accept

$$\frac{7}{8}cn + (\ln 2)n + o(n)$$

clauses with high probability.

Proof: Let S_i be the set of valid assignments after i clauses.

So, $|S_0| = 2^n$ and $S_i \subseteq S_{i-1}$, with equality if clause i is rejected.

Let B_i be the number of clauses accepted minus $\frac{7}{8}cn$, i.e. the number accepted “beyond" what we can get easily.
Note that $B_{i+1} - B_i \leq \frac{1}{8}$.

Need to show $B_{cn} \leq n \ln 2 + o(n)$ with high probability.

Proof sketch Fix $k = 3$

Any step of the algorithm will fit one of these two cases:

Proof sketch   Fix $k = 3$

Any step of the algorithm will fit one of these two cases:

Case 1: Be “non-ambitious" and turn down any “beyond" clauses.

Proof sketch   Fix $k = 3$

Any step of the algorithm will fit one of these two cases:

Case 1: Be “non-ambitious" and turn down any “beyond" clauses. Here we will only assume $B_i \leq B_{i-1}$ and $|S_i| \leq |S_{i-1}|$.

Proof sketch Fix $k = 3$

Any step of the algorithm will fit one of these two cases:

Case 1: Be “non-ambitious" and turn down any “beyond" clauses. Here we will only assume $B_i \leq B_{i-1}$ and $|S_i| \leq |S_{i-1}|$.

Case 2: Accept all clauses.

Proof sketch Fix $k = 3$

Any step of the algorithm will fit one of these two cases:

Case 1: Be “non-ambitious" and turn down any “beyond" clauses. Here we will only assume $B_i \leq B_{i-1}$ and $|S_i| \leq |S_{i-1}|$.

Case 2: Accept all clauses. $B_i \leq B_{i-1} + \frac{1}{8}$ is always required.

Proof sketch Fix $k = 3$

Any step of the algorithm will fit one of these two cases:

Case 1: Be "non-ambitious" and turn down any "beyond" clauses. Here we will only assume $B_i \leq B_{i-1}$ and $|S_i| \leq |S_{i-1}|$.

Case 2: Accept all clauses. $B_i \leq B_{i-1} + \frac{1}{8}$ is always required. $E\,[\ |S_i|\ ] \;\leq\; \frac{7}{8}|S_{i-1}|$ comes from Jensen's Inequality.

Proof sketch Fix $k = 3$

Any step of the algorithm will fit one of these two cases:

Case 1: Be “non-ambitious" and turn down any “beyond" clauses. Here we will only assume $B_i \leq B_{i-1}$ and $|S_i| \leq |S_{i-1}|$.

Case 2: Accept all clauses. $B_i \leq B_{i-1} + \frac{1}{8}$ is always required. $E\,[\;|S_i|\;] \;\leq\; \frac{7}{8}|S_{i-1}|$ comes from Jensen’s Inequality.

Define $Y_i \;:=\; \ln|S_i| - 2^3 \ln(1 - \frac{1}{8})B_i$.

Proof sketch Fix $k = 3$

Any step of the algorithm will fit one of these two cases:

Case 1: Be “non-ambitious" and turn down any “beyond" clauses. Here we will only assume $B_i \leq B_{i-1}$ and $|S_i| \leq |S_{i-1}|$.

Case 2: Accept all clauses. $B_i \leq B_{i-1} + \frac{1}{8}$ is always required. $E\,[\,|S_i|\,] \;\leq\; \frac{7}{8}|S_{i-1}|$ comes from Jensen’s Inequality.

Define $Y_i \;:=\; \ln|S_i| - 2^3 \ln(1 - \frac{1}{8})B_i$.

Note: We have $E[Y_i] \leq Y_{i-1}$ in either case.

Proof sketch Fix $k = 3$

Any step of the algorithm will fit one of these two cases:

Case 1: Be “non-ambitious" and turn down any “beyond" clauses. Here we will only assume $B_i \leq B_{i-1}$ and $|S_i| \leq |S_{i-1}|$.

Case 2: Accept all clauses. $B_i \leq B_{i-1} + \frac{1}{8}$ is always required. $E[\,|S_i|\,] \leq \frac{7}{8}|S_{i-1}|$ comes from Jensen’s Inequality.

Define $Y_i := \ln|S_i| - 2^3 \ln(1 - \frac{1}{8})B_i$.

Note: We have $E[Y_i] \leq Y_{i-1}$ in either case.

This means $Y_{cn} \leq Y_0 + o(n)$ is true **whp**.

Proof sketch Fix $k = 3$

$$Y_i := \ln|S_i| - 2^3 \ln(1 - \frac{1}{8}) B_i.$$

Proof sketch Fix $k = 3$

$$Y_i \;:=\; \ln|S_i| - 2^3 \ln(1 - \tfrac{1}{8})B_i.$$

$B_0 = 0$ and $|S_0| = 2^n \quad \Rightarrow \quad Y_0 = n \ln 2.$

Proof sketch Fix $k = 3$

$$Y_i \;:=\; \ln|S_i| - 2^3 \ln(1 - \tfrac{1}{8}) B_i.$$

$B_0 = 0$ and $|S_0| = 2^n \quad \Rightarrow \quad Y_0 = n \ln 2.$

$|S_{cn}| \geq 1$

Proof sketch Fix $k = 3$

$$Y_i \ := \ \ln|S_i| - 2^3 \ln(1 - \tfrac{1}{8}) B_i.$$

$B_0 = 0$ and $|S_0| = 2^n \quad \Rightarrow \quad Y_0 = n \ln 2.$

$$|S_{cn}| \geq 1 \quad \Rightarrow \quad Y_{cn} \ \geq \ 2^3 \ln(1 - \tfrac{1}{8}) B_{cn}$$

Proof sketch Fix $k = 3$

$$Y_i \;:=\; \ln|S_i| - 2^3 \ln(1 - \tfrac{1}{8}) B_i.$$

$B_0 = 0$ and $|S_0| = 2^n \quad \Rightarrow \quad Y_0 = n \ln 2.$

$$|S_{cn}| \geq 1 \;\;\Rightarrow\;\; Y_{cn} \;\geq\; 2^3 \ln(1 - \tfrac{1}{8}) B_{cn} \;\geq\; B_{cn}.$$

Proof sketch Fix $k = 3$

$$Y_i \; := \; \ln|S_i| - 2^3 \ln(1 - \tfrac{1}{8}) B_i.$$

$B_0 = 0$ and $|S_0| = 2^n \quad \Rightarrow \quad Y_0 = n \ln 2.$

$$|S_{cn}| \geq 1 \;\; \Rightarrow \;\; Y_{cn} \; \geq \; 2^3 \ln(1 - \tfrac{1}{8}) B_{cn} \; \geq \; B_{cn}.$$

$$B_{cn} \; \leq \; Y_{cn} \; \overset{\mathbf{whp}}{\leq} \; Y_0 + o(n) \; = \; n \ln 2 + o(n).$$

Proof sketch Fix $k = 3$

$$Y_i \;:=\; \ln|S_i| - 2^3 \ln(1 - \tfrac{1}{8}) B_i.$$

$B_0 = 0$ and $|S_0| = 2^n \quad \Rightarrow \quad Y_0 = n \ln 2.$

$$|S_{cn}| \geq 1 \;\Rightarrow\; Y_{cn} \;\geq\; 2^3 \ln(1 - \tfrac{1}{8}) B_{cn} \;\geq\; B_{cn}.$$

$$B_{cn} \;\leq\; Y_{cn} \;\overset{\textbf{whp}}{\leq}\; Y_0 + o(n) \;=\; n \ln 2 + o(n).$$

$\square$

More about offline

More about offline

Suppose $k = 2$.

More about offline

Suppose $k = 2$.

Theorem: (Chvátal, Reed, 1992)
$c = 1$ is the threshold for satisfiability of $F_2(cn)$.

More about offline

Suppose $k = 2$.

Theorem: (Chvátal, Reed, 1992)
$c = 1$ is the threshold for satisfiability of $F_2(cn)$.

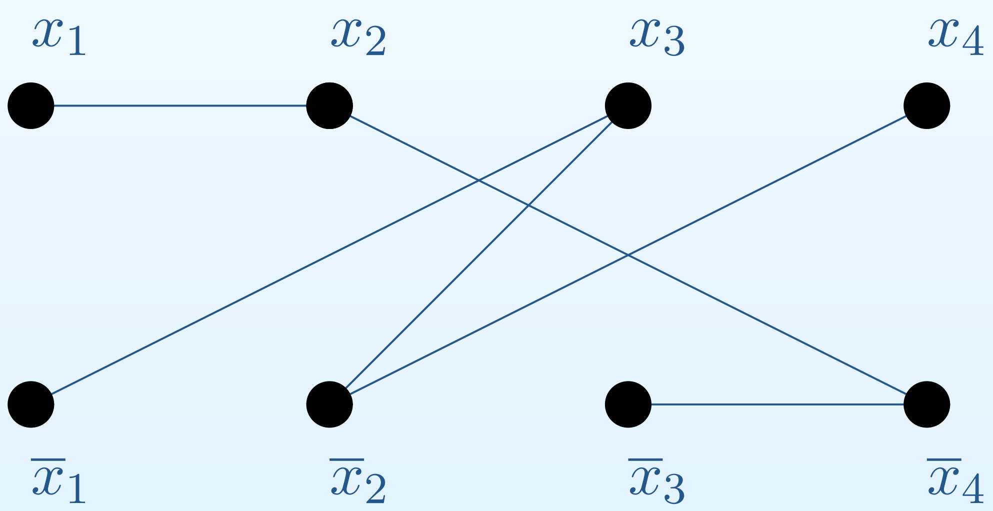
$$\{x_1, x_2\} \quad \{\overline{x}_2, x_4\} \quad \{\overline{x}_1, x_3\} \quad \{\overline{x}_2, x_3\} \quad \{x_2, \overline{x}_4\} \quad \{\overline{x}_3, \overline{x}_4\}$$

More about offline

Suppose $k = 2$.

Theorem: (Chvátal, Reed, 1992)
$c = 1$ is the threshold for satisfiability of $F_2(cn)$.

$$\{x_1, x_2\} \quad \{\overline{x}_2, x_4\} \quad \{\overline{x}_1, x_3\} \quad \{\overline{x}_2, x_3\} \quad \{x_2, \overline{x}_4\} \quad \{\overline{x}_3, \overline{x}_4\}$$

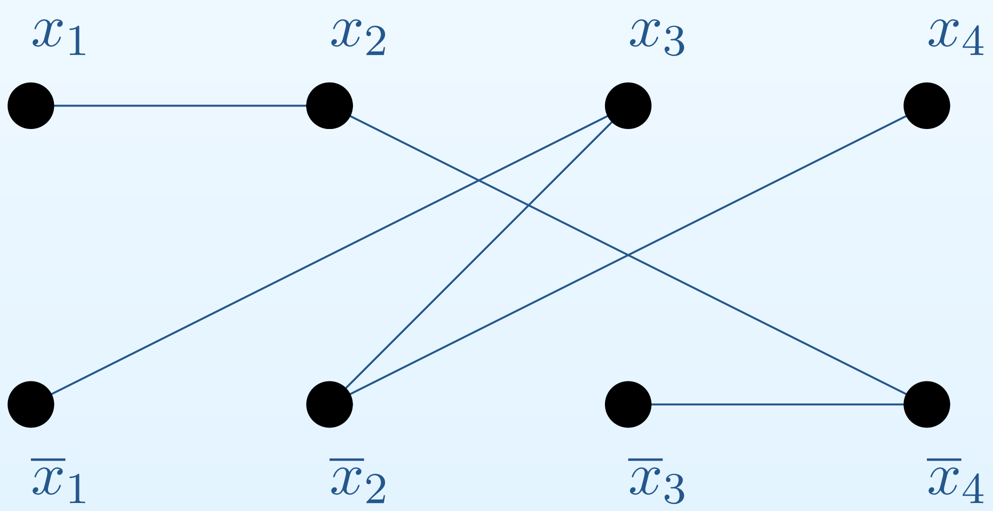


More about offline

Suppose $k = 2$.

Theorem: (Chvátal, Reed, 1992)
$c = 1$ is the threshold for satisfiability of $F_2(cn)$.

$\{x_1, x_2\}$ $\{\overline{x}_2, x_4\}$ $\{\overline{x}_1, x_3\}$ $\{\overline{x}_2, x_3\}$ $\{x_2, \overline{x}_4\}$ $\{\overline{x}_3, \overline{x}_4\}$



$2n$ vertices

cn edges

More about offline

Suppose $k = 2$.

Theorem: (Chvátal, Reed, 1992)
$c = 1$ is the threshold for satisfiability of $F_2(cn)$.

Theorem: (Erdős, Rényi, 1960)
$c = 1$ is the threshold for appearance of a giant component in the random graph with $2n$ vertices and cn edges.

More about offline

Suppose $k = 2$.

Theorem: (Chvátal, Reed, 1992)
$c = 1$ is the threshold for satisfiability of $F_2(cn)$.

Theorem: (Erdős, Rényi, 1960)
$c = 1$ is the threshold for appearance of a giant component in the random graph with $2n$ vertices and cn edges.

Question: Is there a correlation?

Our Model

Our Model

- Let G be *any* simple graph with $2n$ vertices and cn edges.

Our Model

- Let G be *any* simple graph with $2n$ vertices and cn edges.
- Make a family of clauses $S(G)$ by randomly assigning $\{x_1, \overline{x}_1, x_2, \overline{x}_2, \ldots, x_n, \overline{x}_n\}$ to the $2n$ vertices, so each edge in G corresponds to one clause.

Our Model

- Let G be *any* simple graph with $2n$ vertices and cn edges.
- Make a family of clauses $S(G)$ by randomly assigning $\{x_1, \overline{x}_1, x_2, \overline{x}_2, \ldots, x_n, \overline{x}_n\}$ to the $2n$ vertices, so each edge in G corresponds to one clause.
- We would like to know the probability that $S(G)$ is satisfiable over the space of all possible assignments to the vertices.

Our Model

- Let G be *any* simple graph with $2n$ vertices and cn edges.
- Make a family of clauses $S(G)$ by randomly assigning $\{x_1, \overline{x}_1, x_2, \overline{x}_2, \ldots, x_n, \overline{x}_n\}$ to the $2n$ vertices, so each edge in G corresponds to one clause.
- We would like to know the probability that $S(G)$ is satisfiable over the space of all possible assignments to the vertices.
- This question is equivalent to Random $2-SAT$ with n variables and cn clauses if G is a random graph, but we allow G to be *anything*, provided $\Delta(G)$ isn't too large.

Our Model

- Let G be *any* simple graph with $2n$ vertices and cn edges.
- Make a family of clauses $S(G)$ by randomly assigning $\{x_1, \overline{x}_1, x_2, \overline{x}_2, \ldots, x_n, \overline{x}_n\}$ to the $2n$ vertices, so each edge in G corresponds to one clause.
- We would like to know the probability that $S(G)$ is satisfiable over the space of all possible assignments to the vertices.
- This question is equivalent to Random $2 - SAT$ with n variables and cn clauses if G is a random graph, but we allow G to be *anything*, provided $\Delta(G)$ isn't too large.

Notation: For any graph G, $\Delta(G)$ is the maximum degree and $d_i(G)$ is the number of vertices of degree i ($i \geq 0$).

There is no correlation!

There is no correlation!

Suppose G is a graph with $2n$ vertices and $\epsilon > 0$ is any constant.

There is no correlation!

Suppose G is a graph with $2n$ vertices and $\epsilon > 0$ is any constant.

Theorem: If G has less than $(1-\epsilon)n$ edges and $\Delta(G) = o(\frac{n^{1/10}}{\log n})$, then $S(G)$ is satisfiable **whp**.

There is no correlation!

Suppose G is a graph with $2n$ vertices and $\epsilon > 0$ is any constant.

Theorem: If G has less than $(1-\epsilon)n$ edges and $\Delta(G) = o(\frac{n^{1/10}}{\log n})$, then $S(G)$ is satisfiable **whp**.

Theorem: If $\Delta(G) = o(n^{1/8})$ and there is some function $\tau = o(\log n)$ such that

$$\sum_{i=0}^{\tau} i d_i(G) = (1+\epsilon)2n$$

then $S(G)$ is not satisfiable **whp**.

There is no correlation!

Suppose G is a graph with $2n$ vertices and $\epsilon > 0$ is any constant.

Theorem: If G has less than $(1-\epsilon)n$ edges and $\Delta(G) = o(\frac{n^{1/10}}{\log n})$, then $S(G)$ is satisfiable **whp**.

Theorem: If $\Delta(G) = o(n^{1/8})$ and there is some function $\tau = o(\log n)$ such that

$$\sum_{i=0}^{\tau} i d_i(G) = (1+\epsilon)2n$$

then $S(G)$ is not satisfiable **whp**.

Conjecture: If G has more than $(1+\epsilon)n$ edges then there exists ϕ such that if $\Delta(G) = o(n^{\phi})$ then $S(G)$ is not satisfiable **whp**.

Questions for the Future

Questions for the Future

- Find the largest z_k for which there exists an online algorithm that accepts $(1 - \frac{1}{2^k})cn + z_k n$ out of cn clauses for any $k > 1$.

Questions for the Future

- Find the largest z_k for which there exists an online algorithm that accepts $(1 - \frac{1}{2^k})cn + z_k n$ out of cn clauses for any $k > 1$.
  - When $k = 1$ a greedy algorithm is easily seen to be optimal.

Questions for the Future

- Find the largest z_k for which there exists an online algorithm that accepts $(1 - \frac{1}{2^k})cn + z_k n$ out of cn clauses for any $k > 1$.
  - When $k = 1$ a greedy algorithm is easily seen to be optimal.
  - We have z_2 between 0.453 and 0.624.

Questions for the Future

- Find the largest z_k for which there exists an online algorithm that accepts $(1 - \frac{1}{2^k})cn + z_k n$ out of cn clauses for any $k > 1$.
  - When $k = 1$ a greedy algorithm is easily seen to be optimal.
  - We have z_2 between 0.453 and 0.624.
  - As k gets large, $z_k \to \ln 2$.

Questions for the Future

- Find the largest z_k for which there exists an online algorithm that accepts $(1 - \frac{1}{2^k})cn + z_k n$ out of cn clauses for any $k > 1$.
  - When $k = 1$ a greedy algorithm is easily seen to be optimal.
  - We have z_2 between 0.453 and 0.624.
  - As k gets large, $z_k \to \ln 2$.
- Does satisfiability in *online* $2 - SAT$ correspond with the appearance of a giant component in the corresponding $2n$-vertex graph?

Questions for the Future

- Find the largest z_k for which there exists an online algorithm that accepts $(1 - \frac{1}{2^k})cn + z_k n$ out of cn clauses for any $k > 1$.
  - When $k = 1$ a greedy algorithm is easily seen to be optimal.
  - We have z_2 between 0.453 and 0.624.
  - As k gets large, $z_k \to \ln 2$.
- Does satisfiability in *online* $2 - SAT$ correspond with the appearance of a giant component in the corresponding $2n$-vertex graph?
- When $c = 1 + \epsilon$, the number of rejected clauses is somewhere between $O(\epsilon)n$ and $O(\epsilon^3)n$. Where is the truth?

Questions for the Future

- Find the largest z_k for which there exists an online algorithm that accepts $(1 - \frac{1}{2^k})cn + z_k n$ out of cn clauses for any $k > 1$.
  - When $k = 1$ a greedy algorithm is easily seen to be optimal.
  - We have z_2 between 0.453 and 0.624.
  - As k gets large, $z_k \longrightarrow \ln 2$.
- Does satisfiability in *online* $2 - SAT$ correspond with the appearance of a giant component in the corresponding $2n$-vertex graph?
- When $c = 1 + \epsilon$, the number of rejected clauses is somewhere between $O(\epsilon)n$ and $O(\epsilon^3)n$. Where is the truth?
- Find the constant for Random $3 - SAT$, if it exists.

Questions for the Future

- Find the largest z_k for which there exists an online algorithm that accepts $(1 - \frac{1}{2^k})cn + z_k n$ out of cn clauses for any $k > 1$.
  - When $k = 1$ a greedy algorithm is easily seen to be optimal.
  - We have z_2 between 0.453 and 0.624.
  - As k gets large, $z_k \to \ln 2$.
- Does satisfiability in *online* $2 - SAT$ correspond with the appearance of a giant component in the corresponding $2n$-vertex graph?
- When $c = 1 + \epsilon$, the number of rejected clauses is somewhere between $O(\epsilon)n$ and $O(\epsilon^3)n$. Where is the truth?
- Find the constant for Random $3 - SAT$, if it exists.
- Solve the conjecture!

Thank you!

Thank you!

I will stop now.

Thank you!

I will stop now.

You're welcome.

Thank you!

I will stop now.

You're welcome.

Are there any questions?