Critical behaviour of combinatorial search algorithms, and the unitary-propagation universality class

Christophe Deroulers
LPTENS Paris

and

Rémi Monasson
LPTENS Paris

Universität zu Köln — May, 13th 2005

Outline

P & NP problems in computer science

Random $2 + p$-SAT

Combinatorial search algorithms

Analysis of the UC algorithm

The critical behaviour: first approach

The critical behaviour: computation

Conclusion

NP problems in computer science

P and NP problems

Let N be the size of a computer science problem. Roughly:

P = {“easy” problems } = { problems solvable in a time bounded by a Polynom in N }

NP = { “hard problems” } = { others, but where a solution can be tested in polynomial time }

P and NP problems

Let N be the size of a computer science problem. Roughly:

P = {“easy” problems } = { problems solvable in a time bounded by a Polynom in N }

NP = { “hard problems” } = { others, but where a solution can be tested in polynomial time }

Examples:

- Sorting N objects is easy (can be done in time $\simeq N \ln N$)
- Inverting a matrix of size N is easy (Gauss algorithm)

P and NP problems– cont'd

Believed to he hard:

- Finding the exact ground-state of a 3-dimensional sample of an Ising spin-glass (given the energy of the nearest-neighbours interactions)
- Traveling Salesman Problem (TSP): is the shortest path that goes at least once through each of N given cities shorter than 1000km?
- The graph coloring problem (COL): given a geographic map of N countries, is it possible to color it with k colors so that no two countries that share a frontier have the same color?
- Finding a non-trivial factorization of an integer with N digits

The SAT problem

Given boolean variables a, b, c, ...

Values of $a, b, c, \ldots \in \{\mathrm{TRUE}, \mathrm{FALSE}\}$

The SAT problem

Given boolean variables a, b, c, ...

and a boolean formula, e.g. $(a \text{ AND } (\text{NOT} b)) \text{ OR } c = \text{TRUE}$,

The SAT problem

Given boolean variables a, b, c, ...

and a boolean formula, e.g. $(a \text{ AND } (\text{NOT} b)) \text{ OR } c = \text{TRUE}$,

problem: is these an assignment of the variables that SATisfies the formula, or not ?
Answer to problem is yes or no.

Here, answer to problem is yes: solutions to formula are $\{a = \text{TRUE}$ and $b = \text{FALSE}$, whatever $c\}$ and $\{c = \text{TRUE}$, whatever a and $b\}$.

Size of problem N = size of the formula.

The SAT problem

Given boolean variables a, b, c, ...

and a boolean formula, e.g. $(a \text{ AND } (\text{NOT} b)) \text{ OR } c = \text{TRUE}$,

problem: is these an assignment of the variables that SATisfies the formula, or not ?
Answer to problem is yes or no.

Here, answer to problem is yes: solutions to formula are $\{a = \text{TRUE}$ and $b = \text{FALSE}$, whatever $c\}$ and $\{c = \text{TRUE}$, whatever a and $b\}$.

Size of problem N = size of the formula.

Historical motivation: automatic proof of theorems

Industrial interest: a test step during the conception of computer chips

Reduction

Reduction of problems in Polynomial time:

e.g. writing a SAT formula to decide if a graph can be colored with k colors;

drawing a graph, a k-coloration of which would solve a SAT formula

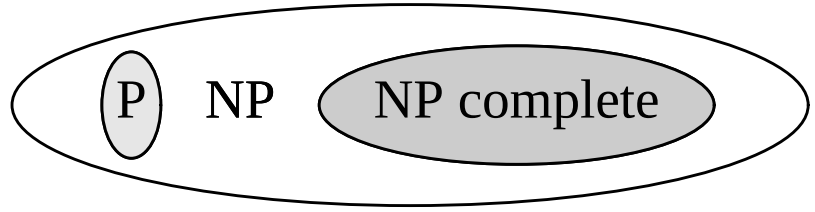
Reduction

Reduction of problems in Polynomial time:
e.g. writing a SAT formula to decide if a graph can be colored with k colors; drawing a graph, a k-coloration of which would solve a SAT formula

$\rightarrow$ {NP-complete problems} = { NP problems to which any NP problem can be reduced } (= hardest NP problems)

$\Rightarrow$ Solving one of them in polynomial time would solve all others.

Examples: SAT, COL, TSP.
Not integer factorization.



NP problems: comments

Comments:

- A non-polynomial time algorithm (method) to solve NP problems is easy: enumeration (needs polyn. time for each solution $\times$ exp. number of solutions).
- This is a worst-case analysis: for some spin-glass samples, some dispositions of the cities (TSP) or some geographic maps (COL), the answer is easy to find. But for some others, it is difficult. Tells nothing about *typical* case.
- This classification dates from the 1970s. No proof that P $\neq$ NP since then.

Random $2 + p$-SAT

Random K-SAT

Observation: in practice, many boolean formulae are easy to solve.

We want:

- a systematic way to generate difficult formulae
- study the average or typical case

Random K-SAT

Observation: in practice, many boolean formulae are easy to solve.

We want:

- a systematic way to generate difficult formulae
- study the average or typical case

$\rightarrow$ use a *random* distribution of formulae, with some parameter — the random K-SAT problem

Random K-SAT

Let a, b, c,... be N boolean variables and K a fixed integer.

Draw uniformly at random M K-uples of variables:

a OR b OR c	a OR e OR i	f OR g OR h	
d OR g OR j	b OR c OR i	e OR i OR j	$\updownarrow M$ clauses
$\underbrace{a \text{ OR } f \text{ OR } h}_{K \text{ variables}}$	a OR g OR j	a OR e OR f	

Random K-SAT

Let a, b, c,... be N boolean variables and K a fixed integer.

Draw uniformly at random M K-uples of variables:

a OR $\overline{b}$ OR $\overline{c}$	a OR $\overline{e}$ OR i	$\overline{f}$ OR g OR $\overline{h}$
d OR g OR $\overline{j}$	b OR c OR $\overline{i}$	e OR i OR j
a OR $\overline{f}$ OR $\overline{h}$	$\overline{a}$ OR $\overline{g}$ OR $\overline{j}$	a OR $\overline{e}$ OR f

negate each variable with uniform probability $\frac{1}{2}$
($\overline{b}$ means "NOT b")

Random K-SAT

Let a, b, c,... be N boolean variables and K a fixed integer.

Draw uniformly at random M K-uples of variables:

(a OR $\overline{b}$ OR $\overline{c}$) AND (a OR $\overline{e}$ OR i) AND ($\overline{f}$ OR g OR $\overline{h}$)

(d OR g OR $\overline{j}$) AND (b OR c OR $\overline{i}$) AND (e OR i OR j)

(a OR $\overline{f}$ OR $\overline{h}$) AND ($\overline{a}$ OR $\overline{g}$ OR $\overline{j}$) AND (a OR $\overline{e}$ OR f)

negate each variable with uniform probability $\frac{1}{2}$

and add ANDs between the K-clauses (K-uples) to build up a formula.

Static phase transition

Fix α and let $M, N \to +\infty$ with $M = \alpha N$ (thermodynamic limit with fixed clauses-per-variable ratio).

Static phase transition

Fix α and let $M, N \to +\infty$ with $M = \alpha N$ (thermodynamic limit with fixed clauses-per-variable ratio).

For 2-SAT, rigorous results: If $\alpha = \frac{M}{N} < 1$, almost surely the formula is satisfiable.
If $\alpha = \frac{M}{N} > 1$, almost surely the formula is not satisfiable.
Width of the transition region $\simeq N^{-1/3}$ when $N \to +\infty$.

Borgs, Chayes et al. 2001

Static phase transition

Fix α and let $M, N \to +\infty$ with $M = \alpha N$ (thermodynamic limit with fixed clauses-per-variable ratio).

For 2-SAT, rigorous results: If $\alpha = \frac{M}{N} < 1$, almost surely the formula is satisfiable.
If $\alpha = \frac{M}{N} > 1$, almost surely the formula is not satisfiable.
Width of the transition region $\simeq N^{-1/3}$ when $N \to +\infty$.

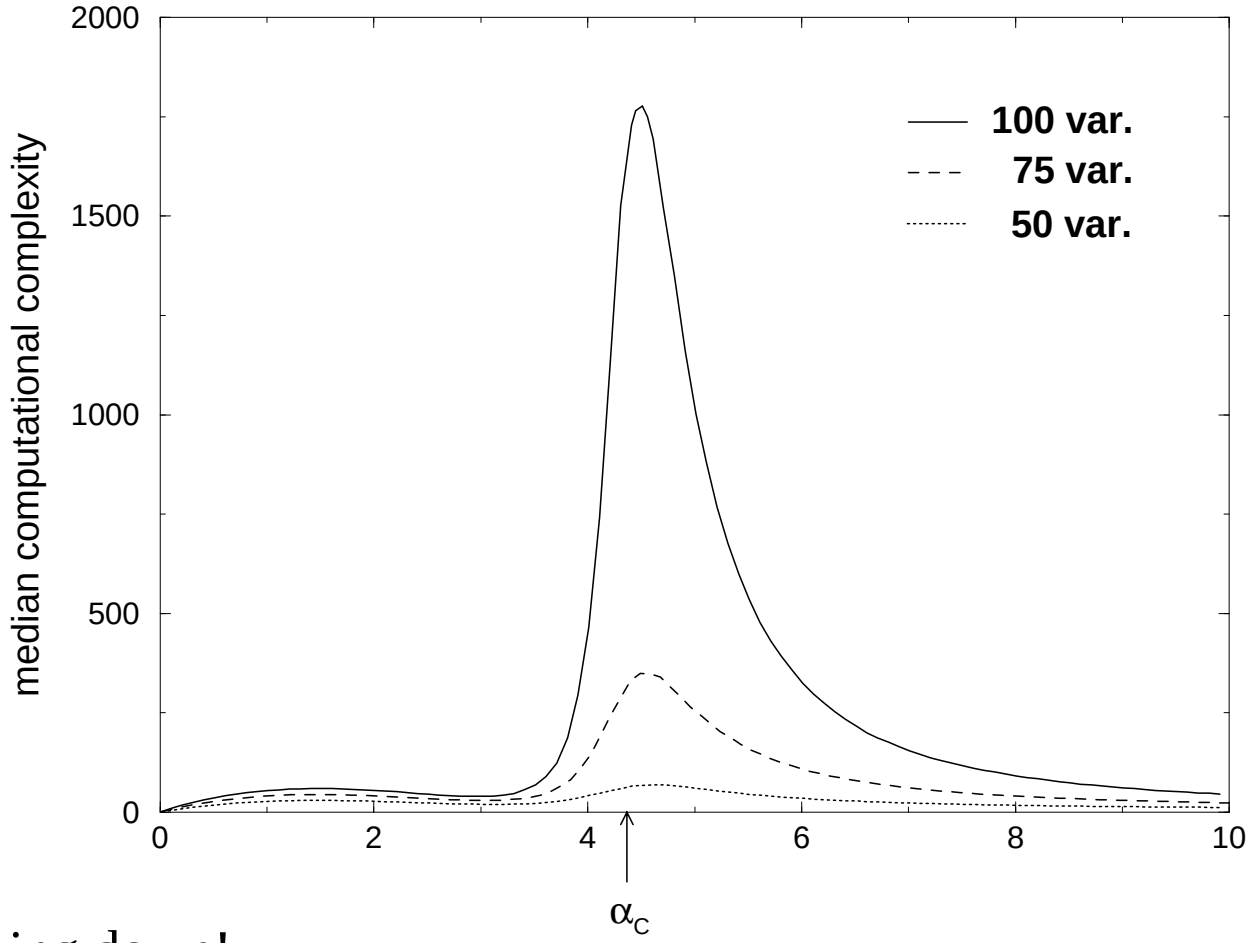
Borgs, Chayes et al. 2001

For 3-SAT, numerical results:
If $\alpha = \frac{M}{N} < 4.3$, almost surely the formula is satisfiable.
If $\alpha = \frac{M}{N} > 4.3$, almost surely the formula is not satisfiable.

late 1980s / early 1990s, Selman

Static phase transition

3-SAT, average computation time:



Critical slowing down!

K-SAT = spin glass (∞-dimension)

K-SAT = spin glass (∞-dimension)

Boolean variables $x_1, x_2, \ldots$	$\leftrightarrow$	Ising spins $S_1, S_2, \ldots$
TRUE, FALSE	$\leftrightarrow$	1, -1
Formula	$\leftrightarrow$	Hamiltonian:
(x_1 OR $\overline{x_2}$) AND x_3	$\leftrightarrow$	$H = (1 - S_1)(1 + S_2) + (1 - S_3)$
Solution	$\leftrightarrow$	Ground state (energy 0)

K-SAT = spin glass (∞-dimension)

Boolean variables $x_1, x_2, \ldots$	$\leftrightarrow$	Ising spins $S_1, S_2, \ldots$
TRUE, FALSE	$\leftrightarrow$	1, -1
Formula	$\leftrightarrow$	Hamiltonian:
$(x_1$ OR $\overline{x_2})$ AND x_3	$\leftrightarrow$	$H = (1 - S_1)(1 + S_2) + (1 - S_3)$
Solution	$\leftrightarrow$	Ground state (energy 0)

Replica computations give (non-rigorously) location of the phase transition

Monasson, Zecchina, Birolo, Weigt, Mezard, Parisi...

K-SAT = spin glass (∞-dimension)

Boolean variables $x_1, x_2, \ldots$	$\leftrightarrow$	Ising spins $S_1, S_2, \ldots$
TRUE, FALSE	$\leftrightarrow$	1, -1
Formula	$\leftrightarrow$	Hamiltonian:
$(x_1$ OR $\overline{x_2})$ AND x_3	$\leftrightarrow$	$H = (1 - S_1)(1 + S_2) + (1 - S_3)$
Solution	$\leftrightarrow$	Ground state (energy 0)

Replica computations give (non-rigorously) location of the phase transition

Monasson, Zecchina, Birolo, Weigt, Mezard, Parisi...

Rigorous results from the mathematics and computer science communities for 3-SAT:

- Width of the transition region $\to 0$ when $N \to +\infty$
- Threshold $\alpha_{\text{static}} < 4.51$ Dubois et al.
- Threshold $\alpha_{\text{static}} > 3.52$ Kirousis et al.

random $2 + p$-SAT

Let $0 \leq p \leq 1$. With N variables, draw uniformly at random M clauses that are

- of length 3 with probability p
- of length 2 with probability 1-p

$p = 0 \leftrightarrow$ 2-SAT
$p = 1 \leftrightarrow$ 3-SAT

Complete algorithms for resolution of SAT

Noncomplete algorithms

Given a SAT formula: start from a random assignment of its variable, and update it by “spin flips” until a solution is found.

Noncomplete algorithms

Given a SAT formula: start from a random assignment of its variable, and update it by "spin flips" until a solution is found.

May be efficient; used in practice

Interesting out-of-equilibrium phenomena (but nonphysical dynamics)

Can't give a proof that no solution exists (answer to problem is yes, or run forever)

Won't be discussed here.

Complete algorithms

Algorithms that, for a formula:

- either give a solution (answer to problem is yes)
- or guarantee that there is no solution (answer to problem is no)

Principle: explore the tree of all possibilities.

Davis, Putnam et al. 1960s

Example

0. Let the formula be:

$$a \text{ OR } \overline{b} \text{ OR } \overline{c} \qquad \overline{a} \text{ OR } e \text{ OR } i \qquad \overline{a} \text{ OR } g \text{ OR } \overline{h}$$

$$d \text{ OR } \overline{g} \text{ OR } j \qquad g \text{ OR } h \text{ OR } \overline{i} \qquad \overline{e} \text{ OR } i \text{ OR } j$$

$$a \text{ OR } \overline{f} \text{ OR } \overline{h} \qquad a \text{ OR } \overline{b} \text{ OR } \overline{j} \qquad \overline{a} \text{ OR } \overline{e} \text{ OR } f$$

Example

1. Choose a variable:

a	OR	$\overline{b}$	OR	$\overline{c}$	$\overline{a}$	OR	e	OR	i	$\overline{a}$	OR	g	OR	$\overline{h}$
d	OR	$\overline{g}$	OR	j	g	OR	h	OR	$\overline{i}$	$\overline{e}$	OR	i	OR	j
a	OR	$\overline{f}$	OR	$\overline{h}$	a	OR	$\overline{b}$	OR	$\overline{j}$	$\overline{a}$	OR	$\overline{e}$	OR	f

Example

2. Assign it:

$$
\begin{array}{llllllllllll}
a & \text{OR} & \overline{b} & \text{OR} & \overline{c} & \quad \overline{a} & \text{OR} & e & \text{OR} & i & \quad \overline{a} & \text{OR} & g & \text{OR} & \overline{h} \\
d & \text{OR} & \overline{g} & \text{OR} & j & \quad g & \text{OR} & h & \text{OR} & \overline{i} & \quad \overline{e} & \text{OR} & i & \text{OR} & j \\
a & \text{OR} & \overline{f} & \text{OR} & \overline{h} & \quad a & \text{OR} & \overline{b} & \text{OR} & \overline{j} & \quad \overline{a} & \text{OR} & \overline{e} & \text{OR} & f
\end{array}
$$

$$a := \text{true}$$

Example

3. Reduce the clauses where it appears:

			TRUE					e	OR	i				g	OR	$\overline{h}$
d	OR	$\overline{g}$	OR	j		g	OR	h	OR	$\overline{i}$		$\overline{e}$	OR	i	OR	j
			TRUE					TRUE						$\overline{e}$	OR	f

$$a := \text{true}$$

Example

4. Choose another variable:

$$\begin{array}{ccc} & \text{TRUE} & \\ d \text{ OR } \overline{g} \text{ OR } j & & \end{array} \qquad \begin{array}{ccc} & e \text{ OR } i & \\ g \text{ OR } h \text{ OR } \overline{i} & & \\ & \text{TRUE} & \end{array} \qquad \begin{array}{ccc} & g \text{ OR } \overline{h} & \\ \overline{e} \text{ OR } i \text{ OR } j & & \\ & \overline{e} \text{ OR } f & \end{array}$$

$$a := \text{true}$$

Example

5. Assign it:

$$\begin{array}{ccccccccccccc} & & \text{TRUE} & & & & e & \text{OR} & i & & & g & \text{OR} & \overline{h} \\ d & \text{OR} & \overline{g} & \text{OR} & j & g & \text{OR} & h & \text{OR} & \overline{i} & \overline{e} & \text{OR} & i & \text{OR} & j \\ & & \text{TRUE} & & & & & \text{TRUE} & & & & & \overline{e} & \text{OR} & f \end{array}$$

$$a := \text{true}$$

$$e := \text{false}$$

Example

6. Reduce the clauses where it appears:

			TRUE							i		g	OR	$\overline{h}$
d	OR	$\overline{g}$	OR	j		g	OR	h	OR	$\overline{i}$		TRUE		
			TRUE					TRUE				TRUE		

$a := \text{true}$

$e := \text{false}$

Example

7. Natural choice: the unitary clause (1-clause).

			TRUE							i		g	OR	$\overline{h}$
d	OR	$\overline{g}$	OR	j		g	OR	h	OR	$\overline{i}$		TRUE		
			TRUE					TRUE				TRUE		

$$a := \text{true}$$

$$e := \text{false}$$

Example

8. Assign the variable...

			TRUE							i		g	OR	$\overline{h}$
d	OR	$\overline{g}$	OR	j		g	OR	h	OR	$\overline{i}$		TRUE		
			TRUE					TRUE				TRUE		

$a :=$ true

$e :=$ false

$i :=$ true

Example

9. Reduce...

		TRUE				TRUE		g	OR	$\overline{h}$
d	OR	$\overline{g}$	OR	j	g	OR	h	TRUE		
		TRUE				TRUE		TRUE		

$$a := \text{true}$$

$$e := \text{false}$$

$$i := \text{true}$$

Example

10. Choose a variable:

		TRUE				TRUE		g	OR	$\overline{h}$
d	OR	$\overline{g}$	OR	j	g	OR	h	TRUE		
		TRUE				TRUE		TRUE		

$$a := \text{true}$$

$$e := \text{false}$$

$$i := \text{true}$$

Example

11. Assign it:

		TRUE				TRUE		g	OR	$\overline{h}$
d	OR	$\overline{g}$	OR	j	g	OR	h	TRUE		
		TRUE				TRUE		TRUE		

$$a := \text{true}$$

$$e := \text{false}$$

$$i := \text{true}$$

$$g := \text{false}$$

Example

12. Reduce...

TRUE	TRUE	$\overline{h}$
TRUE	h	TRUE
TRUE	TRUE	TRUE

$a :=$ true

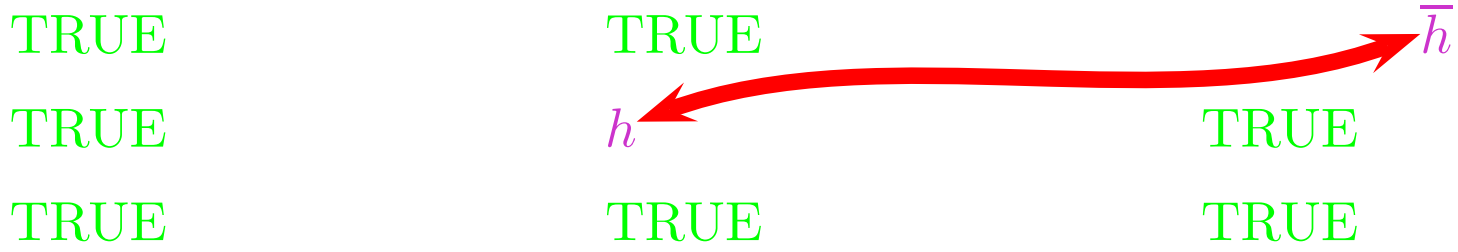
$e :=$ false

$i :=$ true

$g :=$ false

Example

13. Contradiction!



$a :=$ true

$e :=$ false

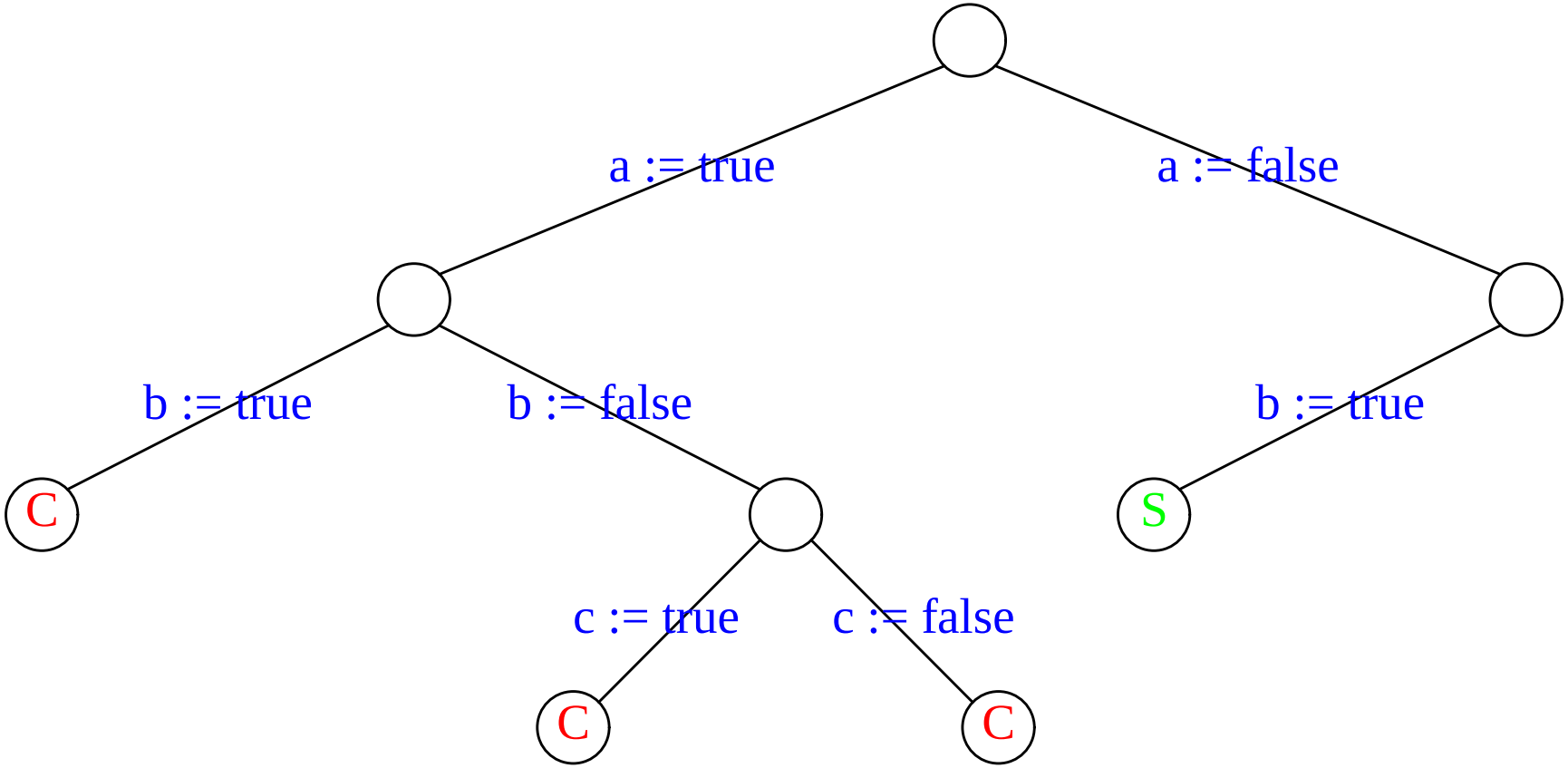
$i :=$ true

$g :=$ false

Complete algorithms

Principle: explore the tree of all possibilities, and backtrack when find a contradiction.

Davis, Putnam et al. 1960s



Heuristics “UC”

How to choose variables to assign?

Heuristics “Unit Clause”:

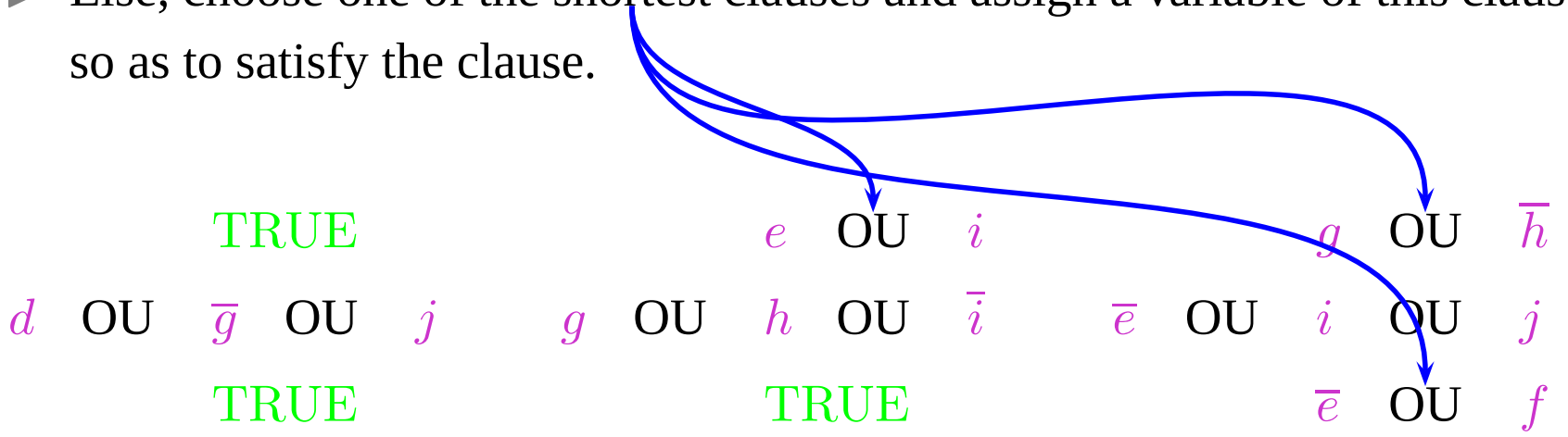
- If there are 1-clauses, take a variable there (you will have to satisfy 1-clauses now or later anyway) and satisfy its 1-clause. Unit Propagation
- Else, take a variable at random and assign it at random.

$$\begin{array}{cccccc} & & \text{TRUE} & & & \\ d & \text{OU} & \overline{g} & \text{OU} & j & \\ & & \text{TRUE} & & & \end{array} \qquad \begin{array}{cccccc} & & e & \text{OU} & i & \\ g & \text{OU} & h & \text{OU} & \overline{i} & \\ & & \text{TRUE} & & & \end{array} \qquad \begin{array}{cccccc} & & g & \text{OU} & \overline{h} & \\ \overline{e} & \text{OU} & i & \text{OU} & j & \\ & & \overline{e} & \text{OU} & f & \end{array}$$

Heuristics “GUC”

Heuristics “Generalized Unit Clause”:

- If there are 1-clauses, take a variable there (you will have to satisfy 1-clauses now or later anyway) and satisfy its 1-clause. Unit Propagation
- Else, choose one of the shortest clauses and assign a variable of this clause so as to satisfy the clause.



Success without backtracking

Central object of our computations: probability P that a solution is found *without* backtracking ("success probability")

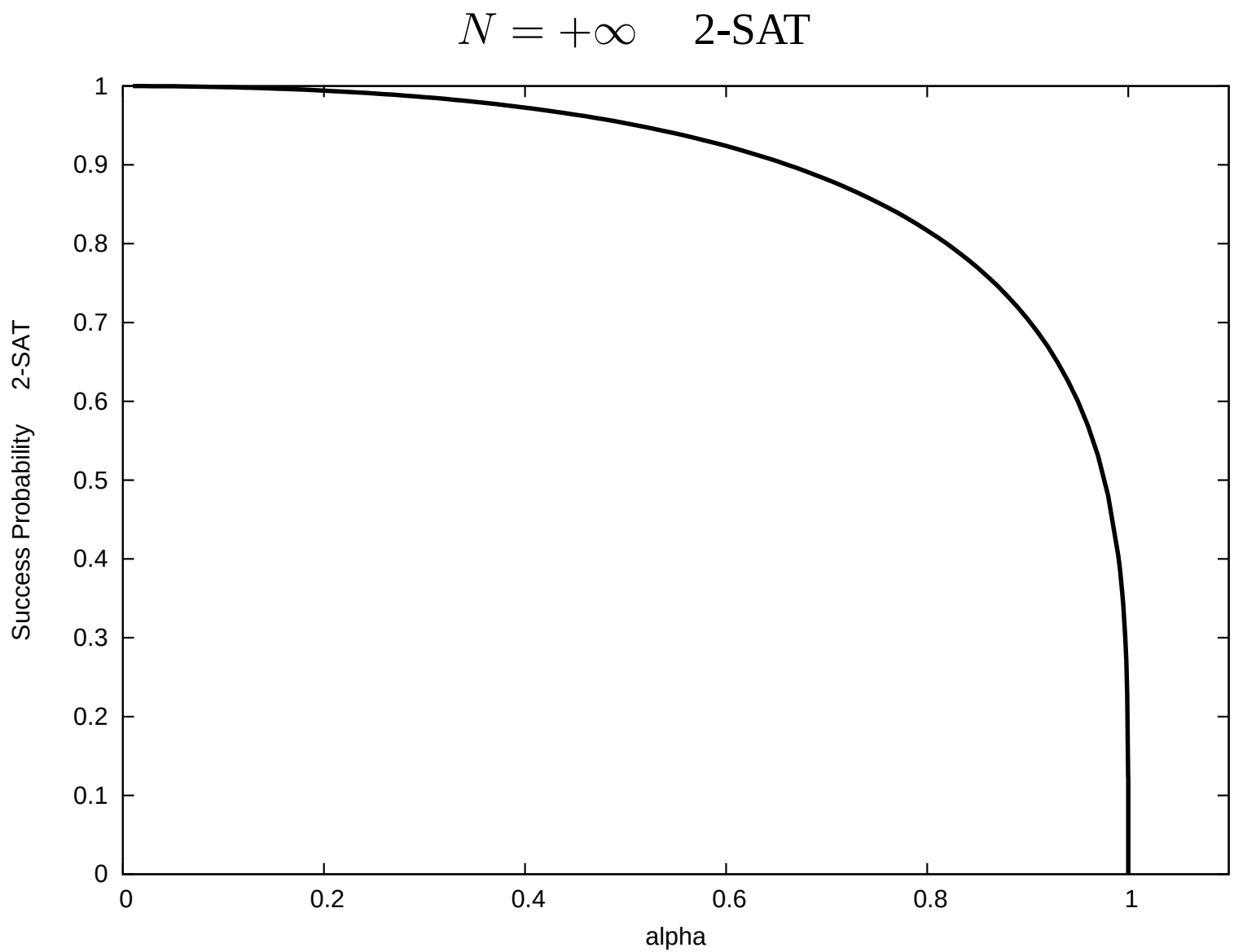
Success without backtracking

Central object of our computations: probability P that a solution is found *without* backtracking ("success probability")

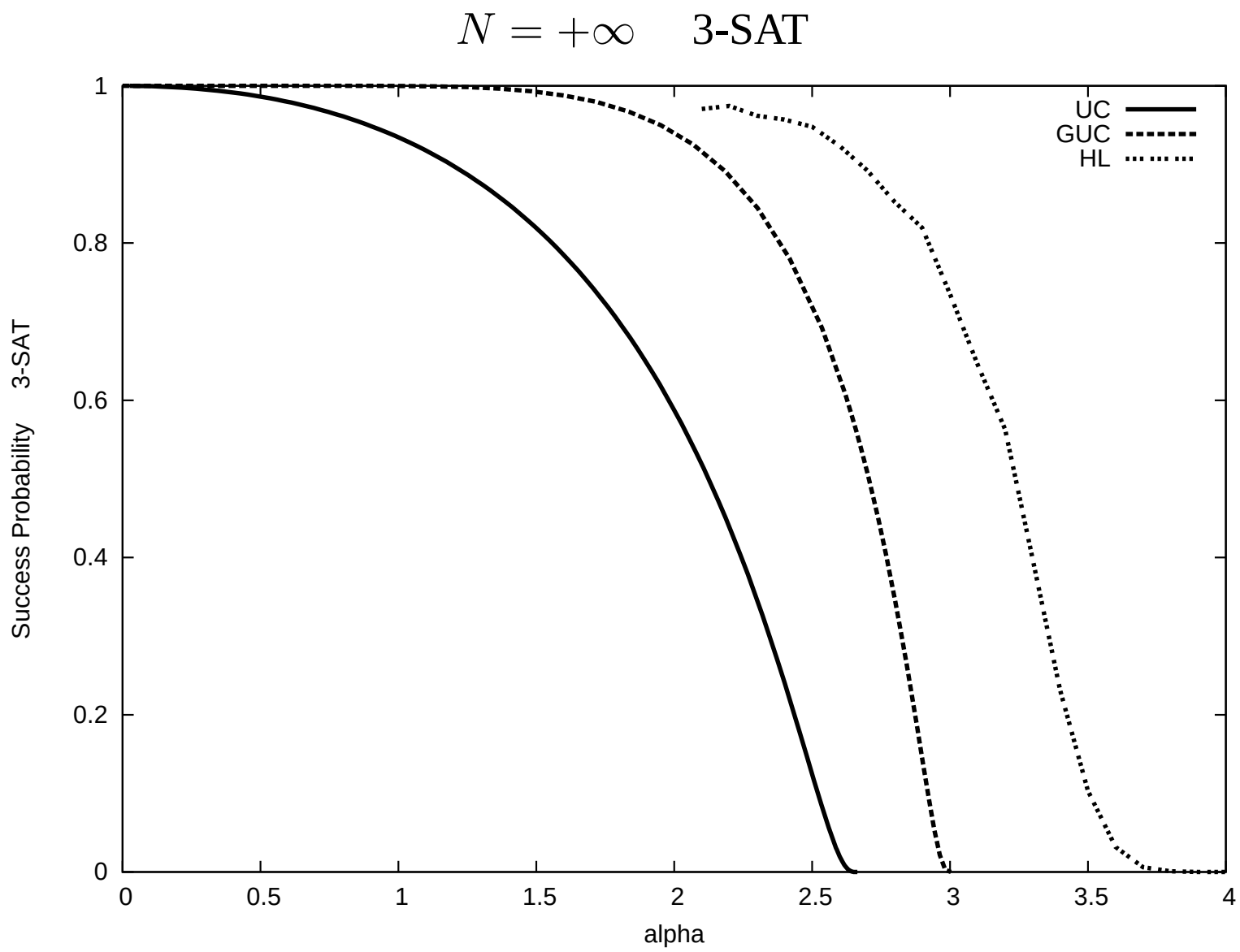
Intuitively, for N large:

No backtracking		Complete algorithm (with back.)
finite P	$\leftrightarrow$	polynomial time T
$P \approx e^{-N}$	$\leftrightarrow$	exponential time T

Success without backtracking - 2-SAT



Success without backtracking - 3-SAT



A dynamic phase transition

$\rightarrow$ 2 kinds of thresholds:

$\alpha_{\mathrm{static}}(p)$ depends only on the disorder distribution : are there solutions to formula?

$\alpha_{\mathrm{algo}}(p)$ depends also on the algorithm (=choice of dynamics): are we smart enough to quickly find solutions to formula?

A dynamic phase transition

$\longrightarrow$ 2 kinds of thresholds:

$\alpha_{\mathrm{static}}(p)$ depends only on the disorder distribution : are there solutions to formula?

$\alpha_{\mathrm{algo}}(p)$ depends also on the algorithm (=choice of dynamics): are we smart enough to quickly find solutions to formula?

Clearly, $\alpha_{\mathrm{algo}}(p) \leq \alpha_{\mathrm{static}}(p)$: if there is no solution, no success!

A dynamic phase transition

$\rightarrow$ 2 kinds of thresholds:

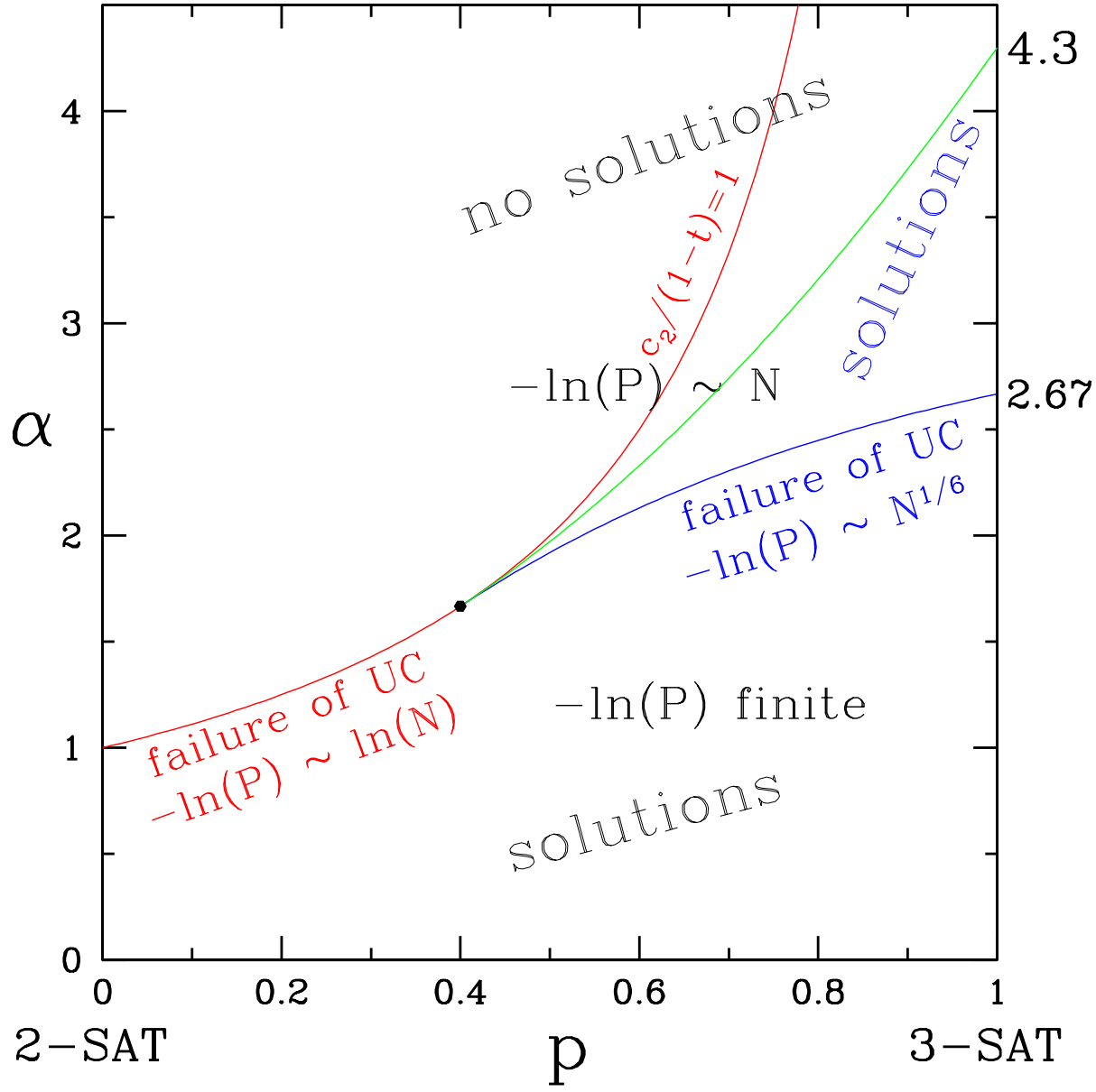
$\alpha_{\mathrm{static}}(p)$ depends only on the disorder distribution : are there solutions to formula?

$\alpha_{\mathrm{algo}}(p)$ depends also on the algorithm (=choice of dynamics): are we smart enough to quickly find solutions to formula?

Clearly, $\alpha_{\mathrm{algo}}(p) \leq \alpha_{\mathrm{static}}(p)$: if there is no solution, no success!

Open problem: find a smart algorithm with $\alpha_{\mathrm{algo}} = \alpha_{\mathrm{static}}$.

Phase space for algo. UC on $2 + p$-SAT



Analysis of algorithm UC

Analysis of UC on $2 + p$-SAT

Input: a random $2 + p$-SAT formula with N variables and αN clauses.

Analysis of UC on $2 + p$-SAT

Input: a random $2 + p$-SAT formula with N variables and αN clauses.

$\downarrow$

`Algorithm UC`

While there are clauses:

- If there are 1-clauses, take a variable there (you will have to satisfy 1-clauses now or later anyway) and satisfy its 1-clause. Unit Propagation
- Else, take a variable at random and assign it at random.

Don't backtrack, stop if contradiction found.

Analysis of UC on $2+p$-SAT

Input: a random $2+p$-SAT formula with N variables and αN clauses.

$\downarrow$

`Algorithm UC`

While there are clauses:

- If there are 1-clauses, take a variable there (you will have to satisfy 1-clauses now or later anyway) and satisfy its 1-clause. Unit Propagation
- Else, take a variable at random and assign it at random.

Don't backtrack, stop if contradiction found.

$\downarrow$

Output: yes (success) or don't know (failure)

Analysis of UC on $2+p$-SAT

Input: a random $2+p$-SAT formula with N variables and αN clauses.

$\downarrow$

`Algorithm UC`

While there are clauses:

- If there are 1-clauses, take a variable there (you will have to satisfy 1-clauses now or later anyway) and satisfy its 1-clause. Unit Propagation
- Else, take a variable at random and assign it at random.

Don't backtrack, stop if contradiction found.

$\downarrow$

Output: yes (success) or don't know (failure)

Probability of success at large N=?

Analysis of UC on $2+p$-SAT

Input: a random $2+p$-SAT formula with N variables and αN clauses.

$\downarrow$

`Algorithm UC`

While there are clauses:

- If there are 1-clauses, take a variable there (you will have to satisfy 1-clauses now or later anyway) and satisfy its 1-clause. Unit Propagation
- Else, take a variable at random and assign it at random.

Don't backtrack, stop if contradiction found.

$\downarrow$

Output: yes (success) or don't know (failure)

Probability of success at large N=?

Analysis of UC on $2 + p$-SAT

Time T := number of assigned variables.

$0 \leq T \leq N$.

Analysis of UC on $2 + p$-SAT

Time T := number of assigned variables.
$0 \leq T \leq N$.

2 important remarks:

After averaging over "disorder", i.e. the input data (the SAT formula), the distribution of partially reduced formulae at time T is completely fixed by the knowledge of the number of 1-, 2-, 3-clauses
$\vec{C}(T) := (C_3(T), C_2(T), C_1(T))$.

Franco

In other words: the annealed approximation is exact.

Analysis of UC on $2 + p$-SAT

Time T := number of assigned variables.
$0 \leq T \leq N$.

2 important remarks:

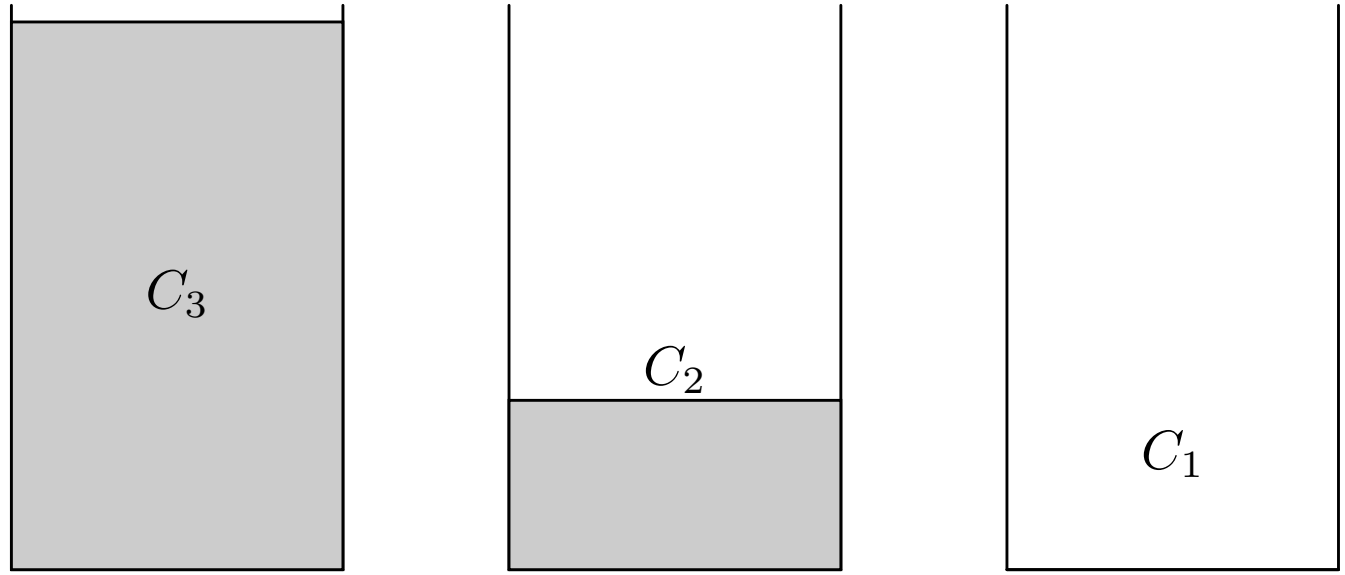
After averaging over "disorder", i.e. the input data (the SAT formula), the distribution of partially reduced formulae at time T is completely fixed by the knowledge of the number of 1-, 2-, 3-clauses
$\vec{C}(T) := (C_3(T), C_2(T), C_1(T))$.

In other words: the annealed approximation is exact.

C_3 and C_2 are self-averaging. $\rightsquigarrow$ We only need to compute $\langle C_2(T) \rangle$ and $\langle C_3(T) \rangle$.

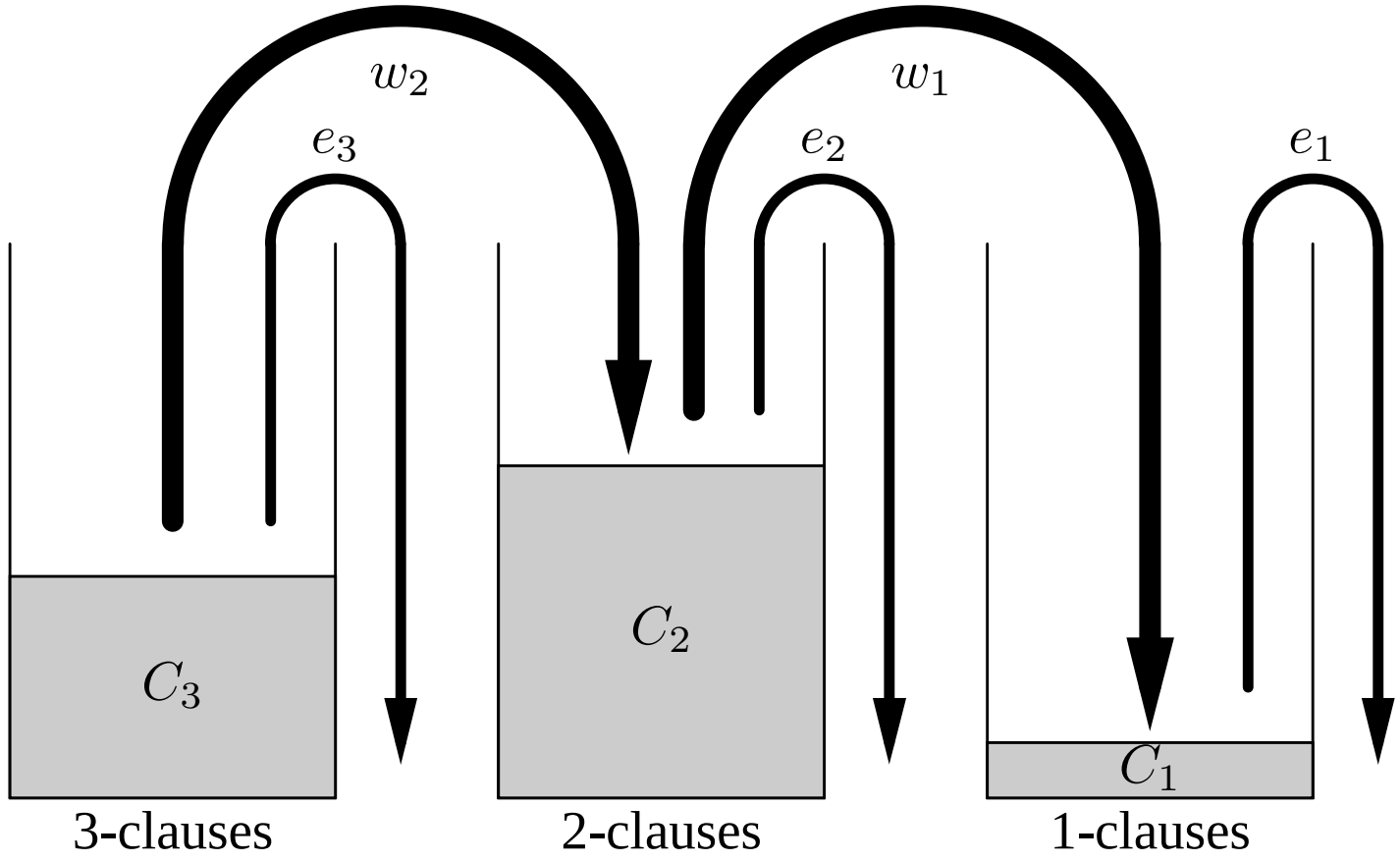
Analysis of UC on $2 + p$-SAT

Flow of clauses:



Analysis of UC on $2 + p$-SAT

Flow of clauses:



Analysis of UC: continuous time limit

For the averages:

Chao & Franco

$$\begin{aligned}
\langle C_3(T+1)\rangle - \langle C_3(T)\rangle &= -\frac{3\langle C_3\rangle}{N-T} \\
\langle C_2(T+1)\rangle - \langle C_2(T)\rangle &= -\frac{2\langle C_2\rangle}{N-T} + \frac{1}{2}\frac{3\langle C_3\rangle}{N-T} \\
\langle C_1(T+1)\rangle - \langle C_1(T)\rangle &= \frac{\langle C_2\rangle}{N-T} - \mathbb{E}\left[C_1 > 0\right]
\end{aligned}$$

Analysis of UC: continuous time limit

For the averages: Chao & Franco

$$\begin{aligned}
\langle C_3(T+1)\rangle - \langle C_3(T)\rangle &= -\frac{3\langle C_3\rangle}{N-T} \\
\langle C_2(T+1)\rangle - \langle C_2(T)\rangle &= -\frac{2\langle C_2\rangle}{N-T} + \frac{1}{2}\frac{3\langle C_3\rangle}{N-T} \\
\langle C_1(T+1)\rangle - \langle C_1(T)\rangle &= \frac{\langle C_2\rangle}{N-T} - \mathbb{E}\left[C_1 > 0\right]
\end{aligned}$$

$C_j(T)$ varies over times T of order 1
$\frac{C_j}{N} =: c_j(t := \frac{T}{N})$ varies over times T of order N

$\Rightarrow C_j(T) = N c_j(\frac{T}{N}) + o(N)$

Hence:

$$\begin{aligned}
\frac{dc_3}{dt} &= -\frac{3c_3}{1-t} \\
\frac{dc_2}{dt} &= -\frac{2c_2}{1-t} + \frac{1}{2}\frac{3c_3}{1-t} \\
\frac{dc_1}{dt} &= \frac{c_2}{1-t} - \rho_1
\end{aligned}$$

where $\rho_1(t) = \mathbb{E}\left[C_1(T) > 0\right]$ is the probability that there is at least one 1-clause at time t.

Hence:

$$\begin{aligned}
\frac{dc_3}{dt} &= -\frac{3c_3}{1-t} \\
\frac{dc_2}{dt} &= -\frac{2c_2}{1-t} + \frac{1}{2}\frac{3c_3}{1-t} \\
\frac{dc_1}{dt} &= \frac{c_2}{1-t} - \rho_1
\end{aligned}$$

where $\rho_1(t) = \mathbb{E}\left[C_1(T) > 0\right]$ is the probability that there is at least one 1-clause at time t.

For 3-SAT: $c_3(t) = \alpha(0)(1-t)^3$ and $c_2(t) = \frac{3}{2}\alpha(0)t(1-t)^2$.

Analysis of UC : study of C_1

Proba. that we notice no contradiction between T and $T+1$:

$$\left(1-\frac{1}{2(N-T)}\right)^{\max(C_1-1,0)}$$

hence proba. that we notice no contradiction from 0 to $T=Nt$ (success):

$$\exp\left(-\int_0^t dt\frac{\langle\max(C_1-1,0)\rangle}{2(1-t)}\right)$$

$\rightarrow$ we need information about C_1.

Transition matrix for C_1

$$H_N[C_1' \leftarrow C_1; T, C_2] = \sum_{s_2, r_2 = 0}^{C_2} \binom{C_2}{s_2, r_2} \left(\frac{1}{N-T}\right)^{s_2+r_2} \left(1 - \frac{2}{N-T}\right)^{C_2 - s_2 - r_2} \times$$

$$\Big[\delta_{C_1=0}\delta_{C_1'=r_2} + (1-\delta_{C_1=0})\times$$

$$\sum_{s_1=0}^{C_1-1} \binom{C_1-1}{s_1} \left(\frac{1}{2(N-T)}\right)^{s_1} \left(1 - \frac{1}{N-T}\right)^{C_1-1-s_1} \delta_{C_1'=C_1-1-s_1+r_2}\Big]$$

Transition matrix for C_1

$$H_N[C_1' \leftarrow C_1; T, C_2] = \sum_{s_2, r_2 = 0}^{C_2} \binom{C_2}{s_2, r_2} \left(\frac{1}{N-T}\right)^{s_2+r_2} \left(1 - \frac{2}{N-T}\right)^{C_2 - s_2 - r_2} \times$$

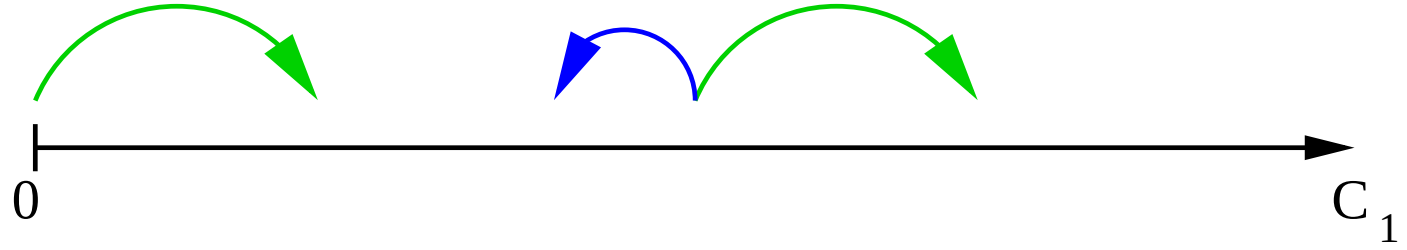
$$\Big[\delta_{C_1=0}\delta_{C_1'=r_2} + (1-\delta_{C_1=0})\times$$

$$\sum_{s_1=0}^{C_1-1} \binom{C_1-1}{s_1} \left(\frac{1}{2(N-T)}\right)^{s_1} \left(1 - \frac{1}{N-T}\right)^{C_1-1-s_1} \delta_{C_1'=C_1-1-s_1+r_2}\Big]$$

- H depends (explicitly) only on C_1, C_2 and T.
- Exact expression, contains the whole information but C_2.
- Correct only for this distribution of disorder (of formulae)
- but correct for all algorithms that use the Unit Propagation rule (variables in 1-clauses taken first).

Random walk with bias

$C_1 \geq 0$ has a biased random walk:

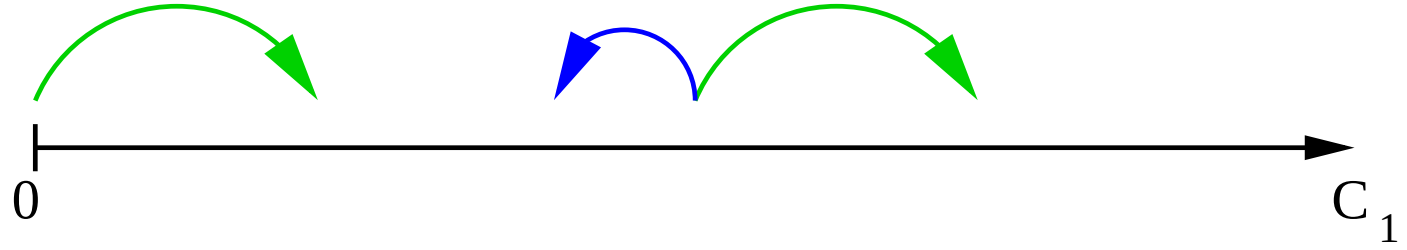


Step left -1 because of one 1-clause elimination

Step right $+d$ [d=binomial variable with proba. $1/(N-T)$ among $C_2(T)$] because of 1- and 2-clauses reduction

Random walk with bias

$C_1 \geq 0$ has a biased random walk:

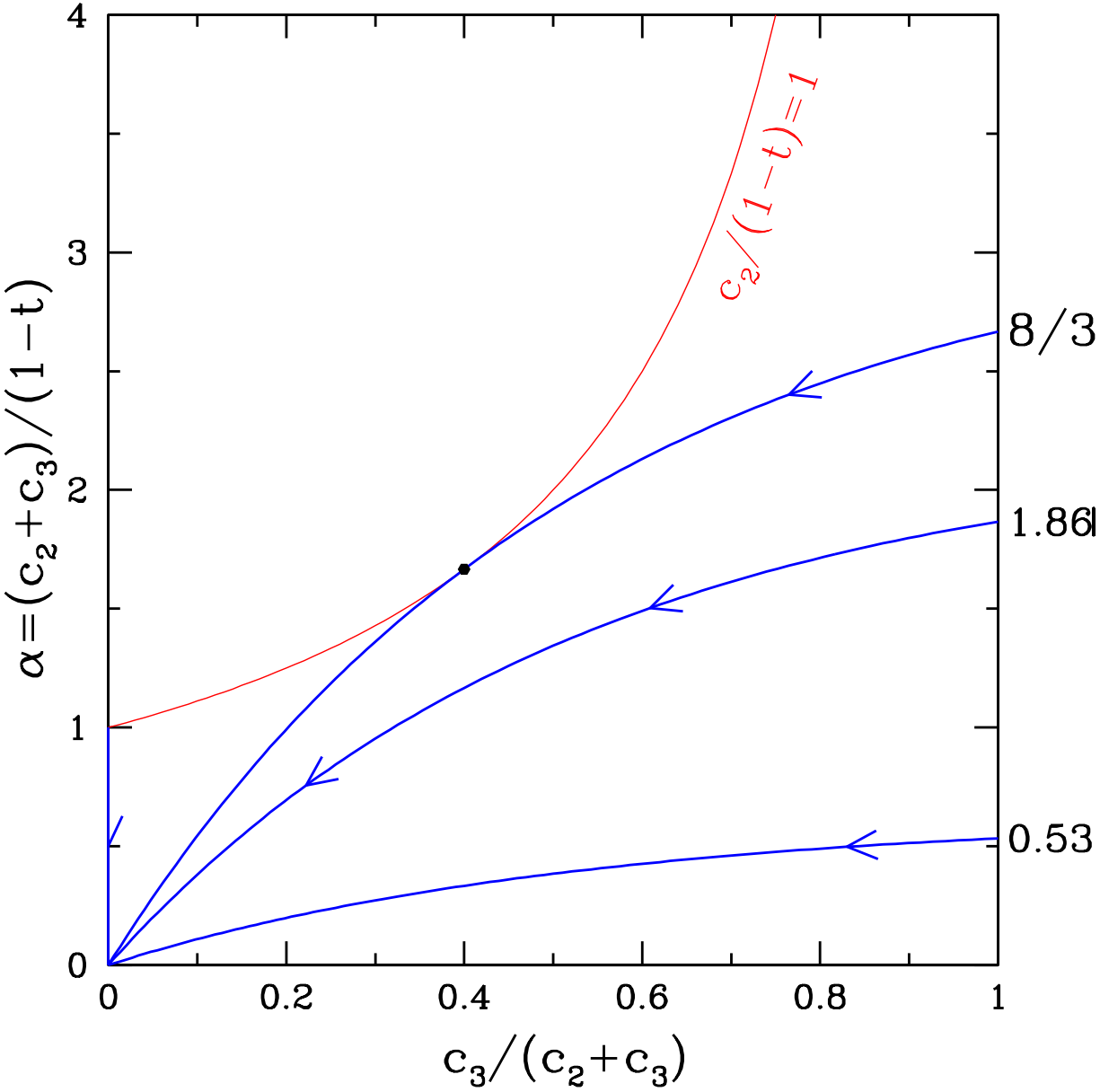


Step left -1 because of one 1-clause elimination

Step right $+d$ [d=binomial variable with proba. $1/(N-T)$ among $C_2(T)$] because of 1- and 2-clauses reduction

$\Rightarrow$ while $\frac{c_2}{1-t} < 1$, C_1 bounded when $N \to +\infty$.

Resolution trajectories



Finite success probability

$\rightarrow$ if initial $\alpha < \alpha_c$ (= $\frac{8}{3}$ for 3-SAT, 1 for 2-SAT), $C_1(T)$ is a. s. bounded and we know its distribution as a function of $c_2(t)$. (If initial $\alpha \geq \alpha_c$, C_1 is no more bounded.)

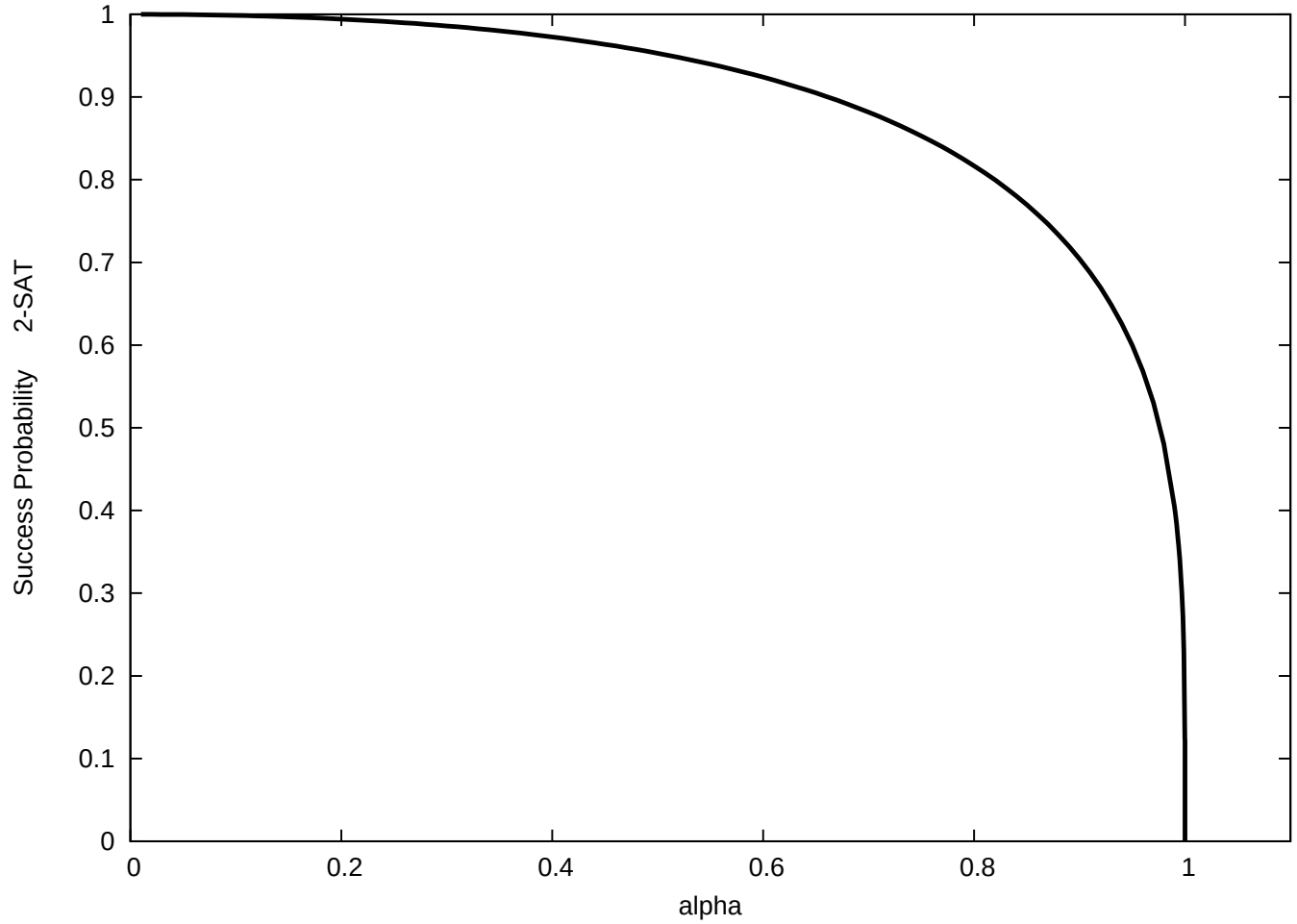
Finite success probability

$\rightarrow$ if initial $\alpha < \alpha_c$ (= $\frac{8}{3}$ for 3-SAT, 1 for 2-SAT), $C_1(T)$ is a. s. bounded and we know its distribution as a function of $c_2(t)$. (If initial $\alpha \geq \alpha_c$, C_1 is no more bounded.)

Conclusion: iff. initial $\alpha < \alpha_c$, probability P of success > 0 :

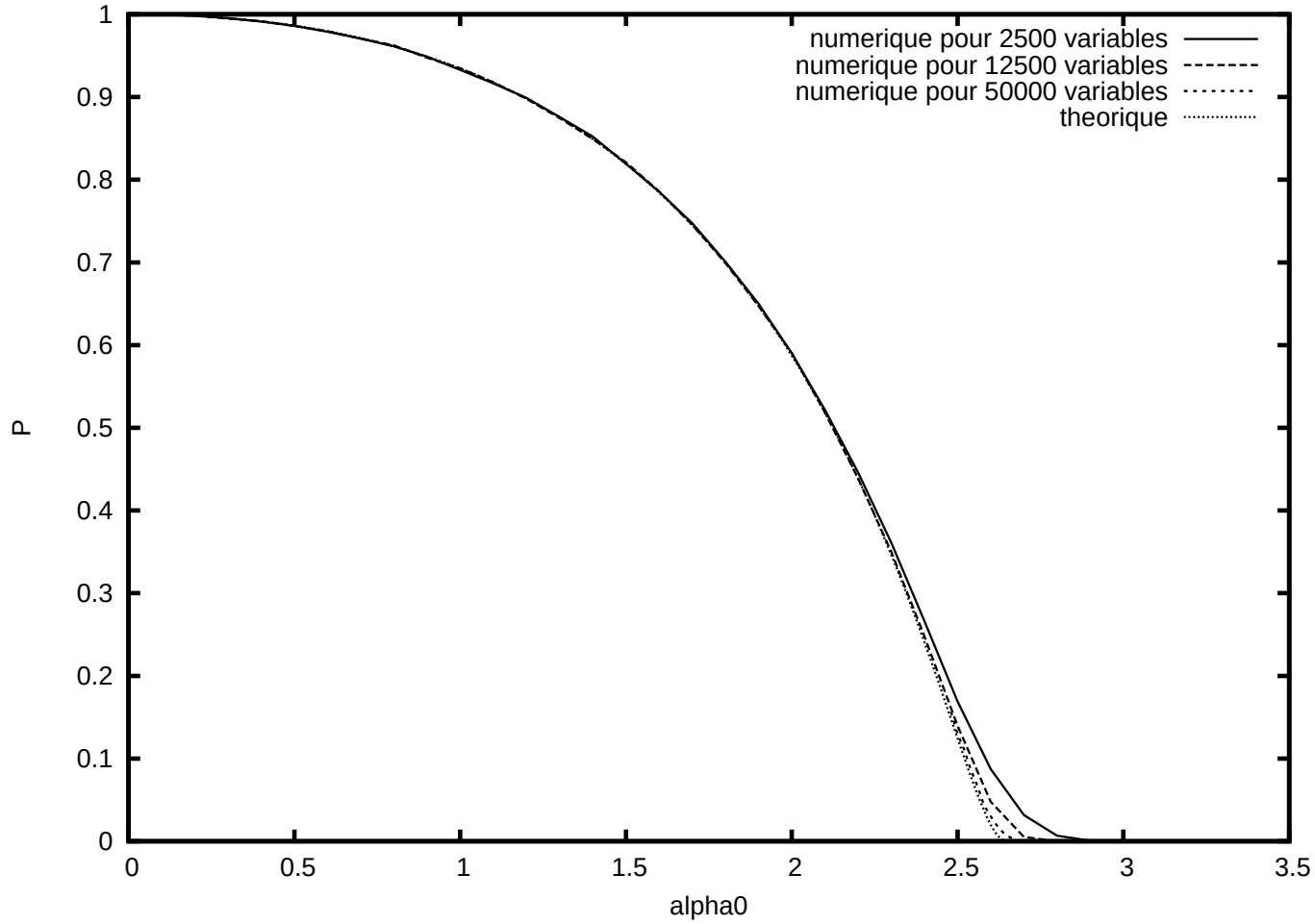
$$\ln(P) = \begin{cases} \frac{3\alpha}{16} - \frac{1}{2\sqrt{\frac{8}{3\alpha}-1}} \arctan\left(\frac{1}{\sqrt{\frac{8}{3\alpha}-1}}\right) & \text{for 3-SAT} \\ \frac{\alpha}{4} + \ln(1-\alpha) & \text{for 2-SAT} \end{cases}$$

Analysis of UC : $P(\alpha)$ for 2-SAT



$-\ln(P(\alpha)) = \Theta\left(\ln|\alpha - \alpha_c|\right)$ when $\alpha \to \alpha_{c-} = 1$.

Analysis of UC : $P(\alpha)$ for 3-SAT



$$-\ln(P(\alpha)) = \Theta\left(\frac{1}{\sqrt{|\alpha-\alpha_c|}}\right) \text{ when } \alpha \to \alpha_{c-} = 8/3.$$

The critical behaviour: a first approach

Critical behaviour

N is large.

For each algorithm without backtracking that uses the unit-propagation principle, $\exists\ \alpha_c$ such that :

- If $\alpha < \alpha_c$, proba of success $P = \Theta(1)$.
- If $\alpha > \alpha_c$, $-\ln(P) \propto N$ (thus $P \to 0$).

Critical behaviour

N is large.

For each algorithm without backtracking that uses the unit-propagation principle, $\exists\ \alpha_c$ such that :

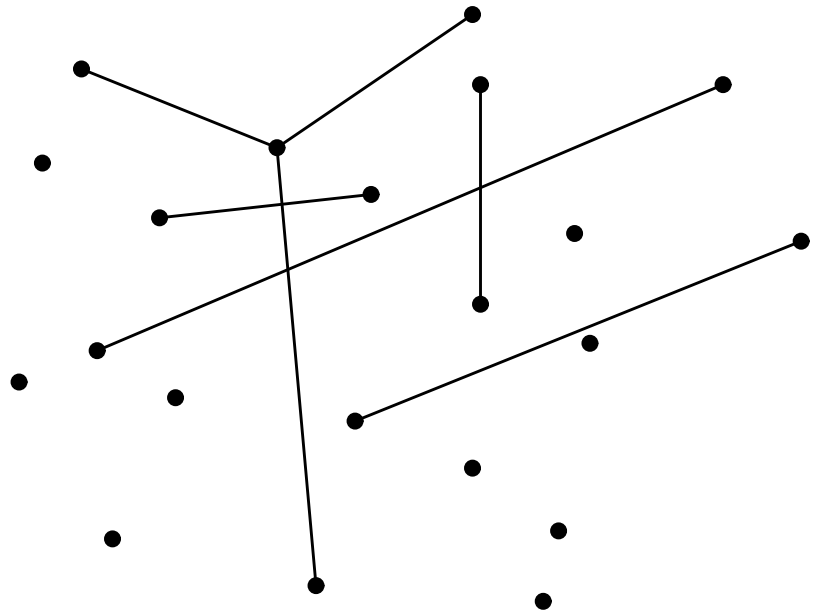
- If $\alpha < \alpha_c$, proba of success $P = \Theta(1)$.
- If $\alpha > \alpha_c$, $-\ln(P) \propto N$ (thus $P \to 0$).

α_c depends on the algorithm.

But do we have critical exponents that do not depend on the algorithm?

Erdös-Rényi random graphs

Fix $c \geq 0$. Given N points, for each set of 2 points, we draw an edge between them with probability $\frac{c}{N}$.



Percolation of Erdös-Rényi random graphs

Size of the largest connected componant:

$$t(c,N) = \begin{cases} \frac{1}{c-1-\ln(c)} \ln(N) & \text{if } c < 1 \\ O(N^{\frac{2}{3}}) & \text{if } c = 1 \\ f(c)\ N & \text{if } c > 1 \end{cases}$$

where $1 - f(c) = e^{-cf(c)}$.

Percolation of Erdös-Rényi random graphs

Size of the largest connected componant:

$$t(c,N) = \begin{cases} \frac{1}{c-1-\ln(c)} \ln(N) & \text{if } c < 1 \\ O(N^{\frac{2}{3}}) & \text{if } c = 1 \\ f(c)\, N & \text{if } c > 1 \end{cases}$$

where $1 - f(c) = e^{-cf(c)}$.

Intuition $\Rightarrow$ $\exists \sigma$ function such that $t(c,N) \sim N^{\frac{2}{3}} \sigma\left(N^{\frac{1}{3}}(c-1)\right)$ with:

$$\begin{cases} \sigma(x) \sim \frac{6 \ln|x|}{x^2} & \text{when } x \to -\infty \\ \sigma(x) \sim 2x & \text{when } x \to +\infty \\ 0 < \sigma(x) < 1 & \forall x \in I\!R \end{cases}$$

Algorithms from the UC family

Here, we are looking for a scaling function:

$$\ln[P(\alpha, N)] = N^{\beta}\pi\Big(N^{\gamma}(\alpha - \alpha_c)\Big)$$

with probably $\gamma = \frac{1}{3}$ by analogy with phase transitions in random graphs: the $\alpha(1-p) = 1$ condition indicates that the graph {variables = vertices, 2-clauses = edges} percolates.

β=?

Algorithms from the UC family

$$\ln[P(\alpha, N)] \quad =? \quad N^{\beta}\pi\Big(N^{\gamma}(\alpha - \alpha_c)\Big)$$

We know ($p > 2/5$-SAT case):

$$-\ln(P) \sim \begin{cases} \frac{cte}{\sqrt{\alpha - \alpha_c}} & \text{when } \alpha \to \alpha_{c-},\ N \to +\infty \\ N\ f(\alpha - \alpha_c) & \text{when } \alpha > \alpha_c,\ N \to +\infty \end{cases}$$

Algorithms from the UC family

$$\ln[P(\alpha, N)] \quad =?\ \ N^{\beta}\pi\Big(N^{\gamma}(\alpha - \alpha_c)\Big)$$

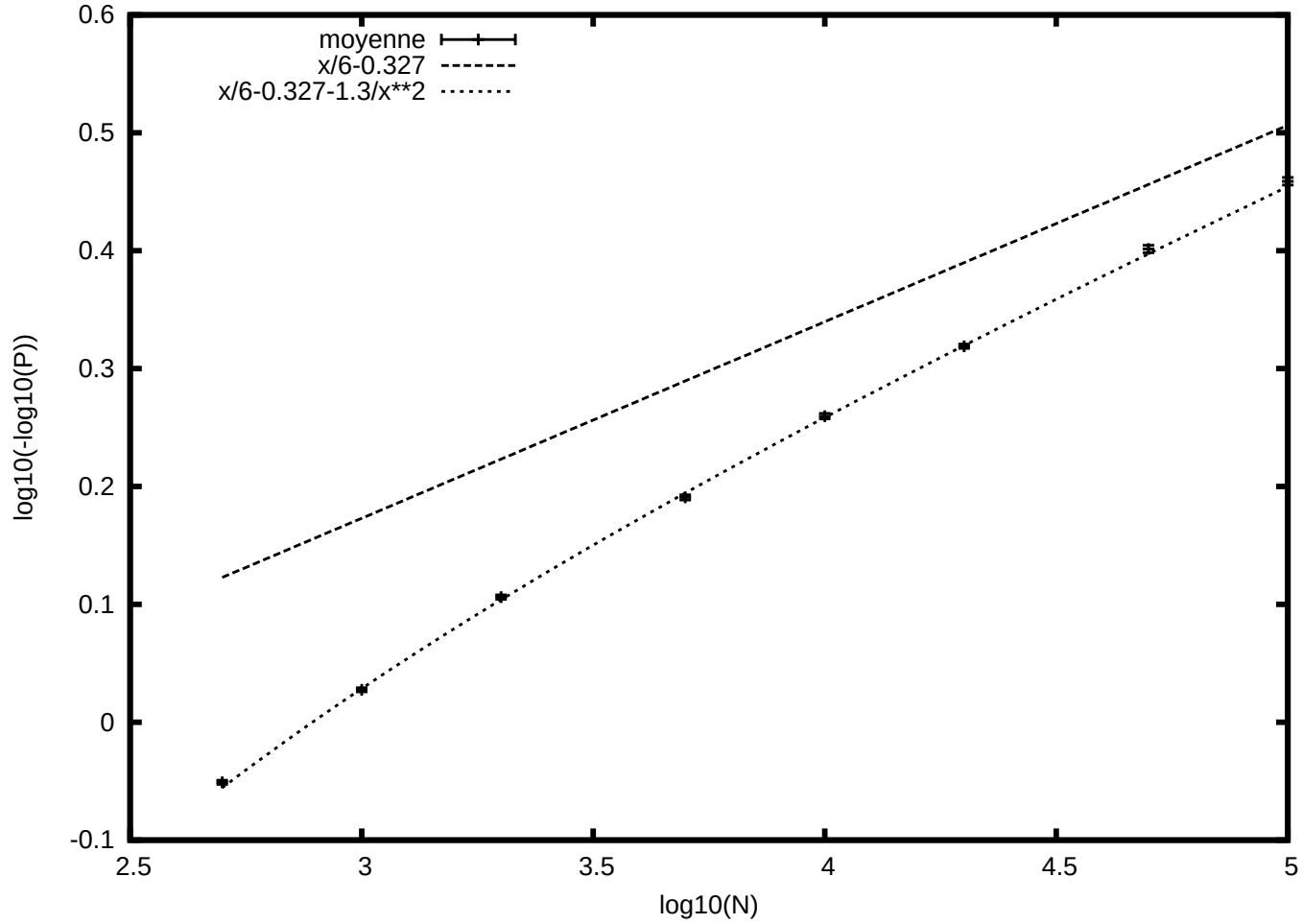
We know ($p > 2/5$-SAT case):

$$-\ln(P) \sim \begin{cases} \frac{cte}{\sqrt{\alpha - \alpha_c}} & \text{when } \alpha \to \alpha_{c-},\ N \to +\infty \\ N\ f(\alpha - \alpha_c) & \text{when } \alpha > \alpha_c,\ N \to +\infty \end{cases}$$

Hence probably $\gamma = \frac{1}{3}$, $\beta = \frac{1}{6}$ et

$$\pi(x) \sim \begin{cases} \frac{cte}{\sqrt{x}} & \text{when } x \to -\infty \\ x^{\frac{5}{2}} & \text{when } x \to +\infty \end{cases}$$

Numerical results (3-SAT)



$$\beta \approx \tfrac{1}{6} \qquad (0.13 < \beta < 0.19).$$

Interpretation

What happens when $\frac{c_2}{1-t} \approx 1$?

Interpretation

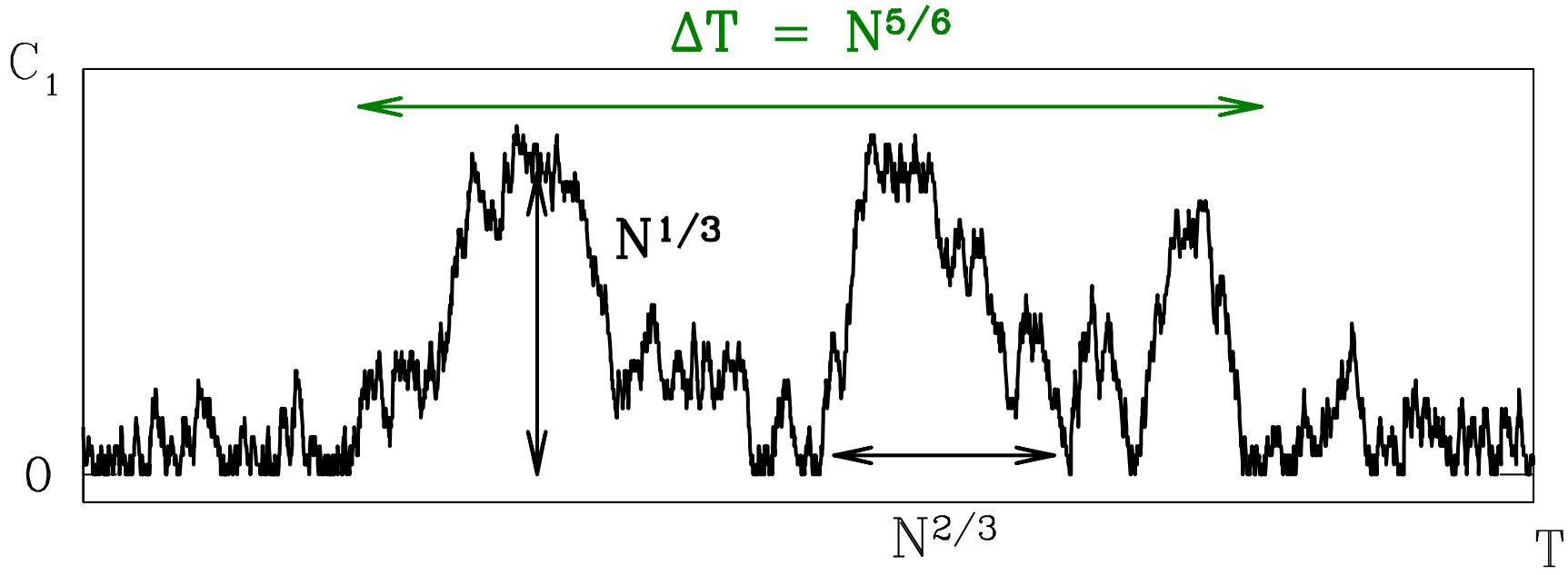
What happens when $\frac{c_2}{1-t} \approx 1$?

The graph where nodes=variables and edges=2-clauses percolates.
Assigning a variable that is in a giant component (size $N^{2/3}$) will produce (successively) $N^{2/3}$ 1-clauses.

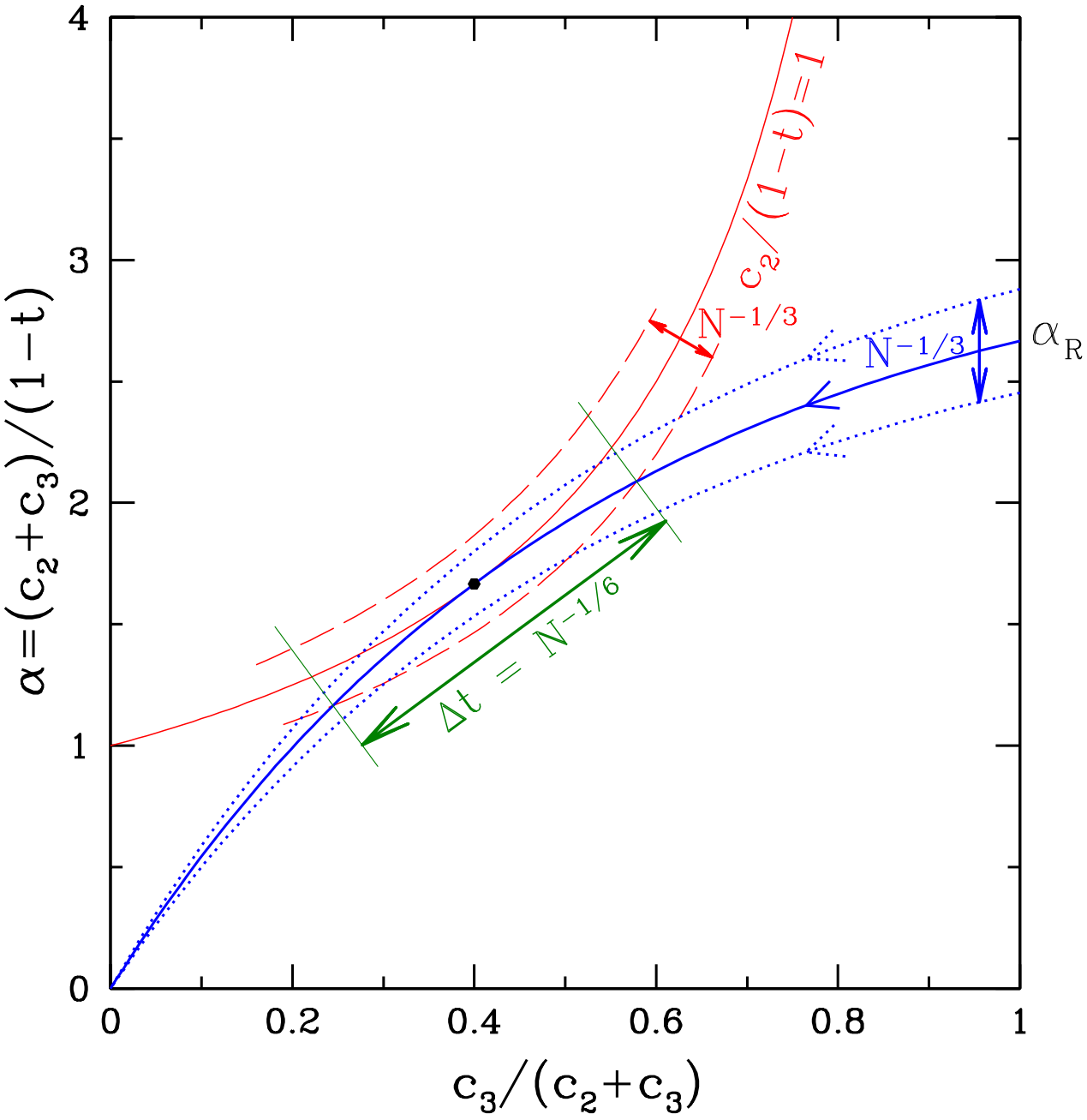
Interpretation

What happens when $\frac{c_2}{1-t} \approx 1$?

The graph where nodes=variables and edges=2-clauses percolates.
Assigning a variable that is in a giant component (size $N^{2/3}$) will produce (successively) $N^{2/3}$ 1-clauses.



Interpretation



Interpretation

Critical window for the random graph where edges are 2-clauses and vertices are variables is $\frac{1}{N^{\frac{1}{3}}}$

Tangential approach $\rightarrow$ we stay in the critical window for $\approx \frac{1}{N^{\frac{1}{6}}}$.

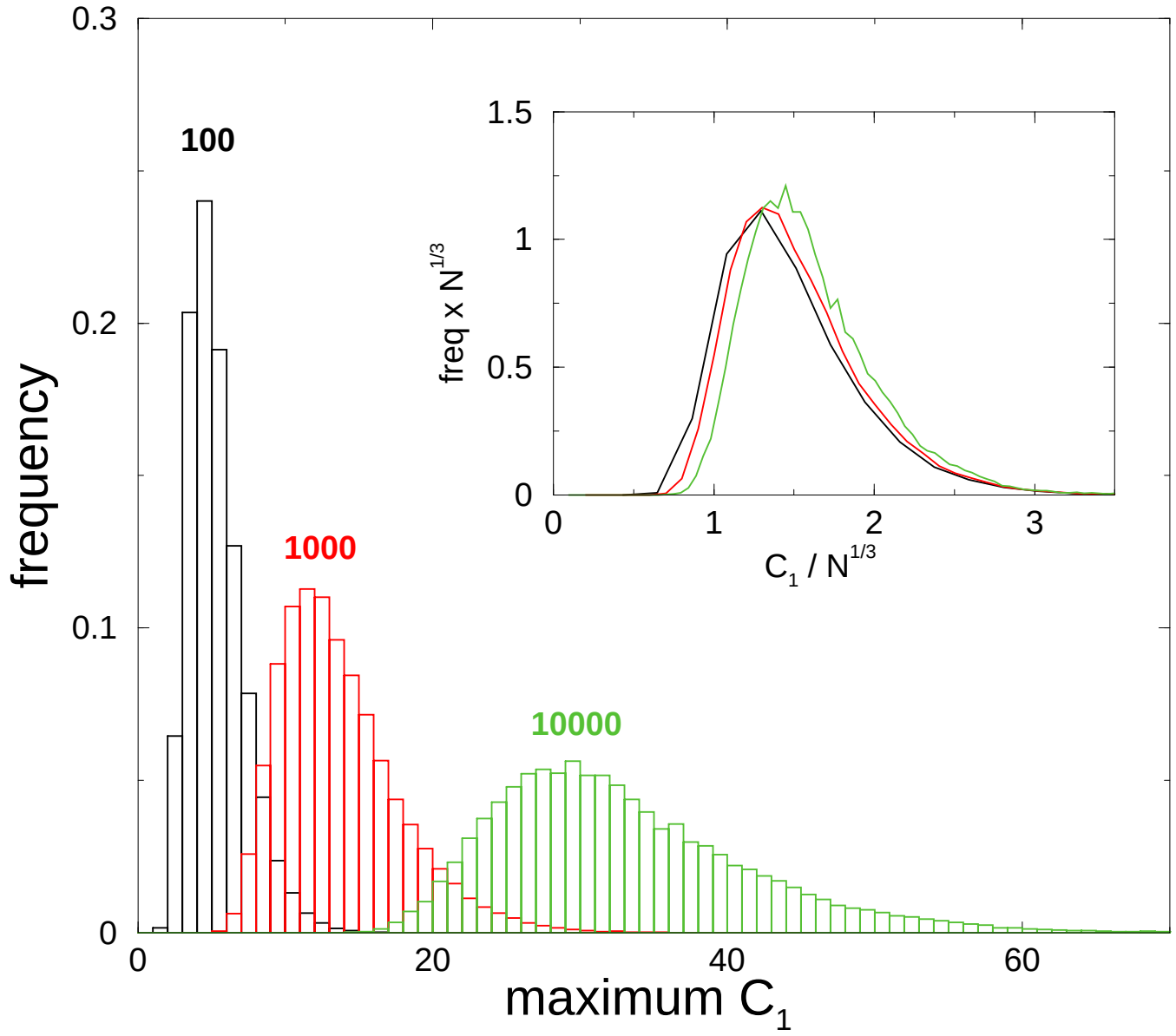
We have to eliminate connected components of size $N^{\frac{2}{3}}$ of the graph that percolates $\rightarrow C_1$ is $\sqrt{N^{\frac{2}{3}}}$.

$$-\ln(P) = \underbrace{-\sum_{0 \leq T \leq N(t^* - \frac{1}{N^{\frac{1}{6}}})} \langle \max(C_1 - 1, 0) \rangle \ln(1 - \frac{1}{2(N-T)})}_{O(1)}$$

$$\underbrace{-\sum_{N(t^* - \frac{1}{N^{\frac{1}{6}}}) \leq T \leq Nt^*}}_{\Theta(N^{\frac{1}{6}})} \underbrace{\langle \max(C_1 - 1, 0) \rangle}_{\Theta(N^{\frac{1}{3}})} \underbrace{\ln(1 - \frac{1}{2(N-T)})}_{\Theta(\frac{1}{N})}$$

hence $\ln[P(\alpha = \alpha_c)] \approx N^{\frac{1}{6}}$.

Interpretation



Critical behaviour: exact computation

Starting point

We know that, at the success/failure transition, C_1 diverges: we have to study its distribution.

Generating function of C_1:

$$p_N(T,x) := \sum_{C_1=0}^{+\infty} x^{C_1} P_N(C_1,T)$$

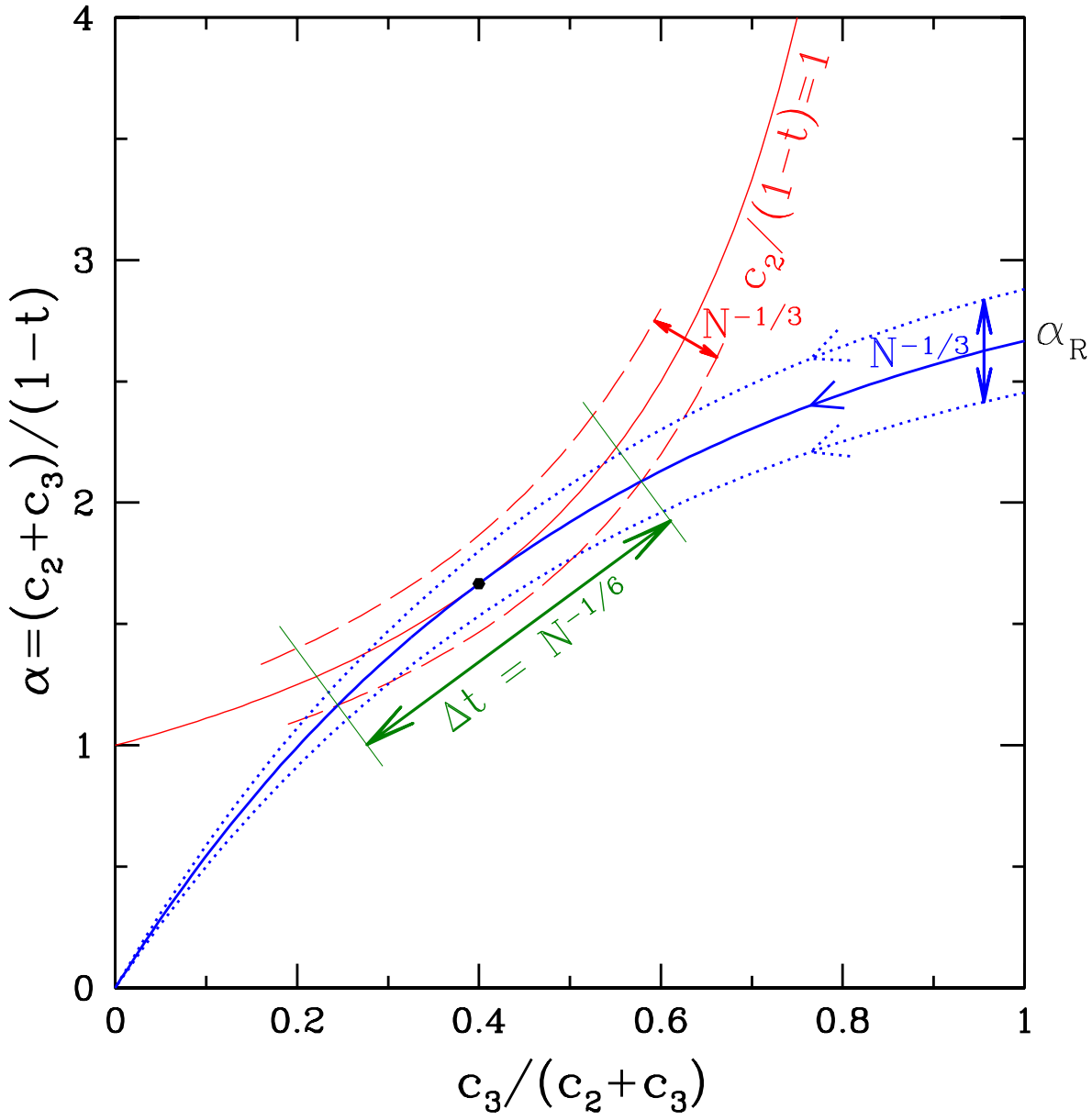
Matrix $H(C_1' \leftarrow C_1)$ becomes:

$$p_N(T+1,x) = \left(1+\frac{x-1}{N-T}\right)^{C_2(T)} \times \left(\frac{1}{A}p_N(T,A) + (1-\frac{1}{A})p_N(T,0)\right)$$

with $A := \frac{1}{2(N-T)} + (1-\frac{1}{N-T})x$.

Exact, all information but C_2 ; relevant for all algorithms that apply the Unit Propagation rule

Zoom in for 3-SAT



Zoom in

Zoom in around the contact point $\frac{c_2}{1-t} = 1$. In the case of the UC algorithm:

- $\alpha = \frac{8}{3}(1 + \epsilon_0 N^{-\theta})$
- $t = \frac{1}{2}(1 + t_0 N^{-\tau})$
- $C_2(T) = 4T(1 - \frac{T}{N})^2 + \mathcal{O}(\sqrt{N})$

θ, τ to be chosen to study the first non-vanishing order.

Computations...

We substitute in

$$p_N(T+1,x) = \left(1+\frac{x-1}{N-T}\right)^{C_2(T)} \times \left(\frac{1}{A}p_N(T,A) + (1-\frac{1}{A})p_N(T,0)\right)$$

with $A := \frac{1}{2(N-T)} + (1-\frac{1}{N-T})x$.

θ and τ are free. If we ask that all first-order non-vanishing terms are of the same order:

$$\boxed{\theta = \gamma = \gamma_0 = \tfrac{1}{3}, \lambda = \tau = \tfrac{1}{6}}.$$

PDE for f

And (for 3-SAT) probability density function $f(c, t_0)$ of $c = C_1/N^{1/3}$ given by:

$$\boxed{\frac{1}{2}\frac{\partial^2 f}{\partial c^2} + (t_0^2 - \epsilon_0)\frac{\partial f}{\partial c} + (c - \overline{c})f = 0}$$

with boundary conditions:

$$\partial_c f(0, t_0) + (t_0^2 - \epsilon_0) f(0, t_0) = 0$$

PDE for f

And (for 3-SAT) probability density function $f(c, t_0)$ of $c = C_1/N^{1/3}$ given by:

$$\boxed{\frac{1}{2}\frac{\partial^2 f}{\partial c^2} + (t_0^2 - \epsilon_0)\frac{\partial f}{\partial c} + (c - \overline{c})f = 0}$$

with boundary conditions:

$$\partial_c f(0, t_0) + (t_0^2 - \epsilon_0) f(0, t_0) = 0$$

We solve for f:

$$\boxed{f(c, t_0) \propto e^{-(t_0^2 - \epsilon_0)c} \mathrm{Ai}[\sqrt[3]{2}c + z(t_0^2 - \epsilon_0)]}$$

where $z(x)$ is the reciprocal function of $x(z) = \sqrt[3]{2}\frac{\mathrm{Ai}'(z)}{\mathrm{Ai}(z)}$.

Final result

$\ln P_{\text{succ}} = -N^{1/6}\phi(t_0 = +\infty) + \mathcal{O}(1)$ with $\phi(t_0 = -\infty) = 0$ and $\partial_{t_0}\phi(t_0) = \overline{c}(t_0) \rightsquigarrow$

$$\boxed{-\ln P_{\text{succ}}((1+\epsilon_0)\alpha_c, N) = N^{1/6}\Phi(\epsilon_0 N^{1/3}) + \mathcal{O}(1)}$$

with

$$\boxed{\Phi(\epsilon_0) = \frac{1}{4}\int_{-\epsilon_0}^{+\infty} \frac{\mathrm{d}x}{\sqrt{\epsilon_0 + x}} \left[x^2 - 2^{2/3}\, z(x)\right]}$$

E.g., for 3-SAT at critical initial α, $\boxed{\Phi(0) \approx 1.1277}$.

Generality of this result

This computations depends on the algorithm only through C_2. Actually, if we zoom in around the contact point t_{A} for another algorithm, we get:

$$\frac{c_2}{1-t} = 1 + b(t - t_{\mathrm{A}})^2 \text{ when } t \to t_{\mathrm{A}}$$

computation is the same, but with a different value of b and t_A. We find:

$$\Phi_{\mathrm{A}}(\epsilon_0) = r_{\mathrm{A}}^{\Phi}\ \Phi(r_{\mathrm{A}}^{\epsilon}\ \epsilon_0)$$

where the r_{A}'s are functions of b and t_A.

Generality of this result

This computations depends on the algorithm only through C_2. Actually, if we zoom in around the contact point t_{A} for another algorithm, we get:

$$\frac{c_2}{1-t} = 1 + b(t - t_{\mathrm{A}})^2 \text{ when } t \to t_{\mathrm{A}}$$

computation is the same, but with a different value of b and t_A. We find:

$$\Phi_{\mathrm{A}}(\epsilon_0) = r_{\mathrm{A}}^{\Phi}\ \Phi(r_{\mathrm{A}}^{\epsilon}\ \epsilon_0)$$

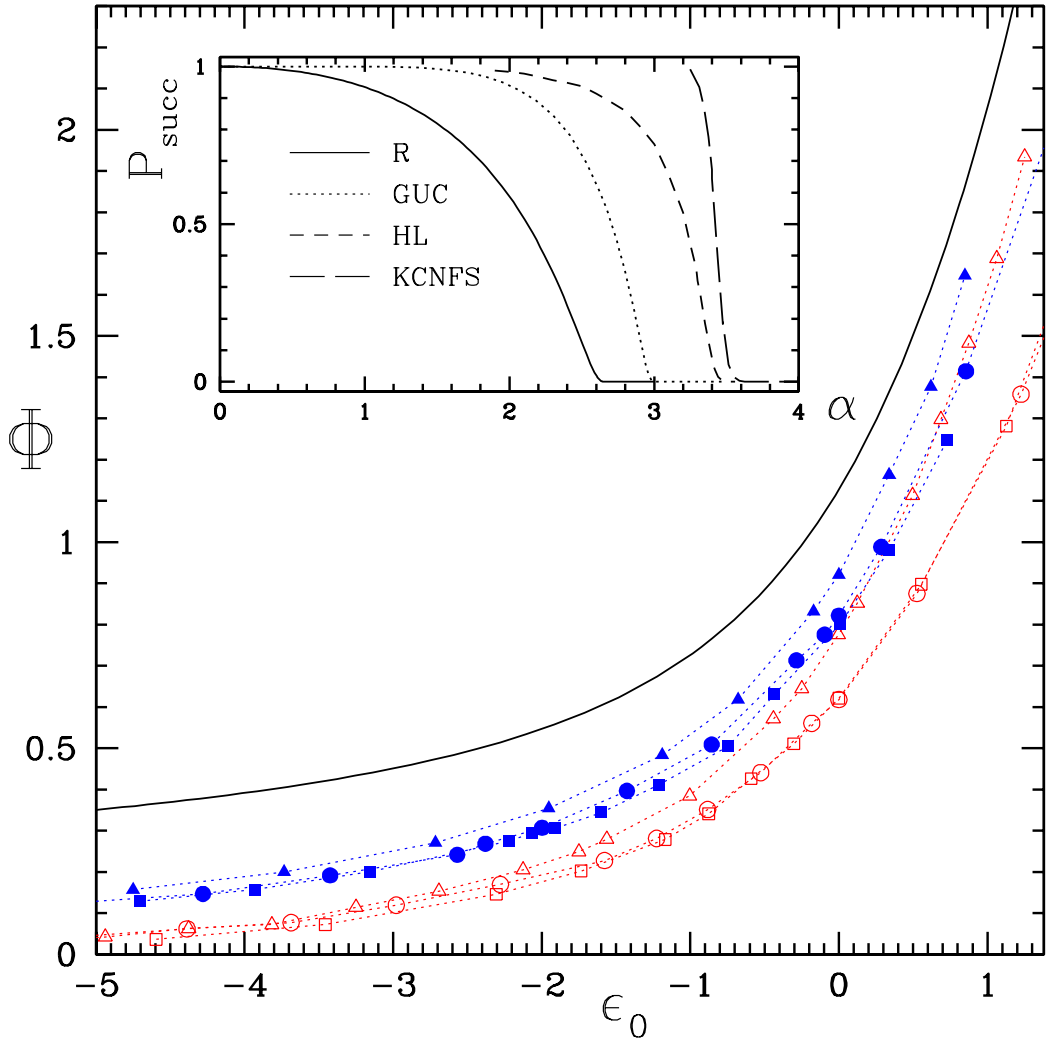
where the r_{A}'s are functions of b and t_A.

Non generical case (possible if $K > 3$-SAT):

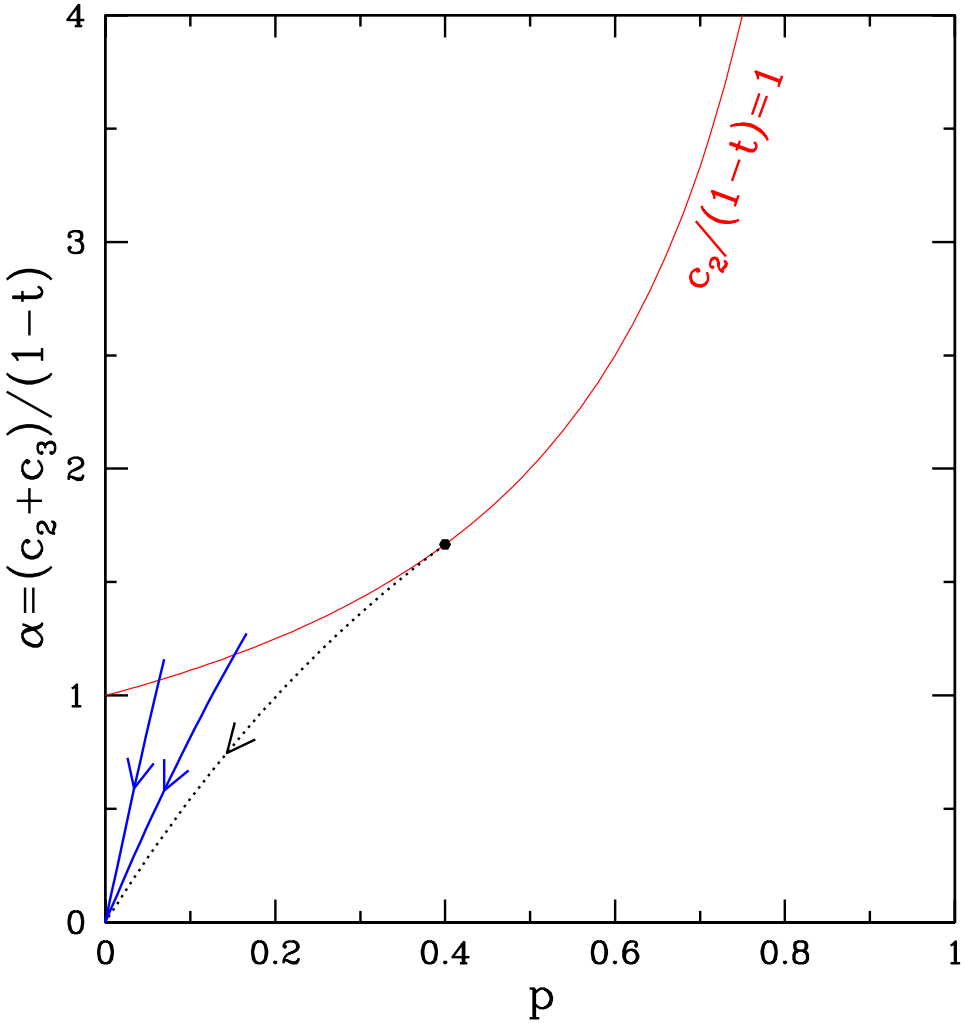
$$\frac{c_2}{1-t} = 1 + b(t - t_{\mathrm{A}})^{4,6,\ldots}$$

Generality of this result

The computation depends on the distribution of formulae: the scaling function Φ may change. But the exponents remain the same because of the robustness of the distribution of sizes of the connected components of a random graph.



Here the resolution trajectory cuts the $\alpha(1-p)=1$ line, instead of becoming tangent to it:



The 2-SAT case

Therefore, $t = 0 + t_0 N^{-\tau}$ with now $\tau = 1/3$ instead of $1/6$, and the time-derivative term in the PDE for f is relevant:

$$\boxed{\frac{1}{2}\frac{\partial^2 f}{\partial c^2} + \beta(p)(t_0 - \epsilon_0)\frac{\partial f}{\partial c} + \frac{1}{2}(c - \overline{c})f = \frac{\partial f}{\partial t_0}}$$

The 2-SAT case

Therefore, $t = 0 + t_0 N^{-\tau}$ with now $\tau = 1/3$ instead of $1/6$, and the time-derivative term in the PDE for f is relevant:

$$\boxed{\tfrac{1}{2}\frac{\partial^2 f}{\partial c^2} + \beta(p)(t_0 - \epsilon_0)\frac{\partial f}{\partial c} + \tfrac{1}{2}(c - \overline{c})f = \frac{\partial f}{\partial t_0}}$$

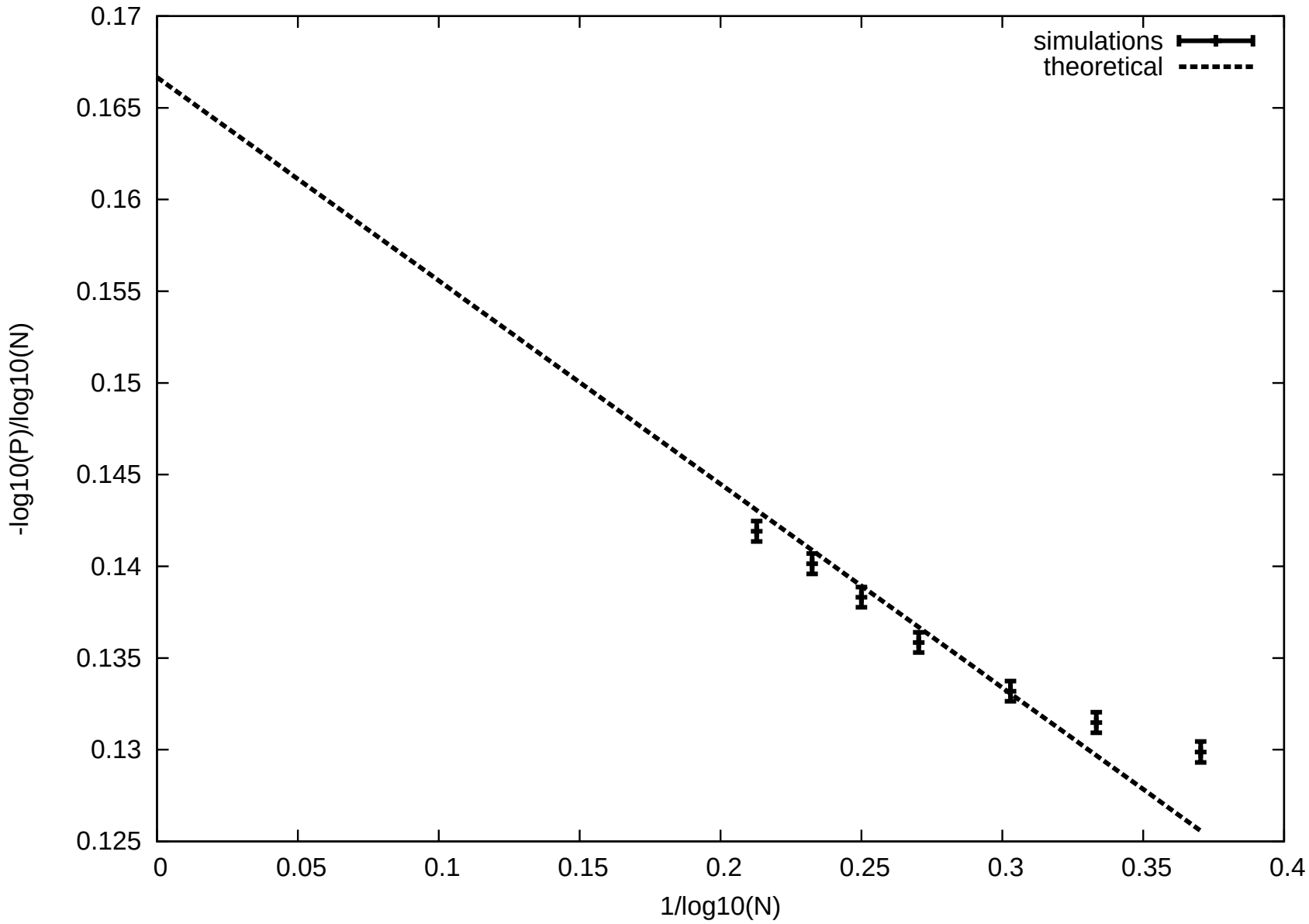
which yields a logarithmic behaviour:

$$\ln P_{\text{succ}} = -\frac{\ln N}{12\beta(p)} + const + o(1)$$

and β vanishes as $p \to 2/5$.

Since we don't know how to solve analytically the PDE, we have to solve it numerically to get the constant. E.g. $P_{\text{succ}} \sim N^{-1/12} \exp\left(1/4 - 0.24370(1)\right)$ for 2-SAT,
$P_{\text{succ}} \sim N^{-1/6} \exp\left(5/16 - \frac{\ln 2}{4} - 0.20157(1)\right)$ for 2+1/4-SAT

The 2-SAT case



The 2+2/5-SAT case

Finally, for $p = \frac{2}{5}(1 + \epsilon_p N^{-1/6})$ and at α close to $\alpha_c(\frac{2}{5}) = \frac{5}{3}$:

$$\alpha = \frac{5}{3}(1 + \frac{2}{3}\epsilon_p N^{-1/6} + \epsilon_\alpha N^{-1/3})$$

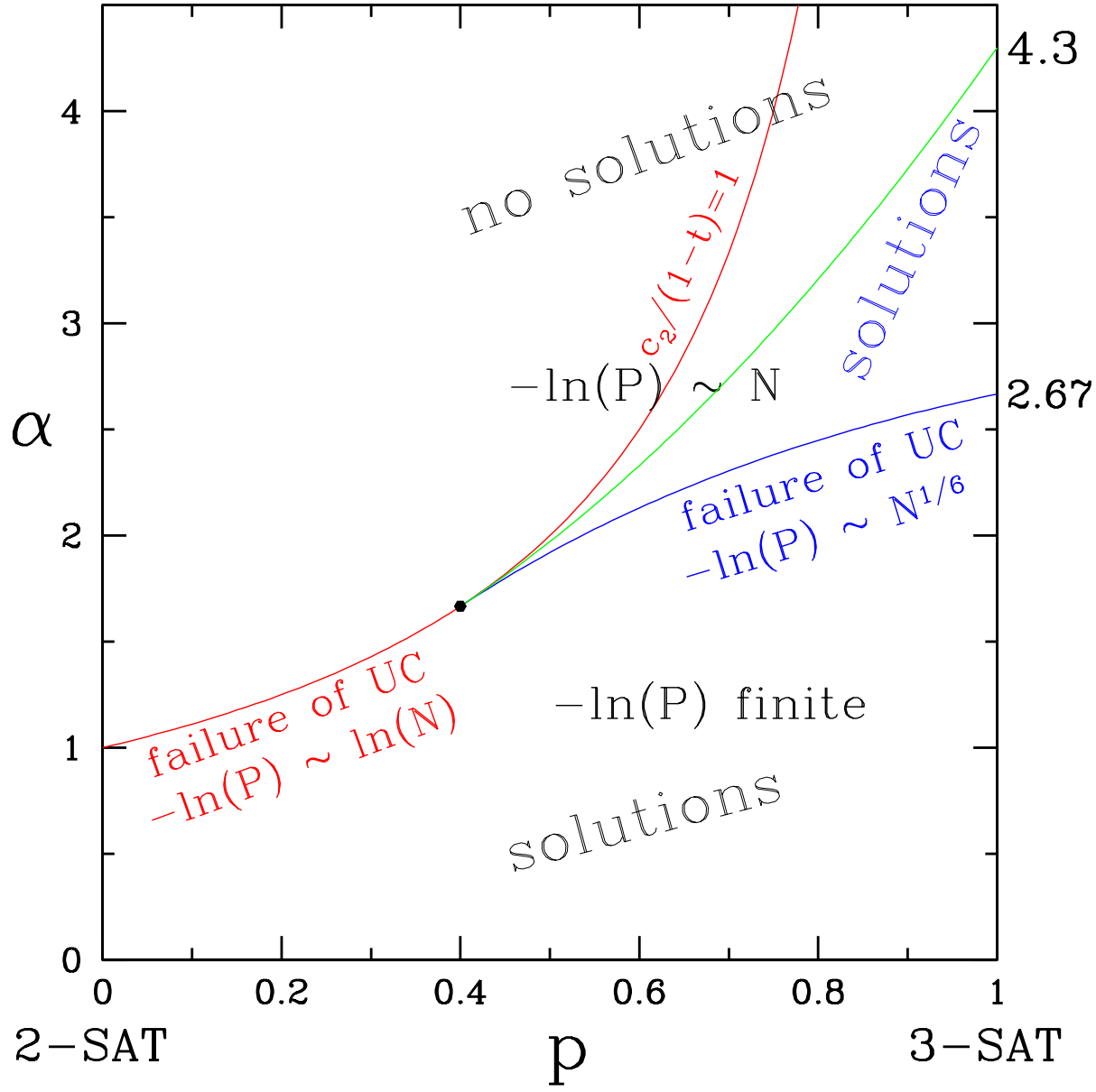
there is a transition between the 2- and 3-SAT cases:

$$\ln(P_{\text{succ}}) = N^{1/6}\Phi_1(\epsilon_p, \epsilon_\alpha) + \frac{1}{24}\ln(N) + \mathcal{O}(1)$$

with $\Phi_1(\epsilon_p) \propto \frac{-\ln \epsilon_p}{\epsilon_p}$ for $\epsilon_\alpha = \frac{4}{9}\epsilon_p^2$ and $\epsilon_p \to -\infty$ (following the $\frac{c_2}{1-t} = 1$ line),
thus the 2-SAT behaviour is recovered if $\epsilon_p \approx -N^{1/6}$.

$\Phi_1(0,0) = \Phi_1(+\infty)2^{-\frac{5}{6}} = 0.6329$ is expressed as an integral with Airy Ai functions ($\Phi_1(\epsilon_p) \to 1.1277$ as $\epsilon_p \to +\infty$).

Phase space for algo. UC on $2 + p$-SAT



Conclusion

Conclusion

- The framework of simple algorithms for solving random $2 + p$-SAT formulae is conceptually simple and mathematically tractable thanks to the independence of random variables (no correlations in the disorder)
- But it yields a rather rich behaviour with several critical exponents and windows, and we can even compute some of the scaling functions and non-universal coefficients.

Conclusion

- The critical exponents are expected to be the same for all algorithms that study a random NP problem (without too much correlations) and that use the Unitary Propagation rule (but they have to use it!).
- What would be the results with strongly different correlations on the random formulae (e.g. finite dimension, planar graphs, power-law degree distributions...)?
- Can this understanding of the failure cause of our algorithms help us in designing better algorithms (that find solutions up to α_{static})?