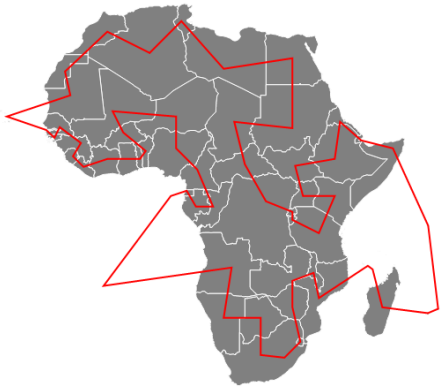
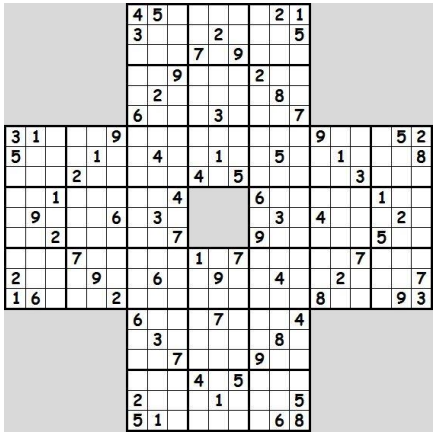
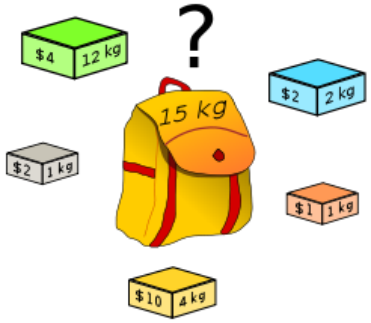




Phase transitions in random satisfiability problems

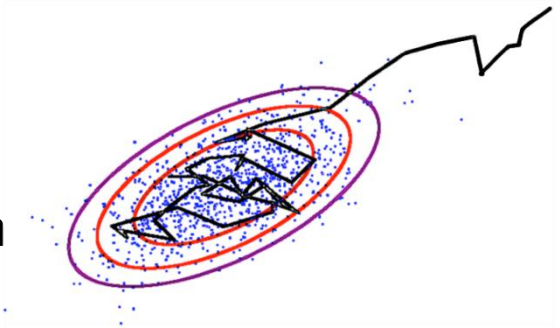
Eli Chertkov

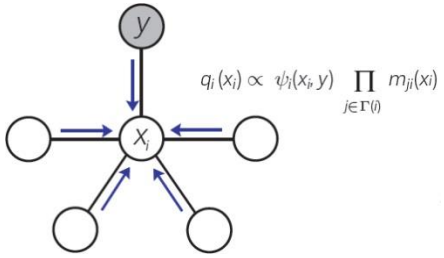
Algorithms Interest Group 9/5/2017

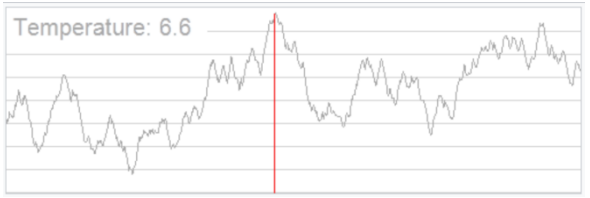


Statistical physics and computation

- Markov chain Monte Carlo (1953)
  - Method to integrate high-dimensional probability distributions.
  - Originally developed to study deviations from the ideal gas law.
- Belief propagation (1982)
  - Method to perform statistical inference on a network. Used often with error-correcting codes.
  - Based on methods from spin glass theory.
- Simulated Annealing (1983)
  - General purpose optimization algorithm.
  - Based on thermal annealing of materials.

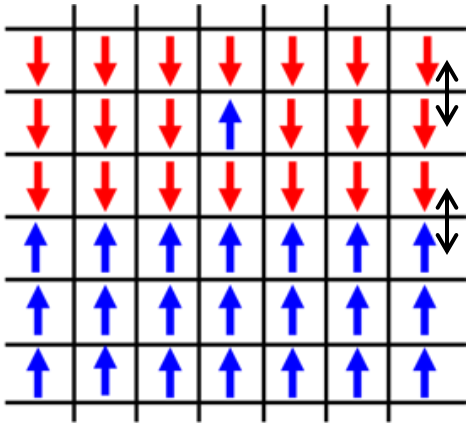






Phase transitions

Example: Ising model $S_i = \pm 1$



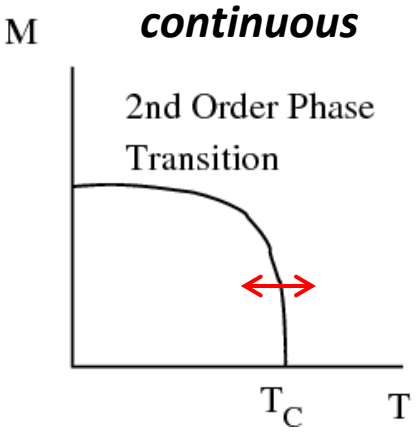
$$H = -J \sum_{\langle ij \rangle} S_i S_j$$

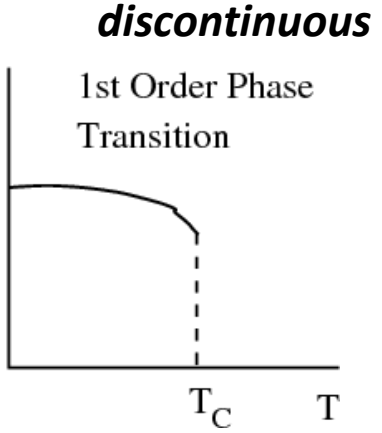
Order parameter

$$M = \sum_i \langle S_i \rangle / N$$

Scaling for continuous PT

$$\xi \sim |t|^{-\nu}$$





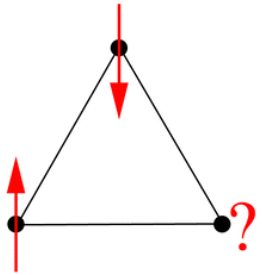
$\xi =$ correlation length

$t = (T - T_C)/T_C =$ reduced temperature

Spin glasses

Magnetic disorder induces frustration

Geometrical frustration



Spin glass, quenched disorder

$$H = -\sum_{\langle ij \rangle} J_{ij} S_i S_j - \sum_i H_i S_i$$

Interaction strengths drawn from random distributions.

There are a variety of analytic techniques that have been developed to study the thermodynamic properties of simple models of spin glasses.

Complexity in computer science

Computer scientists often classify algorithms based on their worst-case performance:

Quicksort	*O(n log(n))*
Selection sort	*O(n^2)*

They categorize algorithms based on their worst case behavior into classes, such as NP-complete:

Boolean satisfiability (SAT)

Travelling salesman problem

Hamiltonian path problem

…

which, intuitively, are problems that are easy to check, but hard to solve.

Practical instances of NP-complete problems can actually be *easy* to solve.

We can use some phase transition analysis to see this.

Boolean Satisfiability (*k*-SAT)

Determine whether a boolean formula made of binary variables can be evaluated to TRUE, or "satisfied."

k-SAT is an NP-complete decision problem that is used as a test bed for the development of heuristic algorithms.

Example *k*-SAT problem instance:

$$(x_1 \vee x_2 \vee \neg x_4) \wedge (x_1 \vee \neg x_3 \vee x_4) \wedge (x_2 \vee x_3 \vee \neg x_4) \wedge (x_1 \vee \neg x_2 \vee \neg x_3) \wedge (x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_3 \vee \neg x_4)$$

N=4 boolean variables
M=6 clauses
k=3 size clauses
Conjunctive normal form (CNF)

Question: Can we pick boolean variables so that this evaluates to TRUE?

Answer: Yes! The logical assignment $(x_1, x_2, x_3, x_4) = (1, 1, 1, 0)$ *evaluates the expression to 1.*

We call this *k*-SAT instance ***satisfiable***.

The relation of random *k*-SAT to spin glasses

The *k*-SAT problem instances can be generated randomly, with the variables participating in clauses chosen at random from all *N* boolean variables.

For each problem instance, the boolean variables can be mapped to spin variables

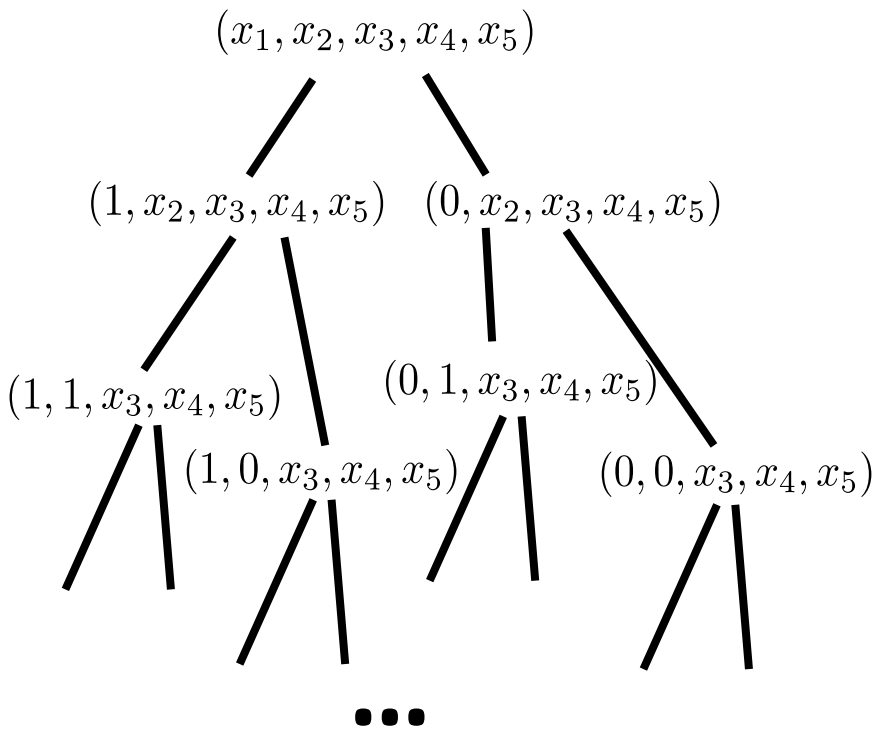
$$S_i = 2(x_i - 1/2)$$

Then, we can define a Hamiltonian for the *k*-SAT problem that looks exactly like a spin glass Hamiltonian, with the randomly generated clauses acting as the random interaction strengths.

Finding whether or not the problem is satisfiable is the same as finding whether the ground state energy is positive.

Solving a SAT problem

Naïve binary search



$(1, 0, 0, 1, 0)$ evaluates to TRUE.

Example k-SAT problem instance with k=3,N=5, M=5:

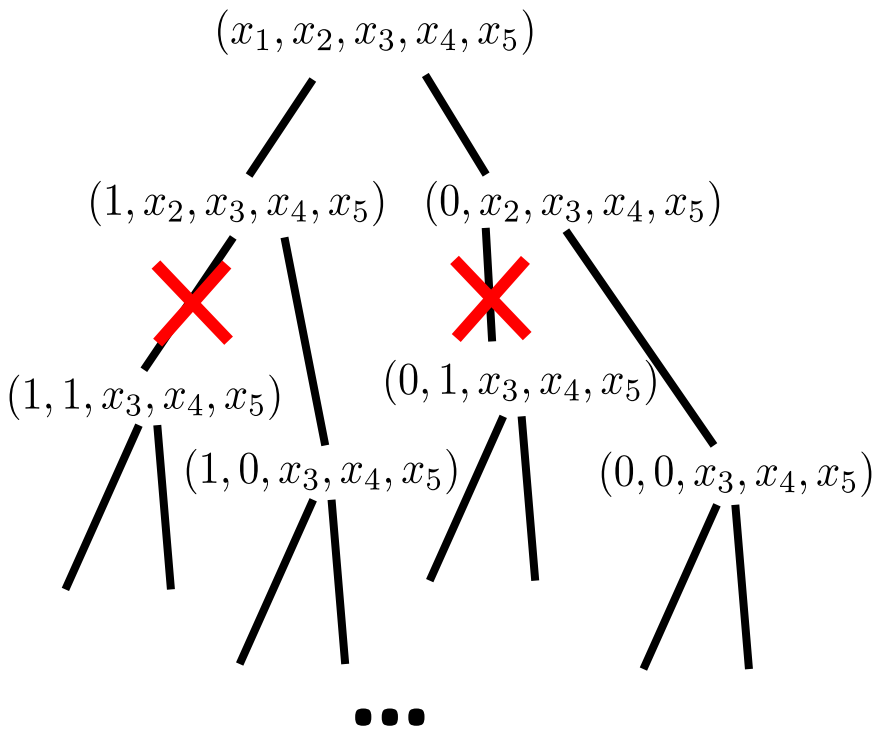
$$(x_1 \vee \neg x_2 \vee x_3) \wedge (\neg x_2 \vee x_3 \vee \neg x_4) \wedge (\neg x_1 \vee \neg x_3 \vee x_4) \wedge (x_1 \vee x_4 \vee x_5) \wedge (\neg x_1 \vee x_3 \vee \neg x_5)$$

Note that x_2 always shows up with the same "sign" in clauses.

x_2 is called a *pure literal.*

Davis-Putnam-Logemann-Loveland (DPLL) algorithm

DPLL



$$(x_1 \vee \neg x_2 \vee x_3) \wedge (\neg x_2 \vee x_3 \vee \neg x_4) \wedge (\neg x_1 \vee \neg x_3 \vee x_4) \wedge (x_1 \vee x_4 \vee x_5) \wedge (\neg x_1 \vee x_3 \vee \neg x_5)$$

Pure literal evaluation (PLE)

Find pure literals in the unassigned clauses and set them to 1 or 0 as needed.

Unit-propagation (UP)

If there is only one unassigned variable left in a clause, set it to 1 or 0 as needed.

Davis-Putnam-Logemann-Loveland (DPLL) algorithm

DPLL

1 $(x_1, x_2, x_3, x_4, x_5)$

↓ **PLE**

2 $(x_1, 0, x_3, x_4, x_5)$

1

$$(x_1 \vee \neg x_2 \vee x_3) \wedge (\neg x_2 \vee x_3 \vee \neg x_4) \wedge (\neg x_1 \vee \neg x_3 \vee x_4) \wedge (x_1 \vee x_4 \vee x_5) \wedge (\neg x_1 \vee x_3 \vee \neg x_5)$$

→

$$(x_1 \vee 1 \vee x_3) \wedge (1 \vee x_3 \vee \neg x_4) \wedge (\neg x_1 \vee \neg x_3 \vee x_4) \wedge (x_1 \vee x_4 \vee x_5) \wedge (\neg x_1 \vee x_3 \vee \neg x_5)$$

2 →

$$(\neg x_1 \vee \neg x_3 \vee x_4) \wedge (x_1 \vee x_4 \vee x_5) \wedge (\neg x_1 \vee x_3 \vee \neg x_5)$$

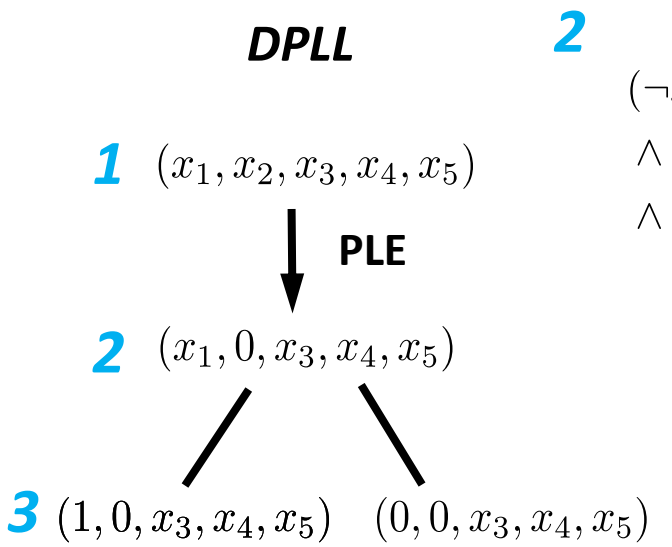
Pure literal evaluation (PLE)

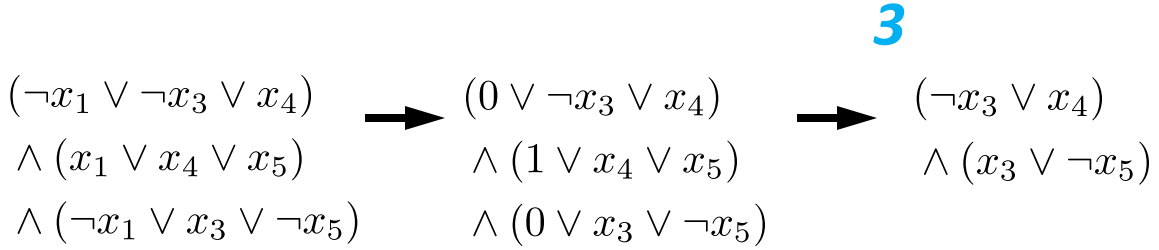
Find pure literals in the unassigned clauses and set them to 1 or 0 as needed.

Unit-propagation (UP)

If there is only one unassigned variable left in a clause, set it to 1 or 0 as needed.

Davis-Putnam-Logemann-Loveland (DPLL) algorithm





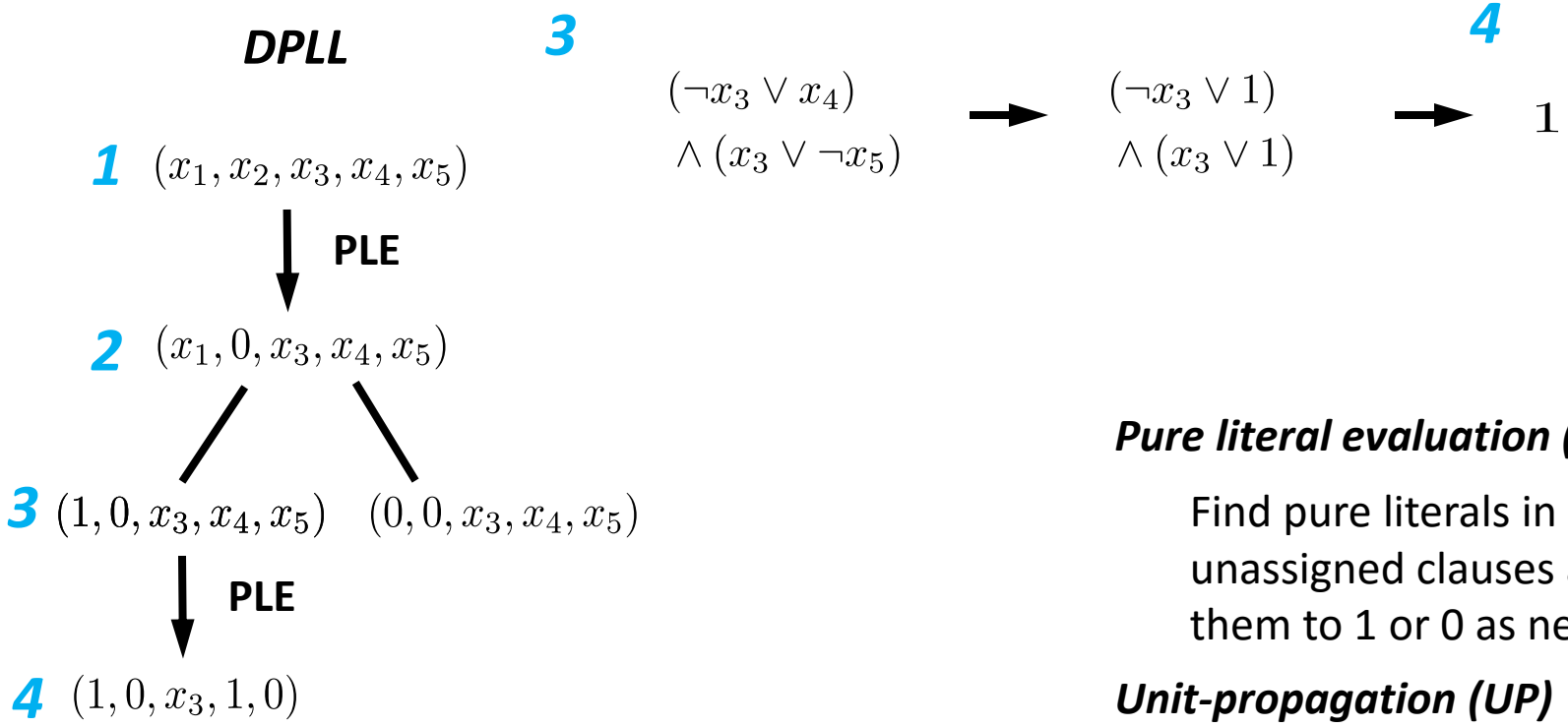
Pure literal evaluation (PLE)

Find pure literals in the unassigned clauses and set them to 1 or 0 as needed.

Unit-propagation (UP)

If there is only one unassigned variable left in a clause, set it to 1 or 0 as needed.

Davis-Putnam-Logemann-Loveland (DPLL) algorithm



We found two examples of vectors of binary variables that satisfy the k-SAT problem instance.

Pure literal evaluation (PLE)

Find pure literals in the unassigned clauses and set them to 1 or 0 as needed.

Unit-propagation (UP)

If there is only one unassigned variable left in a clause, set it to 1 or 0 as needed.

Analyzing random *k*-SAT problem instances

In the following analysis, I generated 10,000 random problem instances of *k*-SAT. The main property I looked at was

$$P_{SAT} = \text{fraction of problem instances that are satisfiable}$$

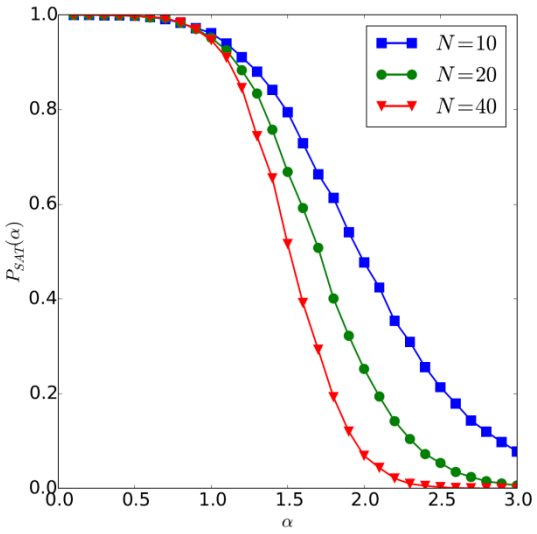
This is presented as a function of *clause density*

$$\alpha = M/N$$

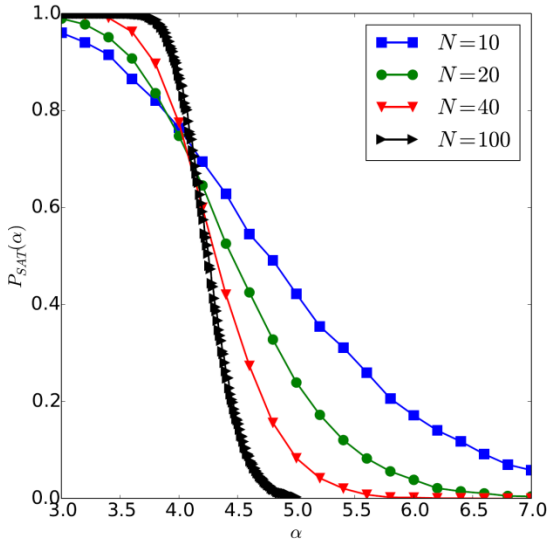
Satisfiability of random *k*-SAT problem instances

(Generated using Zchaff SAT solver)

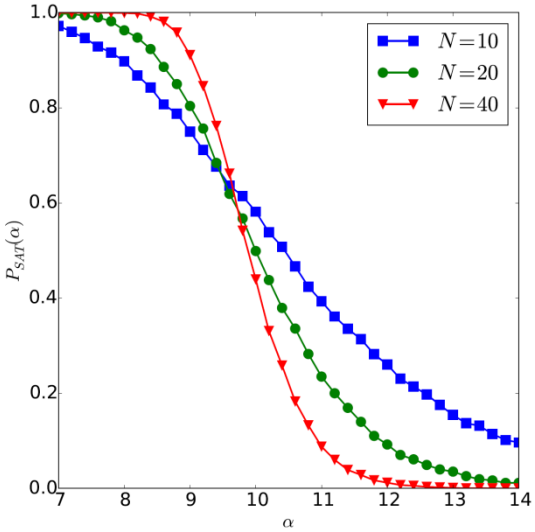
k=2



k=3



k=4



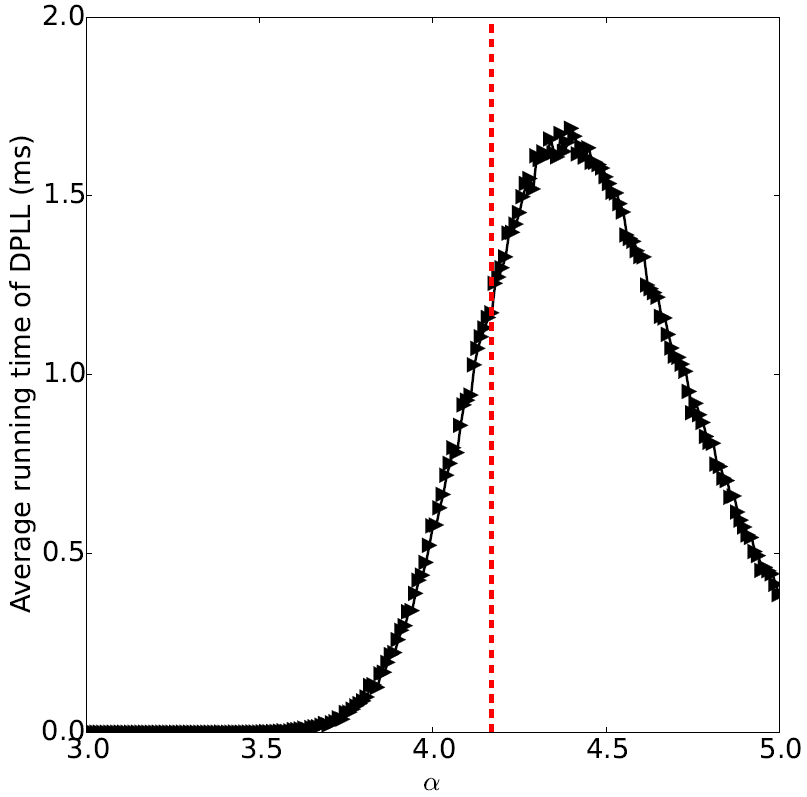
Continuous change in P_{SAT}

Problem instances are *P*

Discontinuous change in P_{SAT}

Problem instances are *NP-complete*

Running time of DPLL on random *k*-SAT



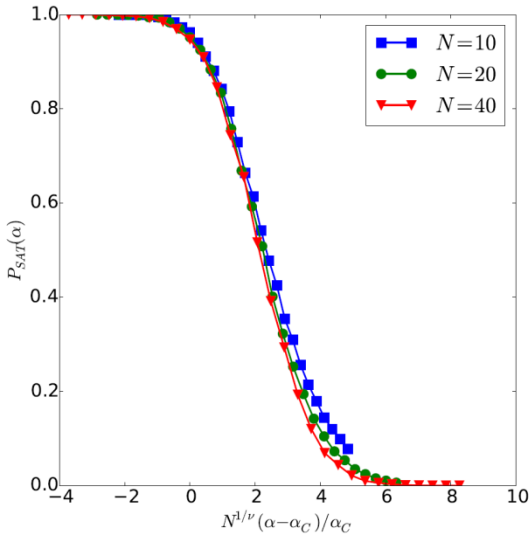
N=100, k=3

The hardest problem instances are the ones near the transition!

Finite size scaling

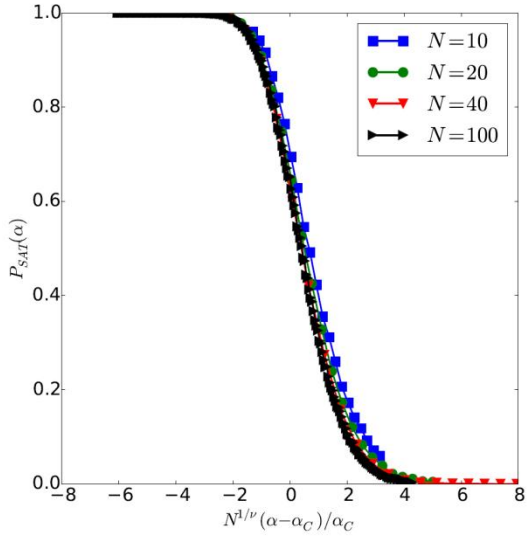
You can even perform “finite size scaling” to determine the critical exponents of the *k*-SAT transition.

k=2



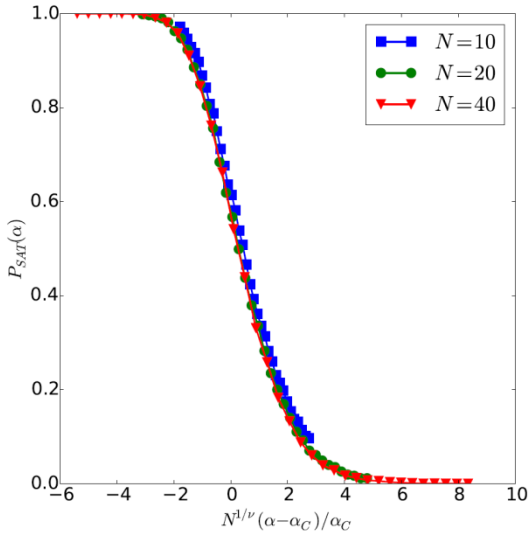
$$\nu = 2.6$$
$$\alpha_C = 1$$

k=3



$$\nu = 1.5$$
$$\alpha_C = 4.17$$

k=4



$$\nu = 1.25$$
$$\alpha_C = 9.75$$

References

My term paper for Prof. Nigel Goldenfeld's *Phase transitions and the Renormalization Group* class has all of the material from this presentation, as well as references:

http://guava.physics.uiuc.edu/~nigel/courses/563/Essays_2017/PDF/chertkov.pdf