Computational Complexity 3:
Phase transitions in physics and computer science

Cristopher Moore
Santa Fe Institute
Magnetism

Curie: when cold enough, iron will stay magnetized, and even magnetize spontaneously

Above a critical temperature, it suddenly ceases to be magnetic

Interactions between atoms remain the same, but global behavior changes!

Like water freezing, outbreaks becoming epidemics, opinions changing...
The Ising model
The Ising model

Lattice (e.g. square) with $n$ sites
The Ising model

Lattice (e.g. square) with $n$ sites

Each has a “spin” $s_i = \pm 1$, “up” or “down”
The Ising model

Lattice (e.g. square) with $n$ sites

Each has a “spin” $s_i=\pm 1$, “up” or “down”

Energy is a sum over neighboring pairs: $E = -\sum_{ij} s_is_j$
The Ising model

Lattice (e.g. square) with \( n \) sites

Each has a “spin” \( s_i = \pm 1 \), “up” or “down”

Energy is a sum over neighboring pairs: 

\[
E = - \sum_{ij} s_i s_j
\]

Lowest energy: all up or all down
The Ising model

Lattice (e.g. square) with $n$ sites

Each has a “spin” $s_i = \pm 1$, “up” or “down”

Energy is a sum over neighboring pairs: $E = - \sum_{ij} s_i s_j$

Lowest energy: all up or all down

Highest energy: checkerboard
Boltzmann Distribution
Boltzmann Distribution

At thermodynamic equilibrium, temperature $T$
Boltzmann Distribution

At thermodynamic equilibrium, temperature $T$

Higher-energy states are less likely:

$$ P(s) \sim e^{-E(s)/T} $$
Boltzmann Distribution

At thermodynamic equilibrium, temperature $T$

Higher-energy states are less likely:

$$P(s) \sim e^{-E(s)/T}$$

When $T \to 0$, only lowest energies appear
Boltzmann Distribution

At thermodynamic equilibrium, temperature $T$

Higher-energy states are less likely:

$$P(s) \sim e^{-E(s)/T}$$

When $T \to 0$, only lowest energies appear

When $T \to \infty$, all states are equally likely
Monte Carlo
Monte Carlo

At each step, choose a random site $i$
Monte Carlo

At each step, choose a random site $i$

Compute the energy change $\Delta E$ of flipping $s_i$
Monte Carlo

At each step, choose a random site $i$

Compute the energy change $\Delta E$ of flipping $s_i$

Metropolis rule:
Monte Carlo

At each step, choose a random site $i$.

Compute the energy change $\Delta E$ of flipping $s_i$.

Metropolis rule:

If $\Delta E < 0$, flip $s_i$.
Monte Carlo

At each step, choose a random site $i$

Compute the energy change $\Delta E$ of flipping $s_i$

Metropolis rule:

If $\Delta E < 0$, flip $s_i$

If $\Delta E > 0$, flip $s_i$ with probability $e^{-\Delta E/T}$
Monte Carlo

At each step, choose a random site $i$

Compute the energy change $\Delta E$ of flipping $s_i$

Metropolis rule:

- If $\Delta E < 0$, flip $s_i$
- If $\Delta E > 0$, flip $s_i$ with probability $e^{-\Delta E/T}$

Keep going until we reach equilibrium (how long will that take?)
Monte Carlo

At each step, choose a random site $i$

Compute the energy change $\Delta E$ of flipping $s_i$

Metropolis rule:

- If $\Delta E < 0$, flip $s_i$
- If $\Delta E > 0$, flip $s_i$ with probability $e^{-\Delta E/T}$

Keep going until we reach equilibrium (how long will that take?)

Let’s watch...
What Happens

Below critical temperature, if $T<T_c$, the system “magnetizes”: mostly up or mostly down

Small islands of the minority; as $T$ increases, these islands grow

If $T>T_c$, at large scales, equal numbers of up and down

When $T=T_c$, islands of all scales: system is scale-invariant!
**Scale Invariance at Criticality**

$T = 0.997T_c$  
$T = T_c$  
$T = 1.003T_c$

- At $T_c$ the correlation length diverges
- Clusters of all sizes appear in the configuration
- System becomes self-similar at different scales
**Scale Invariance at Criticality**

\[ T = 0.997T_c \quad T = T_c \quad T = 1.003T_c \]

- At \( T_c \) the correlation length diverges
- Clusters of all sizes appear in the configuration
- System becomes self-similar at different scales

Douglas Ashton
Universality at the Critical Point

- Universality: Seemingly unrelated systems share universal properties near the critical point.

Douglas Ashton

Ising Model

Lennard-Jones

\[ V(r) = \frac{1}{r^{12}} - \frac{1}{r^6} \]
Universality at the Critical Point

- Universality: Seemingly unrelated systems share universal properties near the critical point

Douglas Ashton
Even more symmetry

Conformal invariance: any analytic map of the complex plane
Mean Field Approximation
Mean Field Approximation

Ignore topology: forget lattice structure
Mean Field Approximation

Ignore topology: forget lattice structure

If $a$ of the sites are up and $1-a$ are down, energy is $E = 2n^2 \left( 2a(1-a) - a^2 - (1-a)^2 \right)$
Mean Field Approximation

Ignore topology: forget lattice structure

If \( a \) of the sites are up and \( 1-a \) are down, energy is

\[
E = 2n^2 \left( 2a(1-a) - a^2 - (1-a)^2 \right)
\]

At any \( T \), most-likely states have \( a=0 \) or \( a=1 \)
Mean Field Approximation

Ignore topology: forget lattice structure

If $a$ of the sites are up and $1-a$ are down, energy is $E = 2n^2\left(2a(1-a) - a^2 - (1-a)^2\right)$

At any $T$, most-likely states have $a=0$ or $a=1$

But the number of such states is $\binom{n}{an}$, which is tightly peaked around $a=1/2$. 
Mean Field Approximation

Ignore topology: forget lattice structure

If $a$ of the sites are up and $1-a$ are down, energy is $E = 2n^2 \left( 2a(1-a) - a^2 - (1-a)^2 \right)$

At any $T$, most-likely states have $a=0$ or $a=1$

But the number of such states is $\binom{n}{a n}$, which is tightly peaked around $a=1/2$.

Total probability($a$) = #states($a$) Boltzmann($a$)
Energy vs. Entropy

$T=5$
Energy vs. Entropy

$T=3$
Correlations
Correlations

\[ C(r) = \text{correlation between two sites } r \text{ apart} \]
Correlations

\[ C(r) = \text{correlation between two sites } r \text{ apart} \]

If \( T > T_c \) correlations decay exponentially:

\[ C(r) \sim e^{-r/\ell} \]
Correlations

\[ C(r) = \text{correlation between two sites } r \text{ apart} \]

If \( T > T_c \) correlations decay exponentially:

\[ C(r) \sim e^{-r/\ell} \]

Correlation length \( \ell \) decreases as \( T \) grows
Correlations

\[ C(r) = \text{correlation between two sites } r \text{ apart} \]

If \( T > T_c \) correlations decay exponentially:

\[ C(r) \sim e^{-r/\ell} \]

Correlation length \( \ell \) decreases as \( T \) grows.

As we approach \( T_c \) correlation length diverges.
Correlations

$C(r) = \text{correlation between two sites } r \text{ apart}$

If $T > T_c$ correlations decay exponentially:

$$C(r) \sim e^{-r/\ell}$$

Correlation length $\ell$ decreases as $T$ grows

As we approach $T_c$ correlation length diverges

At $T=T_c$ power-law correlations (scale-free):

$$C(r) \sim r^{-\alpha}$$
Percolation

Fill a fraction $p$ of the sites in a lattice
Percolation

Fill a fraction $p$ of the sites in a lattice

When $p < p_c$, small islands, whose size is exponentially distributed:

$$P(s) \sim e^{-s/\bar{s}}$$
Percolation

Fill a fraction $p$ of the sites in a lattice

When $p < p_c$, small islands, whose size is exponentially distributed:

$$P(s) \sim e^{-s/\bar{s}}$$

When $p > p_c$ a unique “giant cluster” appears
Percolation

Fill a fraction $p$ of the sites in a lattice

When $p < p_c$, small islands, whose size is exponentially distributed:

$$P(s) \sim e^{-s/\bar{s}}$$

When $p > p_c$ a unique “giant cluster” appears

At $p=p_c$ power-law distribution of cluster sizes:

$$P(s) \sim s^{-\alpha}$$
Percolation

- PHY 411/506 Java Applet: Percolation
Phase transitions in NP-complete problems
Phase transitions in NP-complete problems

NP-completeness is a worst-case notion
Phase transitions in NP-complete problems

NP-completeness is a worst-case notion

3-SAT is hard because hard instances exist...
Phase transitions in NP-complete problems

NP-completeness is a worst-case notion

3-SAT is hard because hard instances exist...

...and we assume instances are designed by a clever adversary (cruel world!)
Phase transitions in NP-complete problems

NP-completeness is a worst-case notion

3-SAT is hard because hard instances exist...

...and we assume instances are designed by a clever adversary (cruel world!)

What if the constraints are chosen randomly instead?
Phase transitions in NP-complete problems

NP-completeness is a worst-case notion

3-SAT is hard because hard instances exist...

...and we assume instances are designed by a clever adversary (cruel world!)

What if the constraints are chosen randomly instead?

As we add more constraints, more contradictions arise...
Random SAT problems
Random SAT problems

A 3-SAT problem with $n$ variables, $m$ clauses
Random SAT problems

A 3-SAT problem with $n$ variables, $m$ clauses

For each clause, choose a random triplet of variables, and negate each one with probability 1/2
Random SAT problems

A 3-SAT problem with $n$ variables, $m$ clauses

For each clause, choose a random triplet of variables, and negate each one with probability $1/2$

Sparse case: $m=\alpha n$ for some constant density $\alpha$
Random SAT problems

A 3-SAT problem with $n$ variables, $m$ clauses

For each clause, choose a random triplet of variables, and negate each one with probability 1/2

Sparse case: $m = \alpha n$ for some constant density $\alpha$

Analogous to spin glasses and random graphs
A transition from solvability to unsolvability

When there are too many constraints, we can’t satisfy all of them
Where the hard problems are

Search times are highest at the boundary

![Graph showing DPPL calls vs. α](image-url)
The Threshold Conjecture

We believe that for each $k \geq 3$ there is a critical clause density $\alpha_k$ such that

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

Until recently, only known rigorously for $k=2$
The Threshold Conjecture

We believe that for each \( k \geq 3 \) there is a critical clause density \( \alpha_k \) such that

\[
\lim_{n \to \infty} \Pr [F_k(n, m = \alpha n) \text{ is satisfiable}] = \begin{cases} 
1 & \text{if } \alpha < \alpha_k \\
0 & \text{if } \alpha > \alpha_k 
\end{cases}
\]

Until recently, only known rigorously for \( k = 2 \)
The Threshold Conjecture

We believe that for each $k \geq 3$ there is a critical clause density $\alpha_k$ such that

$$\lim_{n \to \infty} \Pr \left[ F_k(n, m = \alpha n) \text{ is satisfiable} \right] = \begin{cases} 1 & \text{if } \alpha < \alpha_k \\ 0 & \text{if } \alpha > \alpha_k \end{cases}$$

Until recently, only known rigorously for $k=2$

[for large enough $k$: Ding, Sly, Sun 2014]
An Upper Bound
An Upper Bound

The average number of solutions is

\[ 2^n \left( \frac{7}{8} \right)^m = \left( 2 \left( \frac{7}{8} \right)^\alpha \right)^n \]
An Upper Bound

The average number of solutions is

\[ 2^n \left( \frac{7}{8} \right)^m = \left( 2 \left( \frac{7}{8} \right)^\alpha \right)^n \]

This is exponentially small whenever

\[ \alpha > \log_{8/7} 2 \approx 5.19 \]
An Upper Bound

The average number of solutions is

$$2^n \left( \frac{7}{8} \right)^m = \left( 2 \left( \frac{7}{8} \right)^\alpha \right)^n$$

This is exponentially small whenever

$$\alpha > \log_{8/7} 2 \approx 5.19$$

But the transition is much lower, at $\alpha \approx 4.27$. What’s going on?
A Heavy Tail
A Heavy Tail

In the range $4.27 < \alpha < 5.19$, the average number of solutions is exponentially large.
A Heavy Tail

In the range $4.27 < \alpha < 5.19$, the average number of solutions is exponentially large.

Occasionally, there are exponentially many...
A Heavy Tail

In the range $4.27 < \alpha < 5.19$, the average number of solutions is exponentially large.

Occasionally, there are exponentially many...

...but most of the time there are none!
A Heavy Tail

In the range $4.27 < \alpha < 5.19$, the average number of solutions is exponentially large.

Occasionally, there are exponentially many...

...but most of the time there are none!

A classic “heavy-tailed” distribution
A Heavy Tail

In the range $4.27 < \alpha < 5.19$, the average number of solutions is exponentially large.

Occasionally, there are exponentially many...

...but most of the time there are none!

A classic “heavy-tailed” distribution

Large average doesn’t prove satisfiability!
Lower Bound #1
Lower Bound #1

Idea: track the progress of a simple algorithm!
Lower Bound #1

Idea: track the progress of a simple algorithm!

When we set variables, clauses disappear or get shorter:

$$\overline{x} \land (x \lor y \lor z) \Rightarrow (y \lor z)$$
Lower Bound #1

Idea: track the progress of a simple algorithm!

When we set variables, clauses disappear or get shorter:

\[ \overline{x} \land (x \lor y \lor z) \Rightarrow (y \lor z) \]

*Unit Clauses* propagate:

\[ x \land (\overline{x} \lor y) \Rightarrow y \]
One Path Through the Tree
One Path Through the Tree

If there is a unit clause, satisfy it.
One Path Through the Tree

If there is a unit clause, satisfy it.

Otherwise, choose a random variable and give it a random value!
One Path Through the Tree

If there is a unit clause, satisfy it.

Otherwise, choose a random variable and give it a random value!

The remaining formula is random. We just need to keep track of densities of 2- and 3-clauses:
One Path Through the Tree

If there is a unit clause, satisfy it.

Otherwise, choose a random variable and give it a random value!

The remaining formula is random. We just need to keep track of densities of 2- and 3-clauses:

$$\frac{ds_3}{dt} = -\frac{3s_3}{1-t} \quad , \quad \frac{ds_2}{dt} = \frac{(3/2)s_3 - 2s_2}{1-t}$$
One Path Through the Tree

If there is a unit clause, satisfy it.

Otherwise, choose a random variable and give it a random value!

The remaining formula is random. We just need to keep track of densities of 2- and 3-clauses:

\[
\begin{align*}
    \frac{ds_3}{dt} &= -\frac{3s_3}{1 - t}, \\
    \frac{ds_2}{dt} &= \frac{(3/2)s_3 - 2s_2}{1 - t}
\end{align*}
\]

\[s_3(0) = \alpha, \quad s_2(0) = 0\]
These differential equations give

\[ s_3(t) = \alpha (1 - t)^3 \]

\[ s_2(t) = \frac{3}{2} \alpha t (1 - t)^2 \]
Branching Unit Clauses
Branching Unit Clauses

Each unit clause has on average $\lambda$ children, where

$$\lambda = \frac{1}{2} \frac{2s_2}{1 - t} = \frac{3}{4} \alpha t (1 - t)$$
Branching UnitClauses

Each unit clause has on average $\lambda$ children, where

$$\lambda = \frac{1}{2} \frac{2s_2}{1 - t} = \frac{3}{4} \alpha t (1 - t)$$

If $\lambda > 1$ an epidemic of contradictions
Branching Unit Clauses

Each unit clause has on average $\lambda$ children, where

$$\lambda = \frac{1}{2} \frac{2s_2}{1 - t} = \frac{3}{4} \alpha t(1 - t)$$

If $\lambda > 1$ an epidemic of contradictions

Maximized at $t = 1/2$
Branching Unit Clauses

Each unit clause has on average $\lambda$ children, where

$$\lambda = \frac{1}{2} \frac{2s_2}{1 - t} = \frac{3}{4} \alpha t (1 - t)$$

If $\lambda > 1$ an epidemic of contradictions

Maximized at $t = 1/2$

If $\alpha < 8/3$ then $\lambda < 1$ always, and the unit clauses stay manageable
Branching Unit Clauses

Each unit clause has on average $\lambda$ children, where

$$\lambda = \frac{1}{2} \frac{2s_2}{1 - t} = \frac{3}{4} \alpha t (1 - t)$$

If $\lambda > 1$ an epidemic of contradictions

Maximized at $t = 1/2$

If $\alpha < 8/3$ then $\lambda < 1$ always, and the unit clauses stay manageable

Thus $8/3$ is a lower bound on the transition
Constructive Methods Fail
Constructive Methods Fail

Fancier algorithms, harder math: $\alpha < 3.52$
Constructive Methods Fail

Fancier algorithms, harder math: $\alpha < 3.52$

But for larger $k$, algorithmic methods are nowhere near the upper bound for $k$-SAT:

$$O\left(\frac{2^k}{k}\right) < \alpha < O(2^k)$$
Constructive Methods Fail

Fancier algorithms, harder math: $\alpha < 3.52$

But for larger $k$, algorithmic methods are nowhere near the upper bound for $k$-SAT:

$$O\left(\frac{2^k}{k}\right) < \alpha < O(2^k)$$

We can close this gap, but the proof is nonconstructive: existence, but no algorithm
Lower Bound #2
Lower Bound #2

Bound the *variance* of the number of solutions.
Bound the *variance* of the number of solutions.

If $X$ is a nonnegative random variable,

$$\Pr[X > 0] \geq \frac{E[X]^2}{E[X^2]}$$
Lower Bound #2

Bound the variance of the number of solutions.

If $X$ is a nonnegative random variable, 

$$\Pr[X > 0] \geq \frac{E[X]^2}{E[X^2]}$$

$E[X]$ is easy; $E[X^2]$ requires us to understand correlations between solutions.
Bound the variance of the number of solutions.

If $X$ is a nonnegative random variable,

$$\Pr[X > 0] \geq \frac{E[X]^2}{E[X^2]}$$

$E[X]$ is easy; $E[X^2]$ requires us to understand correlations between solutions.

Shows the threshold is $2^k \ln 2 - o(1)$
Clustering

Below the critical temperature, magnets have two macrostates (Gibbs measures)

Glasses, and 3-SAT, have exponentially many!
Clustering, freezing, and hardness

Figure 14.30: A refined phase diagram of random $k$-SAT. Gray blobs represent frozen clusters, i.e., those where $\Theta(n)$ variables take fixed values. Above $\alpha_{rigid}$ almost all clusters are frozen, and we believe this is responsible for the average-case hardness of random $k$-SAT.

But even if there are no variables that are frozen in all solutions, we could certainly have variables frozen within a cluster. Let's say that a variable $x_i$ is frozen in a cluster $C$ if $x_i$ takes the same value in every solution in $C$, a constant $\kappa n$ frozen variables for some constant $\kappa > 0$.

Intuitively, these frozen clusters spell doom for local algorithms. Imagine a DPLL algorithm descending into its search tree. With every variable it sets, it contradicts any cluster in which this variable is frozen with the opposite value. If every cluster is frozen, then it contradicts a constant fraction of them at each step, until it has excluded every cluster from the branch ahead. This forces it to backtrack, taking exponential time.

It's also worth noting that if the clusters are a Hamming distance $\delta n$ apart, then the DPLL algorithm is limited to a single cluster as soon as it reaches a certain depth in the search tree. Once it has set $(1-\delta)n$ variables, all the assignments on the resulting subtree are within a Hamming distance $\delta n$ of each other, so they can overlap with at most one cluster. If any of the variables it has already set are frozen in this cluster, and if it set any of them wrong, it is already doomed.

Recent rigorous results strongly suggest that this is exactly what's going on. In addition to the other properties of clusters established by Theorem 14.5 at densities above $(2k/k) \ln k$, one can show that almost all clusters have $\kappa n$ frozen variables for a constant $\kappa > n$. Specifically, if we choose a cluster with probability proportional to its size, then it has $\kappa n$ frozen variables with high probability. Equivalently, if we choose a uniformly random satisfying assignment, then with high probability there are $\kappa n$ variables on which it agrees with every other solution in its cluster.

Conversely, it can be shown that algorithms based on setting one variable at a time using BP messages fail in this frozen region. But in a recent breakthrough, an algorithm was discovered which works at densities up to $(1-\epsilon k)(2k/k) \ln k$ where $\epsilon k \to 0$ as $k \to \infty$. Thus for large $k$, it seems that algorithms end precisely where the frozen phase begins.

For large $k$, clustering and freezing take place at roughly the same density. In contrast, for small $k$ they are widely separated, which explains why some algorithms can probe deep into the clustered phase.

Recent rigorous results strongly suggest that this is exactly what's going on. In addition to the other properties of clusters established by Theorem 14.5 at densities above $(2k/k) \ln k$, one can show that almost all clusters have $\kappa n$ frozen variables for a constant $\kappa > n$. Specifically, if we choose a cluster with probability proportional to its size, then it has $\kappa n$ frozen variables with high probability. Equivalently, if we choose a uniformly random satisfying assignment, then with high probability there are $\kappa n$ variables on which it agrees with every other solution in its cluster.

Conversely, it can be shown that algorithms based on setting one variable at a time using BP messages fail in this frozen region. But in a recent breakthrough, an algorithm was discovered which works at densities up to $(1-\epsilon k)(2k/k) \ln k$ where $\epsilon k \to 0$ as $k \to \infty$. Thus for large $k$, it seems that algorithms end precisely where the frozen phase begins.

For large $k$, clustering and freezing take place at roughly the same density. In constrast, for small $k$ they are widely separated, which explains why some algorithms can probe deep into the clustered phase.
Clustering, freezing, and hardness

At a certain density, solutions break up into clusters.

Figure 14.30: A refined phase diagram of random $k$-SAT. Gray blobs represent frozen clusters, i.e., those where $\Theta(n)$ variables take fixed values. Above $\alpha_{\text{rigid}}$ almost all clusters are frozen, and we believe this is responsible for the average-case hardness of random $k$-SAT.

Clause renders this instance unsatisfiable with finite probability. Therefore we have $\alpha_c < \alpha_{\text{rigid}} + \epsilon$ for any $\epsilon > 0$, a contradiction.

But even if there are no variables that are frozen in all solutions, we could certainly have variables frozen within a cluster. Let's say that a variable $x_i$ is frozen in a cluster $C$ if $x_i$ takes the same value in every solution in $C$, and $C$ has $\kappa n$ frozen variables for some constant $\kappa > 0$.

Intuitively, these frozen clusters spell doom for local algorithms. Imagine a DPLL algorithm descending into its search tree. With every variable it sets, it contradicts any cluster in which this variable is frozen with the opposite value. If every cluster is frozen, then it contradicts a constant fraction of them at each step, until it has excluded every cluster from the branch ahead. This forces it to backtrack, taking exponential time.

It's also worth noting that if the clusters are a Hamming distance $\delta n$ apart, then the DPLL algorithm is limited to a single cluster as soon as it reaches a certain depth in the search tree. Once it has set $\left(1 - \delta\right)n$ variables, all the assignments on the resulting subtree are within a Hamming distance $\delta n$ of each other, so they can overlap with at most one cluster. If any of the variables it has already set are frozen in this cluster, and if it sets any of them wrong, it is already doomed.

Recent rigorous results strongly suggest that this is exactly what's going on. In addition to the other properties of clusters established by Theorem 14.5 at densities above $\left(\frac{2k}{k}\right)\ln k$, one can show that almost all clusters have $\kappa n$ frozen variables for a constant $\kappa > n$. Specifically, if we choose a cluster with probability proportional to its size, then it has $\kappa n$ frozen variables with high probability. Equivalently, if we choose a uniformly random satisfying assignment, then with high probability there are $\kappa n$ variables on which it agrees with every other solution in its cluster.

Conversely, it can be shown that algorithms based on setting one variable at a time using BP messages fail in this frozen region. But in a recent breakthrough, an algorithm was discovered which works at densities up to $\left(1 - \epsilon k\right)\left(\frac{2k}{k}\right)\ln k$ where $\epsilon_k \to 0$ as $k \to \infty$. Thus for large $k$, it seems that algorithms end precisely where the frozen phase begins.

For large $k$, clustering and freezing take place at roughly the same density. In contrast, for small $k$ they are widely separated, which explains why some algorithms can probe deep into the clustered phase.

Figure 14.30 shows a refined picture of random $k$-SAT that includes frozen clusters. The freezing transition is defined by the point $\alpha_{\text{rigid}}$ where the number of unfrozen clusters drops to zero.
Clustering, freezing, and hardness

At a certain density, solutions break up into clusters. These clusters become \textit{frozen} — many variables take a fixed value.

Figure 14.30: A refined phase diagram of random \(k\)-SAT. Gray blobs represent frozen clusters, i.e., those where \(\Theta(n)\) variables take fixed values. Above \(\alpha_{\text{rigid}}\) almost all clusters are frozen, and we believe this is responsible for the average-case hardness of random \(k\)-SAT.

But even if there are no variables that are frozen in all solutions, we could certainly have variables frozen within a cluster. Let's say that a variable \(x_i\) is frozen in a cluster \(C\) if \(x_i\) takes the same value in every solution in \(C\), and there are \(\kappa n\) frozen variables for some constant \(\kappa > 0\).

Intuitively, these frozen clusters spell doom for local algorithms. Imagine a DPLL algorithm descending into its search tree. With every variable it sets, it contradicts any cluster in which this variable is frozen with the opposite value. If every cluster is frozen, then it contradicts a constant fraction of them at each step, until it has excluded every cluster from the branch ahead. This forces it to backtrack, taking exponential time.

It's also worth noting that if the clusters are a Hamming distance \(\delta n\) apart, then the DPLL algorithm is limited to a single cluster as soon as it reaches a certain depth in the search tree. Once it has set \((1 - \delta) n\) variables, all the assignments on the resulting subtree are within a Hamming distance \(\delta n\) of each other, so they can overlap with at most one cluster. If any of the variables it has already set are frozen in this cluster, and if it set any of them wrong, it is already doomed.

Recent rigorous results strongly suggest that this is exactly what's going on. In addition to the other properties of clusters established by Theorem 14.5 at densities above \((2^k/k) \ln k\), one can show that almost all clusters have \(\kappa n\) frozen variables for a constant \(\kappa > n\). Specifically, if we choose a cluster with probability proportional to its size, then it has \(\kappa n\) frozen variables with high probability. Equivalently, if we choose a uniformly random satisfying assignment, then with high probability there are \(\kappa n\) variables on which it agrees with every other solution in its cluster.

Conversely, it can be shown that algorithms based on setting one variable at a time using BP messages fail in this frozen region. But in a recent breakthrough, an algorithm was discovered which works at densities up to \((1 - \epsilon k)(2^k/k) \ln k\) where \(\epsilon k \to 0\) as \(k \to \infty\). Thus for large \(k\), it seems that algorithms end precisely where the frozen phase begins.

For large \(k\), clustering and freezing take place at roughly the same density. In contrast, for small \(k\) they are widely separated, which explains why some algorithms can probe deep into the clustered phase.

Figure 14.30 shows a refined picture of random \(k\)-SAT that includes frozen clusters. The freezing transition is defined by the point \(\alpha_{\text{rigid}}\) where the number of unfrozen clusters drops to zero.
Clustering, freezing, and hardness

At a certain density, solutions break up into clusters

These clusters become *frozen* — many variables take a fixed value

If a search algorithm sets any of these variables wrong, it’s doomed
Clustering, freezing, and hardness

At a certain density, solutions break up into clusters

These clusters become *frozen* — many variables take a fixed value

If a search algorithm sets any of these variables wrong, it’s doomed

A rugged landscape, with many local optima to get stuck in

[Achlioptas, Coja-Oghlan, Krzakala, Mezard, Molloy, Montanari, Moore, Ricci-Tersenghi, Zdeborová, Zecchina...]

Figure 14.30: A refined phase diagram of random $k$-SAT. Gray blobs represent frozen clusters, i.e., those where $\Theta(n)$ variables take fixed values. Above $\alpha_{\text{rigid}}$ almost all clusters are frozen, and we believe this is responsible for the average-case hardness of random $k$-SAT.

Clause renders this instance unsatisfiable with finite probability. Therefore we have $\alpha_{c} < \alpha_{\text{rigid}} + \epsilon$ for any $\epsilon > 0$, a contradiction.

But even if there are no variables that are frozen in all solutions, we could certainly have variables frozen within a cluster. Let’s say that a variable $x_i$ is frozen in a cluster $C$ if $x_i$ takes the same value in every solution in $C$, and there are $\kappa n$ frozen variables for some constant $\kappa > 0$.

Intuitively, these frozen clusters spell doom for local algorithms. Imagine a DPLL algorithm descending into its search tree. With every variable it sets, it contradicts any cluster in which this variable is frozen with the opposite value. If every cluster is frozen, then it contradicts a constant fraction of them at each step, until it has excluded every cluster from the branch ahead. This forces it to backtrack, taking exponential time.

It’s also worth noting that if the clusters are a Hamming distance $\delta n$ apart, then the DPLL algorithm is limited to a single cluster as soon as it reaches a certain depth in the search tree. Once it has set $(1 - \delta) n$ variables, all the assignments on the resulting subtree are within a Hamming distance $\delta n$ of each other, so they can overlap with at most one cluster. If any of the variables it has already set are frozen in this cluster, and if it set any of them wrong, it is already doomed.

Recent rigorous results strongly suggest that this is exactly what’s going on. In addition to the other properties of clusters established by Theorem 14.5 at densities above $(2k/k) \ln k$, one can show that almost all clusters have $\kappa n$ frozen variables for a constant $\kappa > n$. Specifically, if we choose a cluster with probability proportional to its size, then it has $\kappa n$ frozen variables with high probability. Equivalently, if we choose a uniformly random satisfying assignment, then with high probability there are $\kappa n$ variables on which it agrees with every other solution in its cluster.

Conversely, it can be shown that algorithms based on setting one variable at a time using BP messages fail in this frozen region. But in a recent breakthrough, an algorithm was discovered which works at densities up to $(1 - \epsilon k)(2k/k) \ln k$ where $\epsilon k \to 0$ as $k \to \infty$. Thus for large $k$, it seems that algorithms end precisely where the frozen phase begins.

For large $k$, clustering and freezing take place at roughly the same density. In contrast, for small $k$ they are widely separated, which explains why some algorithms can probe deep into the clustered phase.

Figure 14.30 shows a refined picture of random $k$-SAT that includes frozen clusters. The freezing transition is defined by the point $\alpha_{\text{rigid}}$ where the number of unfrozen clusters drops to zero.
Clustering, freezing, and hardness

Figure 14.30: A refined phase diagram of random $k$-SAT. Gray blobs represent frozen clusters, i.e., those where $\Theta(n)$ variables take fixed values. Above $\alpha_{\text{rigid}}$ almost all clusters are frozen, and we believe this is responsible for the average-case hardness of random $k$-SAT. A clause renders this instance unsatisfiable with finite probability. Therefore we have $\alpha_c < \alpha_{\text{rigid}} + \epsilon$ for any $\epsilon > 0$, a contradiction.

But even if there are no variables that are frozen in all solutions, we could certainly have variables frozen within a cluster. Let’s say that a variable $x_i$ is frozen in a cluster $C$ if $x_i$ takes the same value in every solution in $C$, and there are $\kappa n$ frozen variables for some constant $\kappa > 0$.

Intuitively, these frozen clusters spell doom for local algorithms. Imagine a DPLL algorithm descending into its search tree. With every variable it sets, it contradicts any cluster in which this variable is frozen with the opposite value. If every cluster is frozen, then it contradicts a constant fraction of them at each step, until it has excluded every cluster from the branch ahead. This forces it to backtrack, taking exponential time.

It’s also worth noting that if the clusters are a Hamming distance $\delta n$ apart, then the DPLL algorithm is limited to a single cluster as soon as it reaches a certain depth in the search tree. Once it has set $(1 - \delta)n$ variables, all the assignments on the resulting subtree are within a Hamming distance $\delta n$ of each other, so they can overlap with at most one cluster. If any of the variables it has already set are frozen in this cluster, and if it set any of them wrong, it is already doomed.

Recent rigorous results strongly suggest that this is exactly what’s going on. In addition to the other properties of clusters established by Theorem 14.5 at densities above $(2k/k) \ln k$, one can show that almost all clusters have $\kappa n$ frozen variables for a constant $\kappa > n$. Specifically, if we choose a cluster with probability proportional to its size, then it has $\kappa n$ frozen variables with high probability. Equivalently, if we choose a uniformly random satisfying assignment, then with high probability there are $\kappa n$ variables on which it agrees with every other solution in its cluster.

Conversely, it can be shown that algorithms based on setting one variable at a time using BP messages fail in this frozen region. But in a recent breakthrough, an algorithm was discovered which works at densities up to $(1 - \epsilon_k) (2k/k) \ln k$ where $\epsilon_k \to 0$ as $k \to \infty$. Thus for large $k$, it seems that algorithms end precisely where the frozen phase begins.

For large $k$, clustering and freezing take place at roughly the same density. In contrast, for small $k$ they are widely separated, which explains why some algorithms can probe deep into the clustered phase.

Figure 14.30 shows a refined picture of random $k$-SAT that includes frozen clusters. The freezing transition is defined by the point $\alpha_{\text{rigid}}$ where the number of unfrozen clusters drops to zero.
Clustering, freezing, and hardness

The “freezing” transition marks where the problem becomes hard
Clustering, freezing, and hardness

The “freezing” transition marks where the problem becomes hard

All known algorithms for $k$-SAT stop working at $\alpha_{\text{rigid}} \sim 2^k \log k / k$
Clustering, freezing, and hardness

The “freezing” transition marks where the problem becomes hard.

All known algorithms for $k$-SAT stop working at $a_{\text{rigid}} \sim 2^k \log k / k$

Hard, but satisfiable, instances up to $a_c \sim 2^k \log 2$
The “freezing” transition marks where the problem becomes hard.

All known algorithms for $k$-SAT stop working at $a_{\text{rigid}} \sim 2^k \log k / k$.

Hard, but satisfiable, instances up to $a_c \sim 2^k \log 2$.

Can this be made into a proof that P\(\neq\)NP?
XORSAT
Use XOR (addition mod 2) instead of OR:

\[
\begin{align*}
    x_1 \oplus x_2 \oplus x_3 &= 1 \\
    x_1 \oplus x_2 \oplus x_4 &= 0 \\
    x_2 \oplus x_3 \oplus x_4 &= 1
\end{align*}
\]
Use XOR (addition mod 2) instead of OR:

\[ x_1 \oplus x_2 \oplus x_3 = 1 \]
\[ x_1 \oplus x_2 \oplus x_4 = 0 \]
\[ x_2 \oplus x_3 \oplus x_4 = 1 \]

Random instances have many of the same properties as 3-SAT: clustering and freezing (at the same density) and then a transition to unsatisfiability
XORSAT

Use XOR (addition mod 2) instead of OR:

\[ x_1 \oplus x_2 \oplus x_3 = 1 \]
\[ x_1 \oplus x_2 \oplus x_4 = 0 \]
\[ x_2 \oplus x_3 \oplus x_4 = 1 \]

Random instances have many of the same properties as 3-SAT: clustering and freezing (at the same density) and then a transition to unsatisfiability

But XORSAT is easy! Just linear equations:

\[
\begin{pmatrix}
1 & 1 & 1 & 0 \\
1 & 1 & 0 & 1 \\
0 & 1 & 1 & 1 \\
\end{pmatrix}
\begin{pmatrix}
x_1 \\
x_2 \\
x_3 \\
x_4 \\
\end{pmatrix}
=
\begin{pmatrix}
1 \\
0 \\
1 \\
\end{pmatrix}
\]
XORSAT
How is XORSAT like SAT, and how is it different?
XORSAT

How is XORSAT like SAT, and how is it different?

Clustering: local search algorithms can’t explore the space, and hill-climbing algorithms get stuck in local optima
How is XORSAT like SAT, and how is it different?

Clustering: local search algorithms can’t explore the space, and hill-climbing algorithms get stuck in local optima

Freezing: backtracking algorithms take exponential time, repeatedly setting frozen variables the wrong way
XORSAT

How is XORSAT like SAT, and how is it different?

Clustering: local search algorithms can’t explore the space, and hill-climbing algorithms get stuck in local optima

Freezing: backtracking algorithms take exponential time, repeatedly setting frozen variables the wrong way

But Gaussian elimination is a global change of variables, letting us turn a hard-looking problem into an easy one
XORSAT

How is XORSAT like SAT, and how is it different?

Clustering: local search algorithms can’t explore the space, and hill-climbing algorithms get stuck in local optima.

Freezing: backtracking algorithms take exponential time, repeatedly setting frozen variables the wrong way.

But Gaussian elimination is a global change of variables, letting us turn a hard-looking problem into an easy one.

If SAT has a similar kind of rearrangement — something totally different from backtracking or local search — then P=NP.
How is XORSAT like SAT, and how is it different?

Clustering: local search algorithms can’t explore the space, and hill-climbing algorithms get stuck in local optima

Freezing: backtracking algorithms take exponential time, repeatedly setting frozen variables the wrong way

But Gaussian elimination is a global change of variables, letting us turn a hard-looking problem into an easy one

If SAT has a similar kind of rearrangement — something totally different from backtracking or local search — then P=NP

Proving that it doesn’t is hard!
The Physicists’ Algorithm

A “message-passing” algorithm:
The Physicists’ Algorithm

A “message-passing” algorithm:

“I can’t give you what you want”
The Physicists’ Algorithm

A “message-passing” algorithm:

“Y ou’re the only one who can satisfy me”

“I can’t give you what you want”

“You’re the only one who can satisfy me”
Why Does It Work?

Random formulas are locally treelike.
Assume the neighbors are independent:

Proving that this works took a long time...
Building bridges between disciplines

A rich collaboration is growing up between physicists and computer scientists

Techniques from physics inspire mathematical conjectures and proof techniques, leading to new computer science

Algorithms inspired by physics can solve large real-world problems, such as analyzing the structure of social networks

Computer scientists view physical systems, e.g. quantum, in terms of their computational power, leading to new physics