Analytic description of an optimization algorithm: Beliefs Inspired Decimation

Federico Ricci-Tersenghi

Physics Department
Sapienza University of Rome

in collaboration with
Andrea Montanari and Guilhem Semerjian

Proc. Allerton conference, 352 (2007)
J. Stat. Mech., P09001 (2009)

Les Houches, March 10, 2010

Outline of the talk

1. Constraint Satisfaction Problems (CSP) and random CSP (rCSP)
2. Phase transitions in rCSP
3. Solving algorithms
4. Our algorithm
5. Analytical results for random *k*-XORSAT
6. Numerical results for random *k*-SAT

Constraint satisfaction problems

- Assign *N* variables to satisfy *M* constraints
- Examples with binary variables $\sigma = \pm 1$
  and constraints involving *k* variables
  - XORSAT (parity checks)

$$\sigma_{i_1^a} \dots \sigma_{i_k^a} = J^a \qquad a = 1, \dots, M$$

  - SAT

$$(\sigma_{i_1^a}, \dots, \sigma_{i_k^a}) \neq (J_1^a, \dots, J_k^a) \qquad a = 1, \dots, M$$

Random CSP

- For each constraint
  - choose *k* variables at random

    i.e. $(i_1^a, \ldots, i_k^a)$ is a random *k*-tuple

  - choose randomly all the "couplings":

    J^a for XORSAT or $(J_1^a, \ldots, J_k^a)$ for SAT

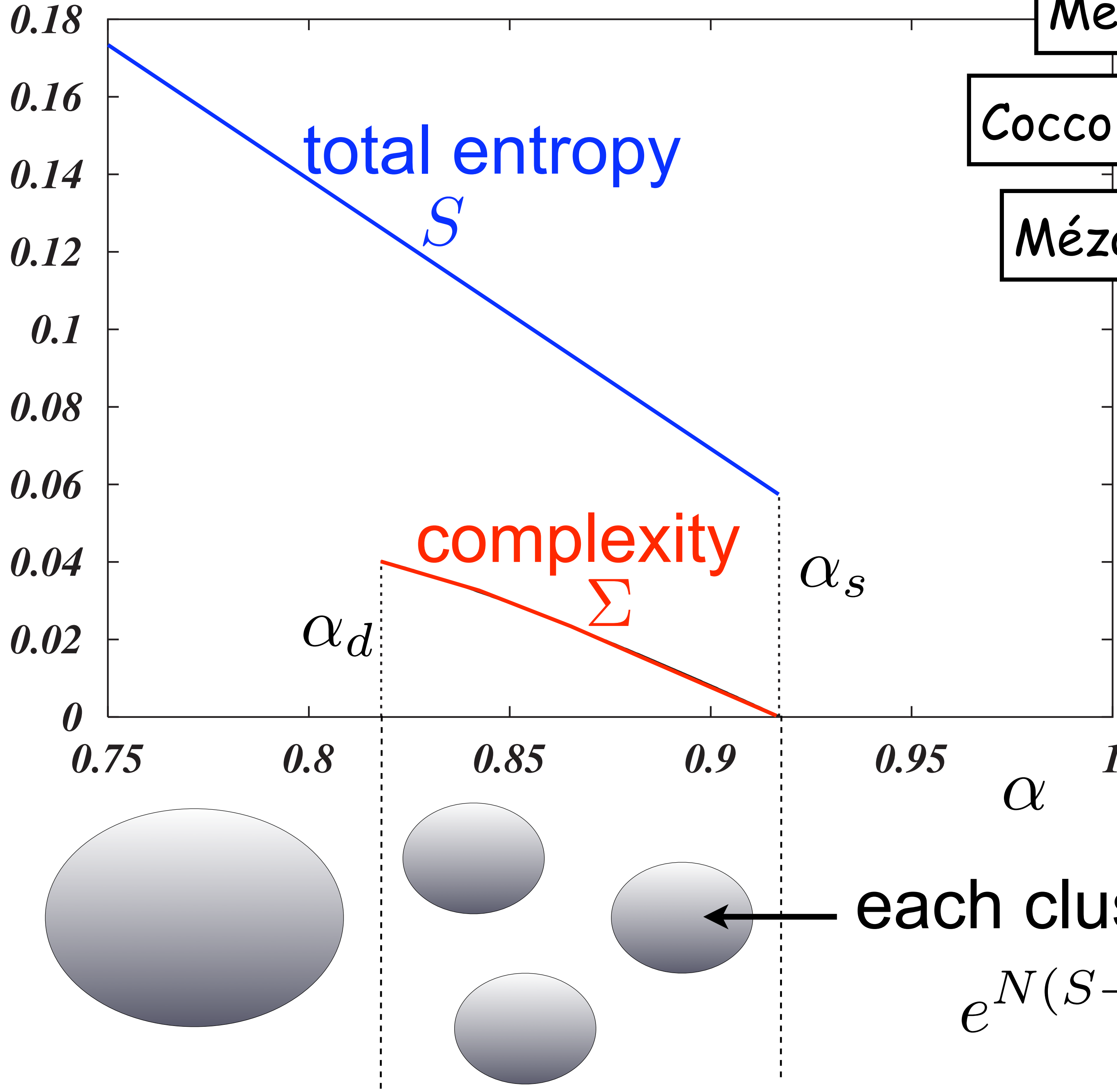
- Relevant parameter $\boxed{\alpha = M/N}$

Phase transitions in random *k*-XORSAT

Mézard, Parisi, Zecchina

Cocco, Dubois, Mandler, Monasson

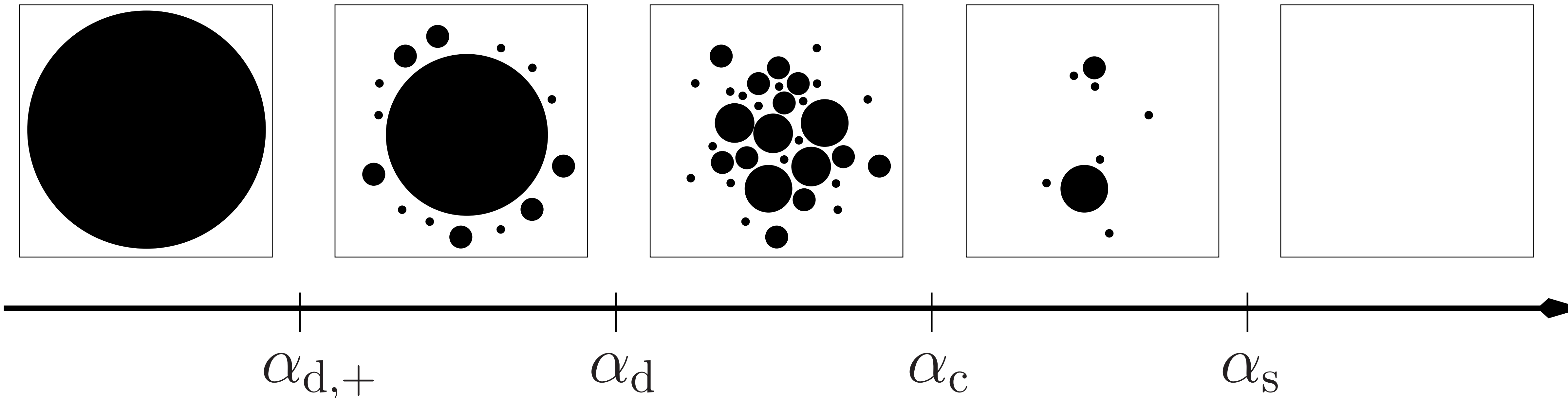
Mézard, Ricci-Tersenghi, Zecchina



clustering phase transition

each cluster contains $e^{N(S-\Sigma)}$ solutions

More phase transitions in random *k*-SAT (*k* > 3)



SAT/UNSAT

condensation

clustering

Krzakala, Montanari, Ricci-Tersenghi, Semerjian, Zdeborova

Solving algorithms

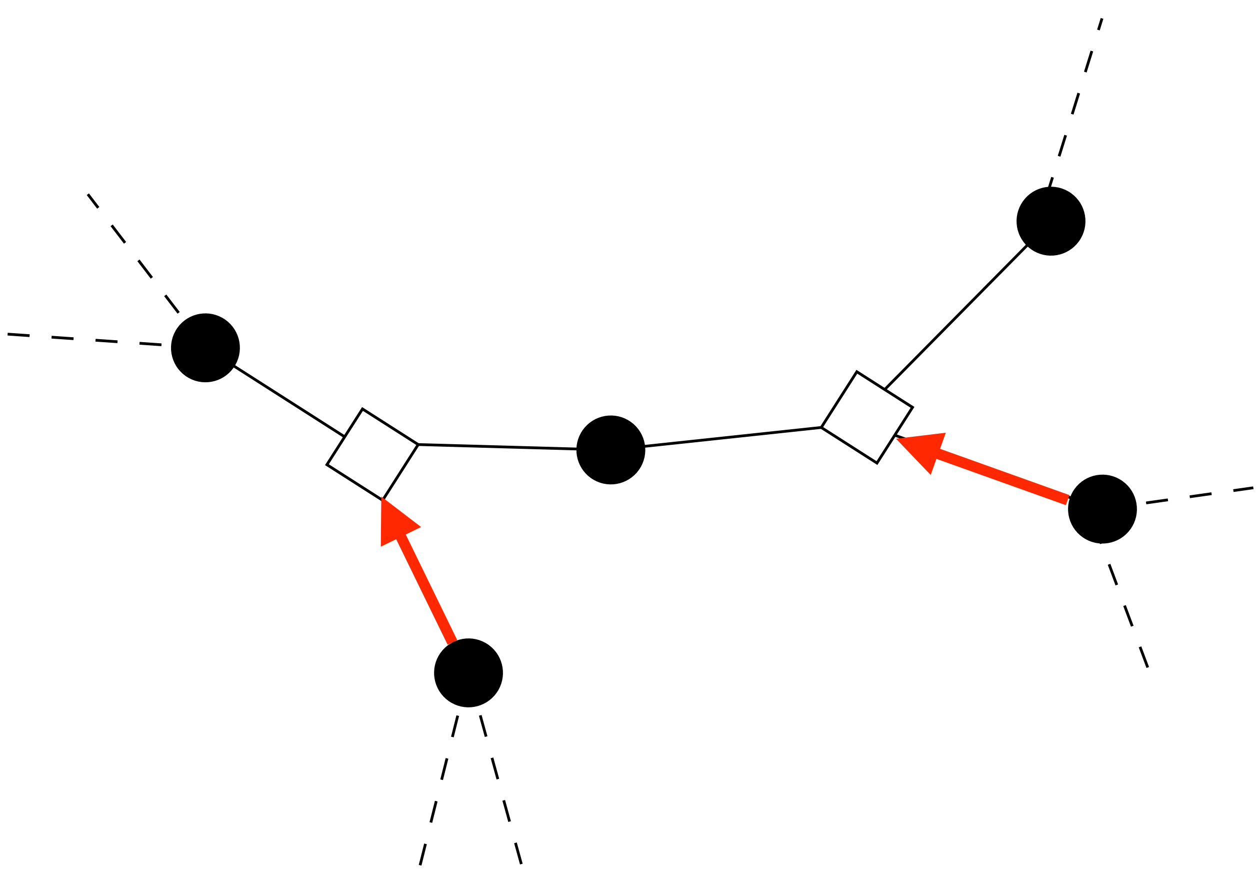
- Procedure to find a solution
(we assume there is at least one)
- Not interested in proving UNSATisfiability
(we work with $\alpha < \alpha_s$)
- A much harder problem:
sampling solutions **uniformly**

Two broad classes of solving algortihms

- **Local search**
  (biased) random walks in
  the space of configurations
  E.g. Monte Carlo, WalkSAT, FMS, ChainSAT, ...
- **Sequential construction**
  at each step a variable is assigned
  E.g. UCP, GUCP, BP, SP, ...
  - the order of assignment of variables
  - the information used to assign variables

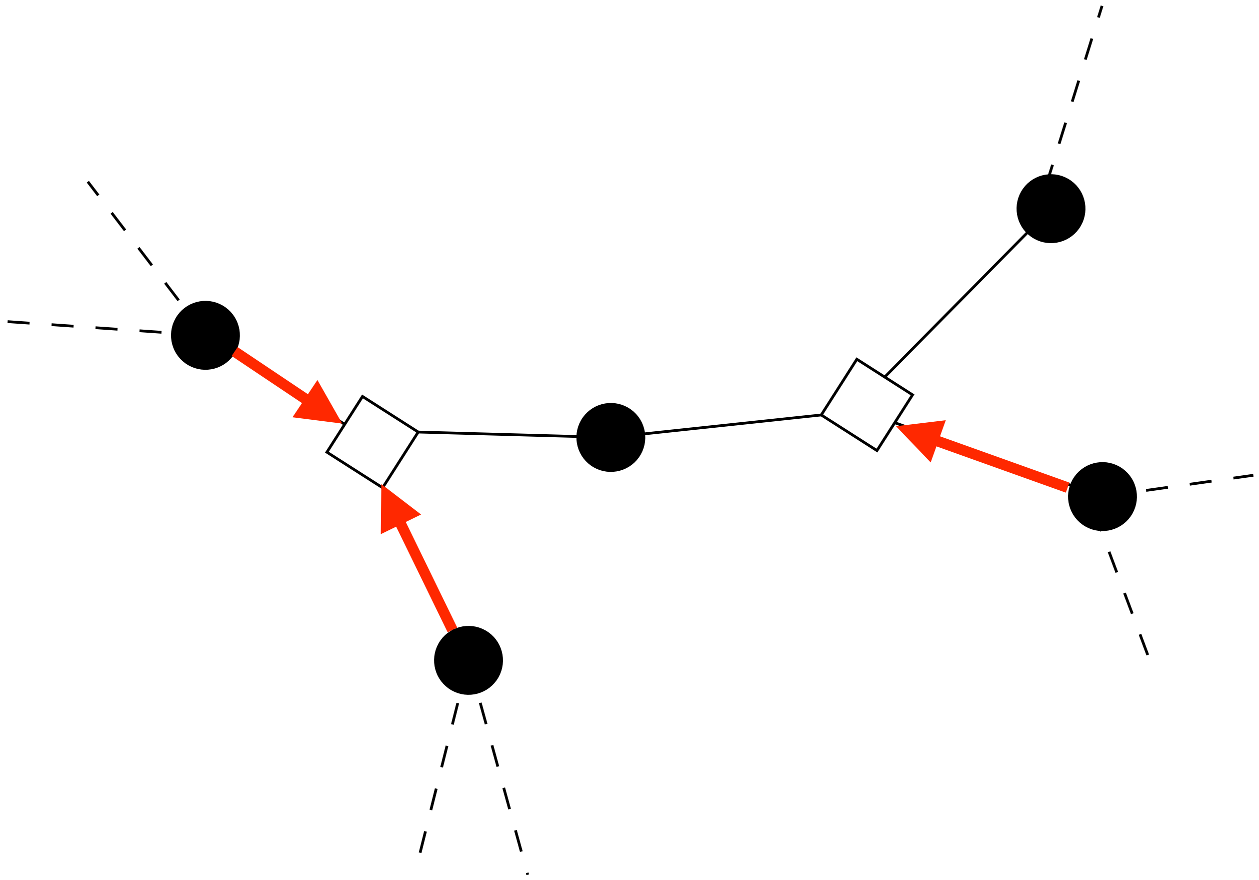
Unit Clause Propagation (UCP) Warning Propagation (WP)

- Looking for solutions
  - ➡ all constraints must be satified
  - ➡ variables may be forced to take a **unique** value (frozen variables)



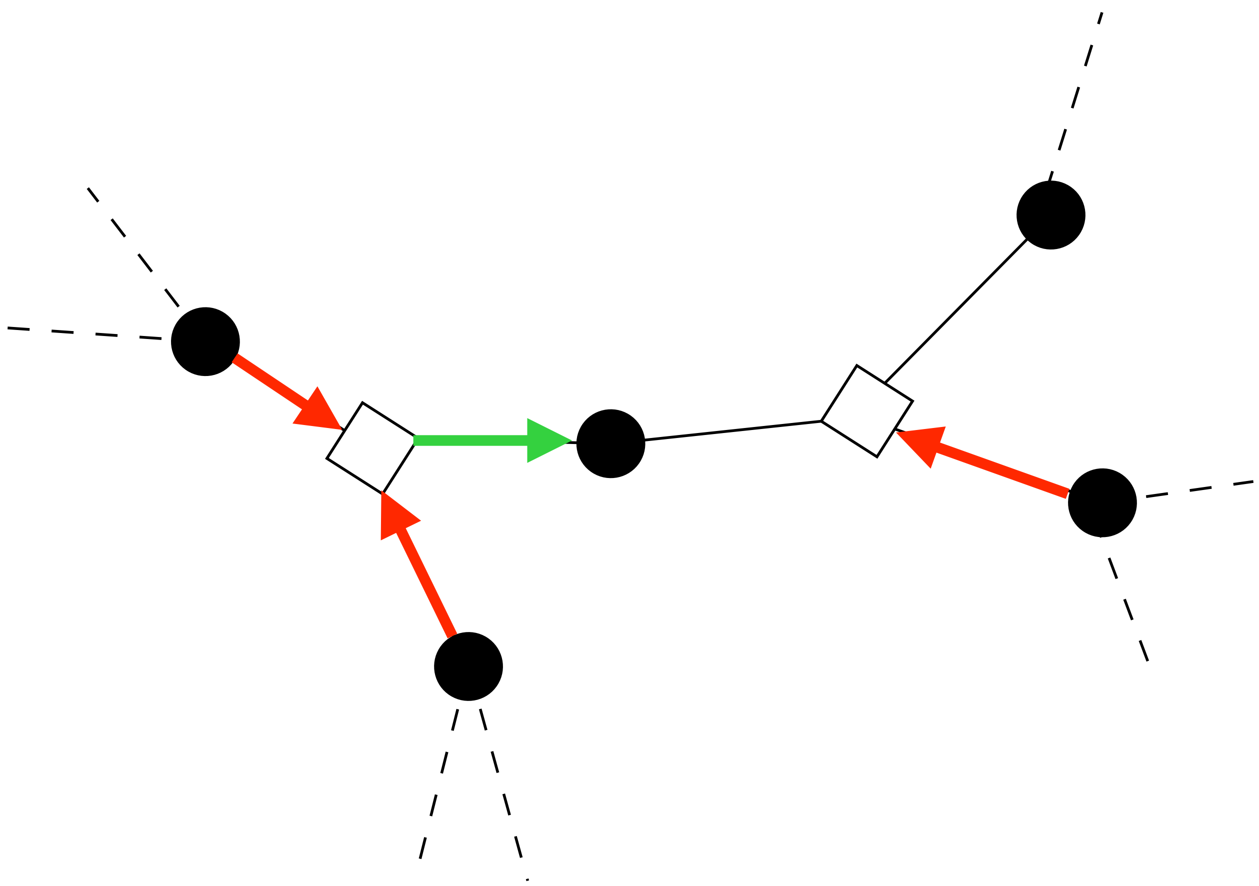
Unit Clause Propagation (UCP) Warning Propagation (WP)

- Looking for solutions
  - ➡ all constraints must be satified
  - ➡ variables may be forced to take a **unique** value (frozen variables)



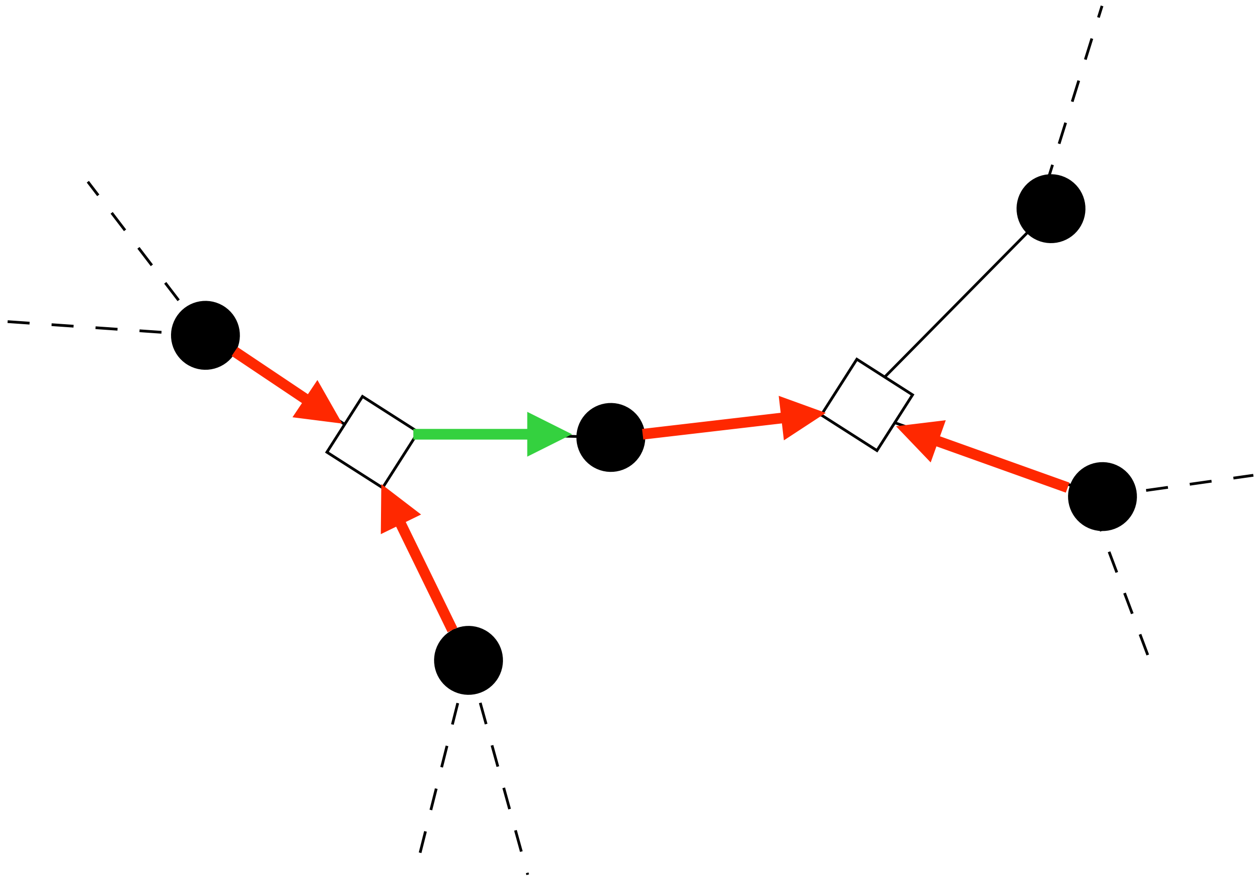
Unit Clause Propagation (UCP)
Warning Propagation (WP)

- Looking for solutions
  - ➡ all constraints must be satified
  - ➡ variables may be forced to take a **unique** value (frozen variables)



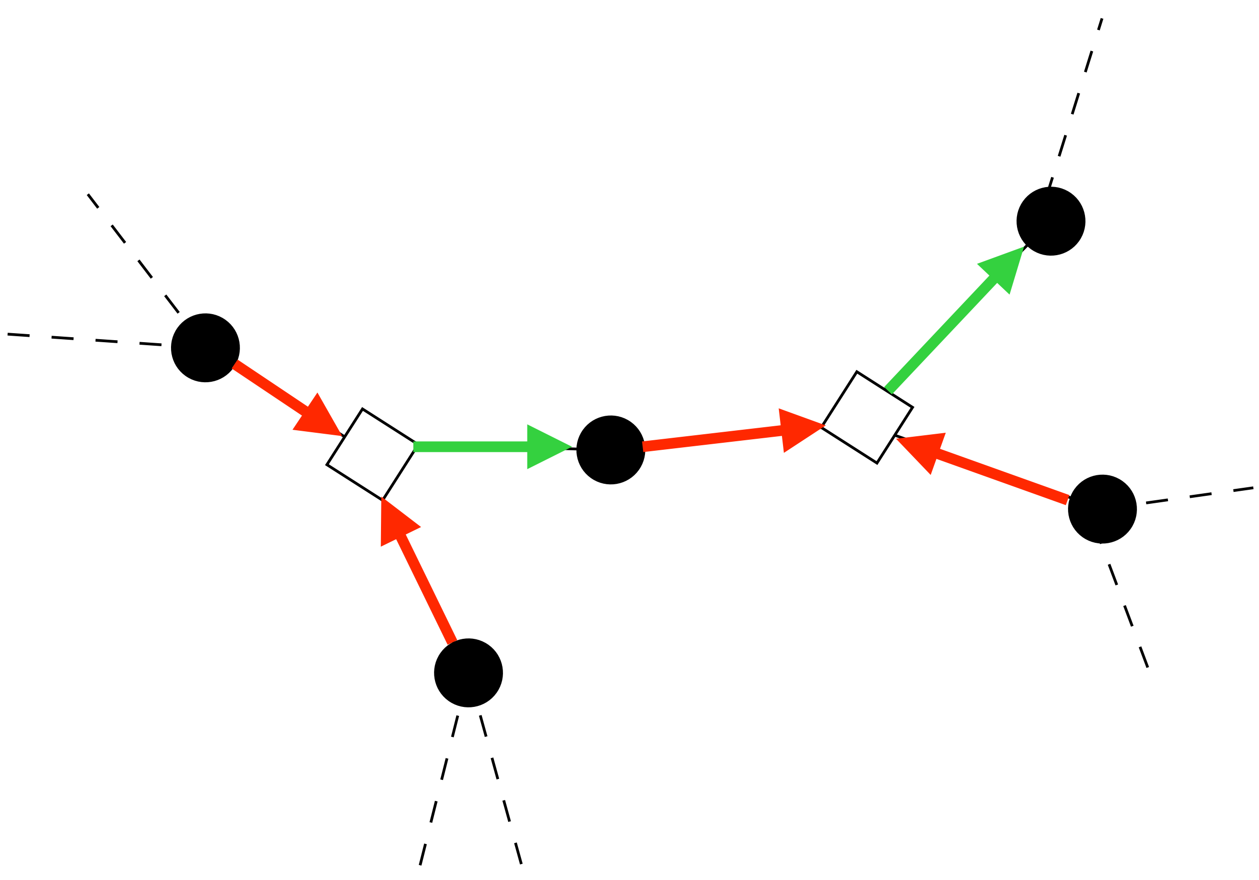
Unit Clause Propagation (UCP) Warning Propagation (WP)

- Looking for solutions
  - ➡ all constraints must be satified
  - ➡ variables may be forced to take a **unique** value (frozen variables)

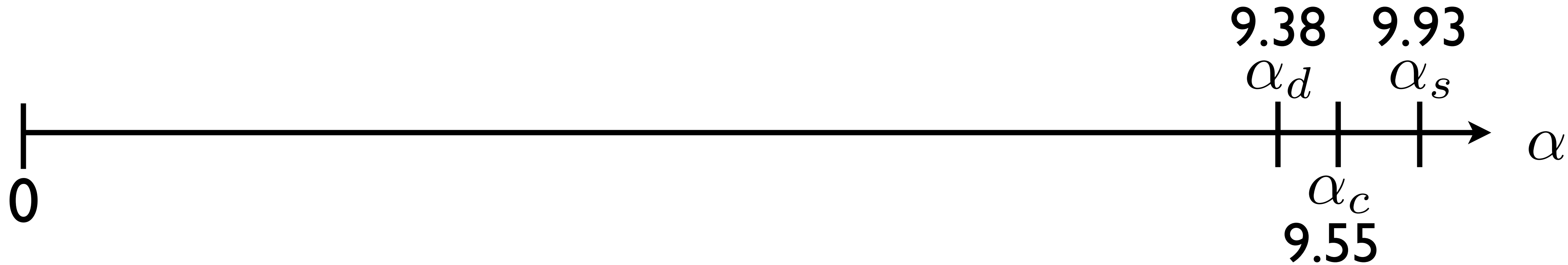


Unit Clause Propagation (UCP) Warning Propagation (WP)

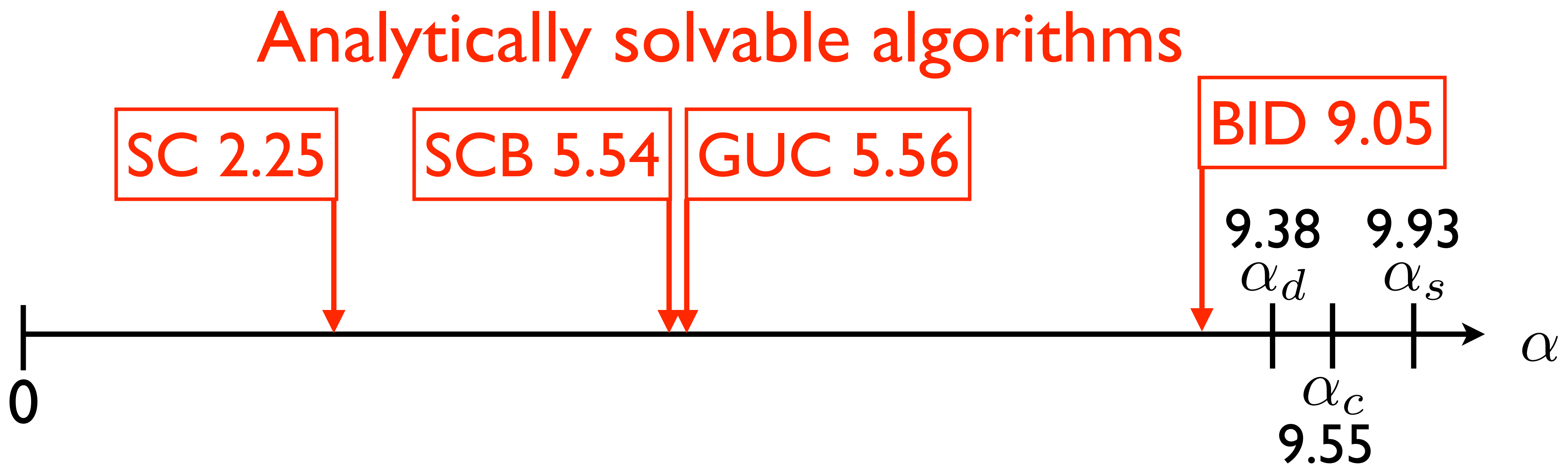
- Looking for solutions
  - ➡ all constraints must be satified
  - ➡ variables may be forced to take a **unique** value (frozen variables)



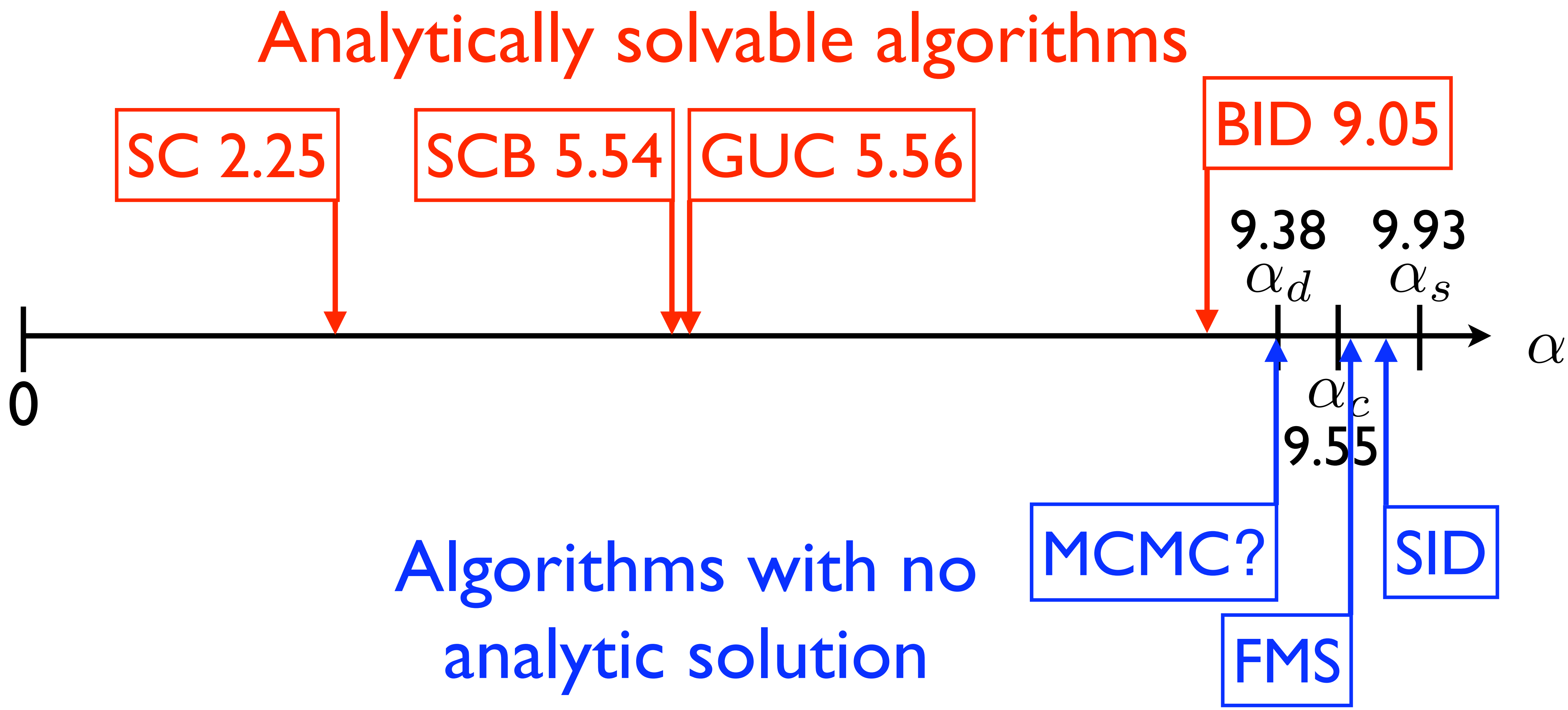
Algorithms performances



Algorithms performances



Algorithms performances



Sequential construction (BID / SID)

- `while` (there are unassigned variables)
  - compute marginals (with BP or SP)
  - choose an unassigned variable (randomly / the most biased)
  - fix it (according to its marginal / to the most probable value)
  - simplify the formula by UCP

BID mimics the perfect algorithm

- Sequential construction: $\{i(t)\}_{t=1,\dots,N}$
- Suppose to have a *perfect marginalizer*

$$\mu\left(\sigma_{i(t)} \middle| \sigma_{i(1)}, \dots, \sigma_{i(t-1)}\right)$$

- Assign variables, according to exact marginals

➡ Every run reaches a solution for sure

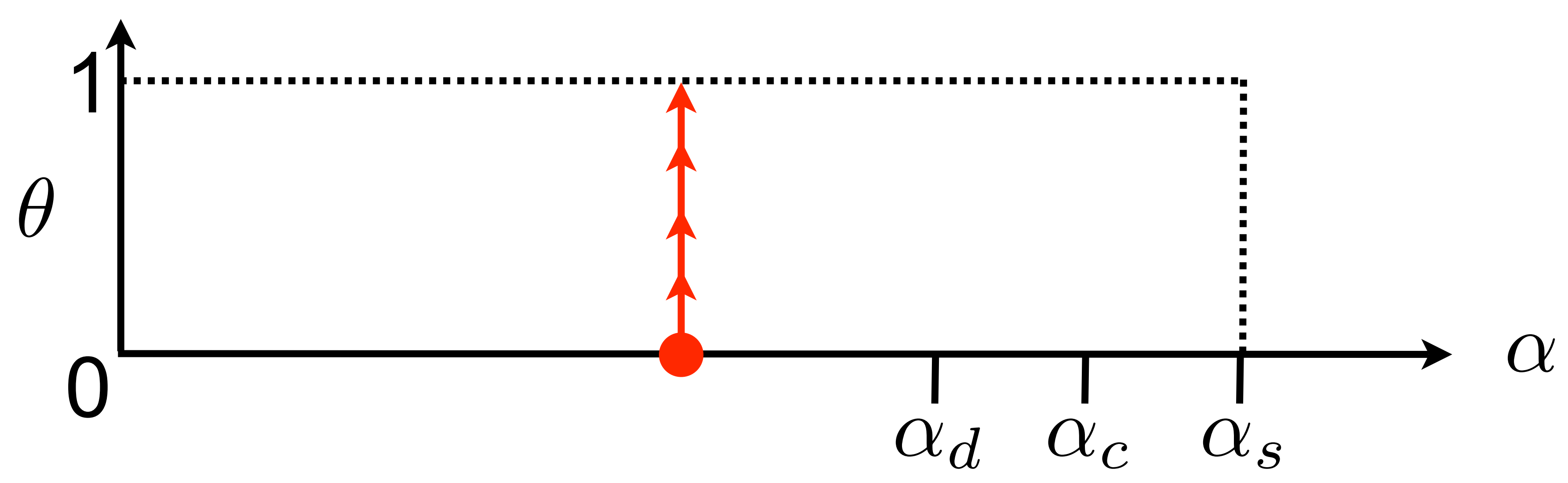
➡ Solutions are sampled uniformly

Our analysis of the algorithm

- The perfect algorithm is equivalent to:
  - Choose a solution uniformly at random $\underline{\sigma}^*$
  - Assign variables in a random order according to the chosen solution
- The typical behavior after $t = \theta N$ steps can be computed by the average $\mathbb{E}_F \; \mathbb{E}_{\underline{\sigma}^*} \; \mathbb{E}_{V_\theta}$

Our analysis of the algorithm

- The perfect algorithm is equivalent to:
  - Choose a solution uniformly at random $\underline{\sigma}^*$
  - Assign variables in a random order according to the chosen solution
- The typical behavior after $t = \theta N$ steps can be computed by the average $\mathbb{E}_F \, \mathbb{E}_{\underline{\sigma}^*} \, \mathbb{E}_{V_\theta}$



What we measure
(numerically and analytically)

- Residual entropy:

$$\omega(\theta) \equiv \frac{1}{N} \mathbb{E}_F \, \mathbb{E}_{\underline{\sigma}^*} \, \mathbb{E}_{V_\theta} \ln Z(\underline{\sigma}^*_{V_\theta})$$

$Z(\underline{\sigma}^*_{V_\theta})$ = number of solutions compatible with the solution "exposed" on V_θ

- Fraction of frozen variables:

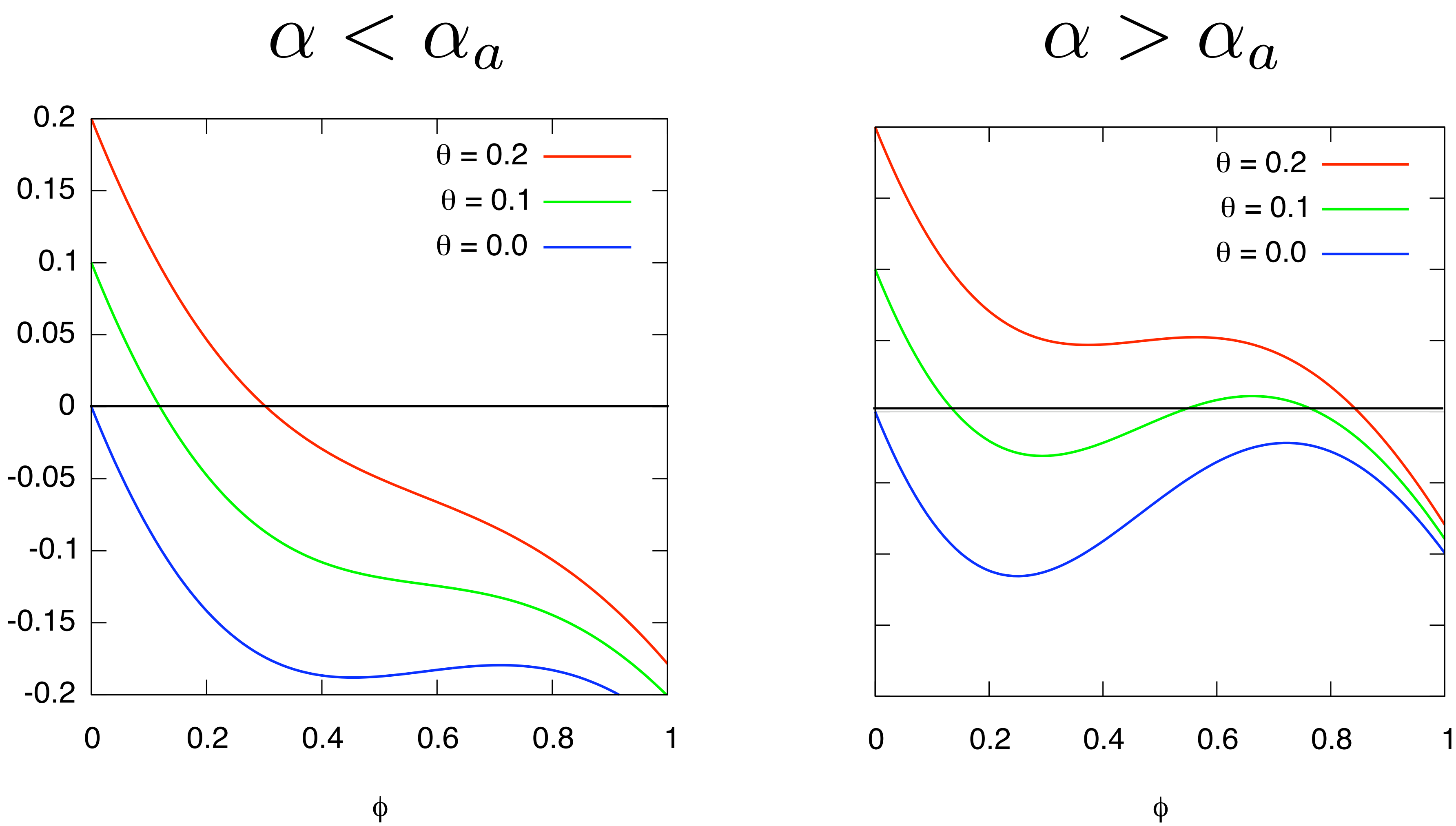
$$\phi(\theta) \equiv \frac{1}{N} \mathbb{E}_F \, \mathbb{E}_{\underline{\sigma}^*} \, \mathbb{E}_{V_\theta} |W_\theta|$$

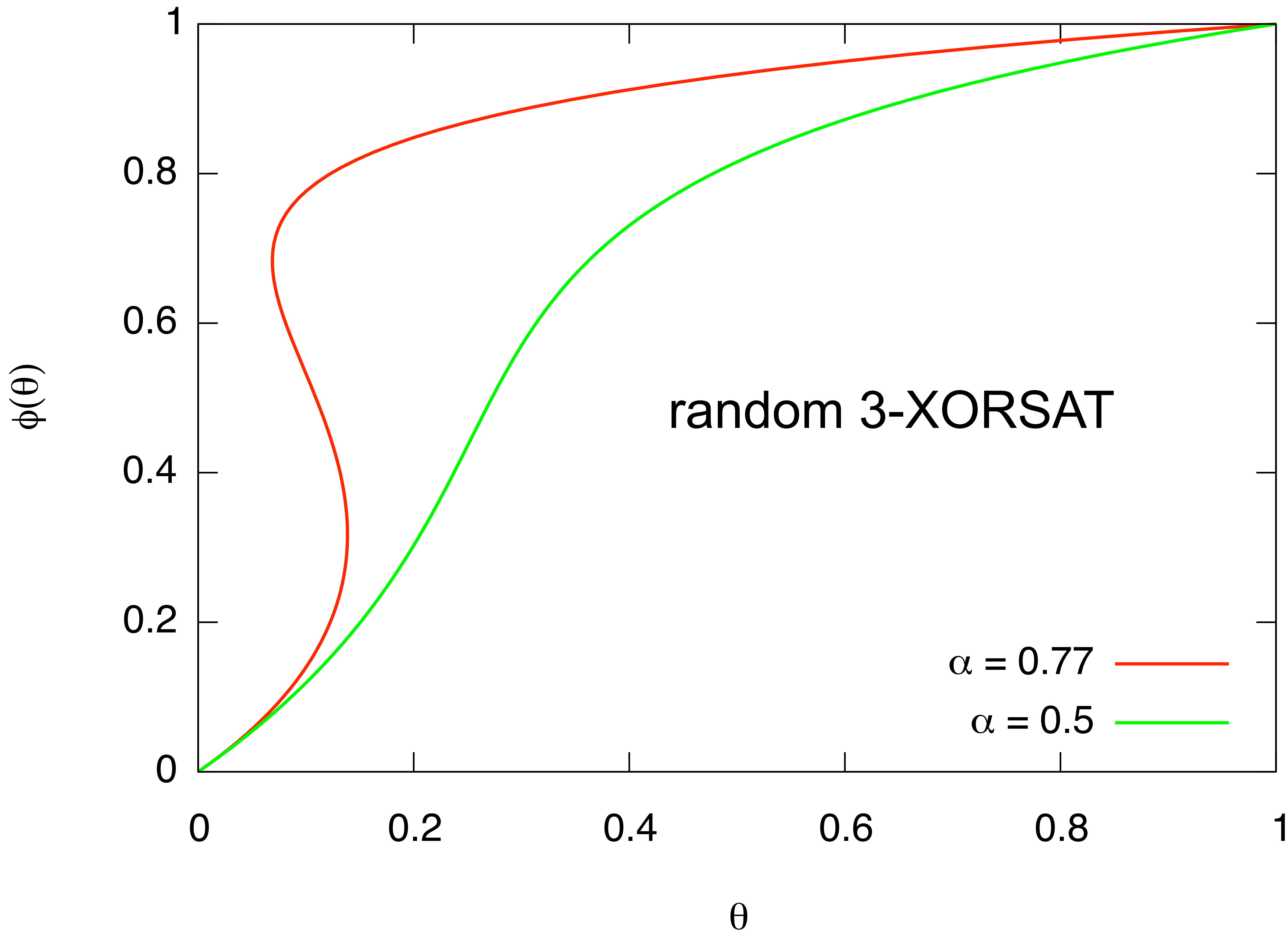
$$W_\theta = V_\theta \cup \{\text{variables implied by } V_\theta\}$$

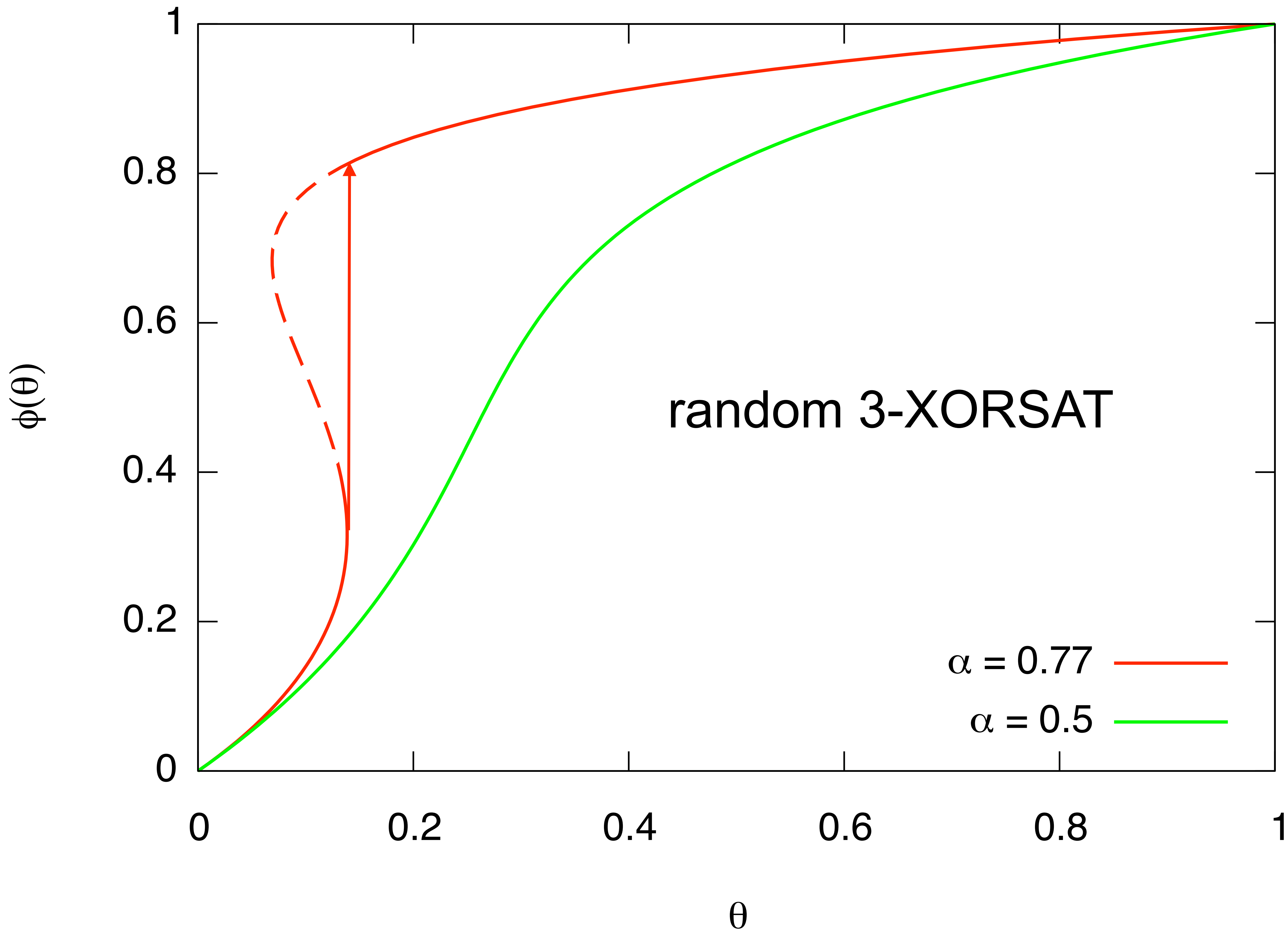
Results for random 3-XORSAT

- Full analytic solution (by diferential equations)

$$\phi = \theta + (1-\theta)\left(1 - e^{-\alpha k \phi^{k-1}}\right)$$




random 3-XORSAT
α = 0.77
α = 0.5
φ(θ)
θ
0
0.2
0.4
0.6
0.8
1



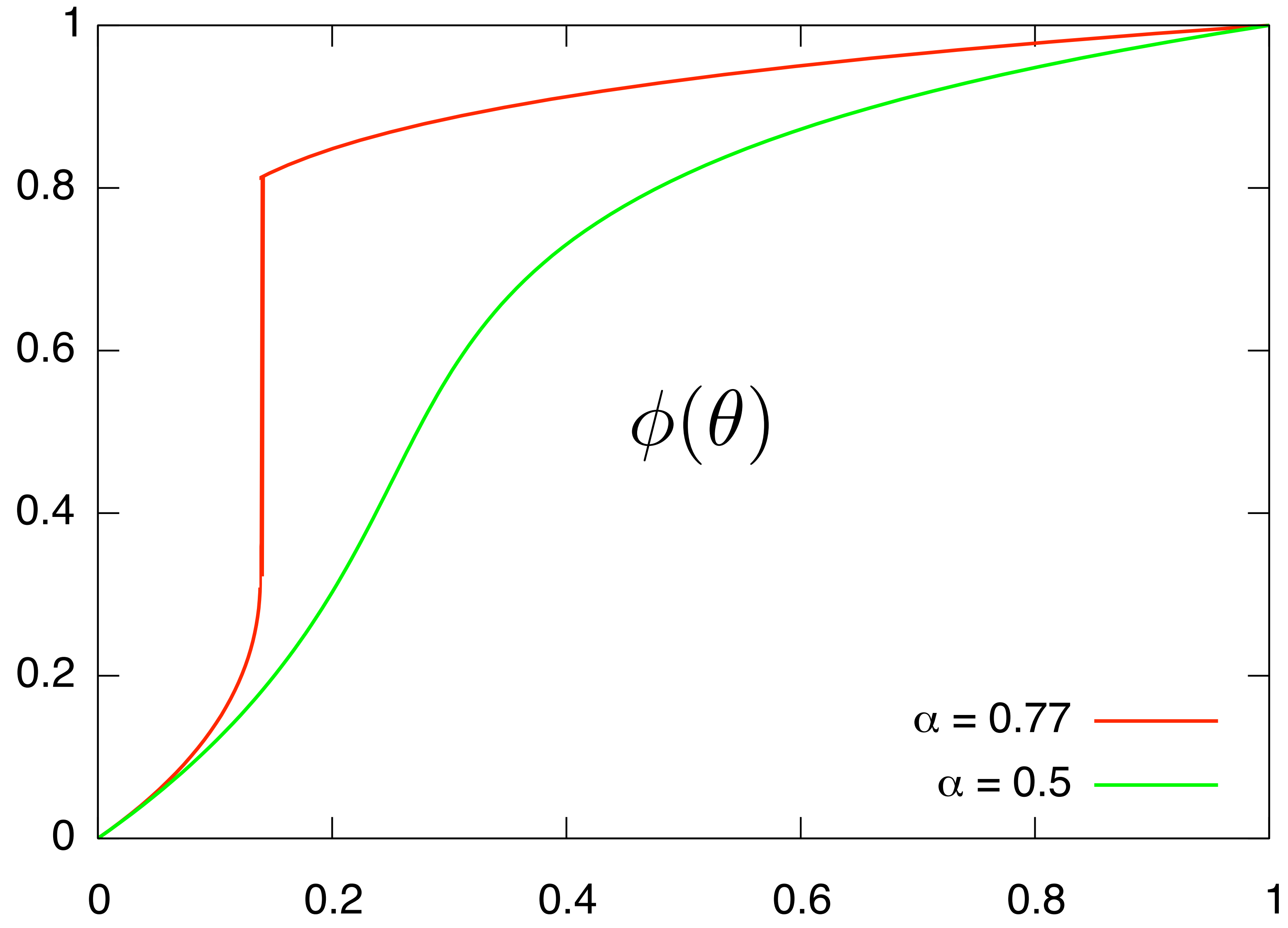
1
0.8
0.6
0.4
0.2
0
φ(θ)
random 3-XORSAT
α = 0.77
α = 0.5
0
0.2
0.4
0.6
0.8
1
θ

Results for random 3-XORSAT

Phase transition for $\alpha > \alpha_a = \frac{1}{k}\left(\frac{k-1}{k-2}\right)^{k-2}$

Jump in $\phi(\theta)$
and
cusp in $\omega(\theta)$

$$\alpha_a(k=3) = \frac{2}{3}$$

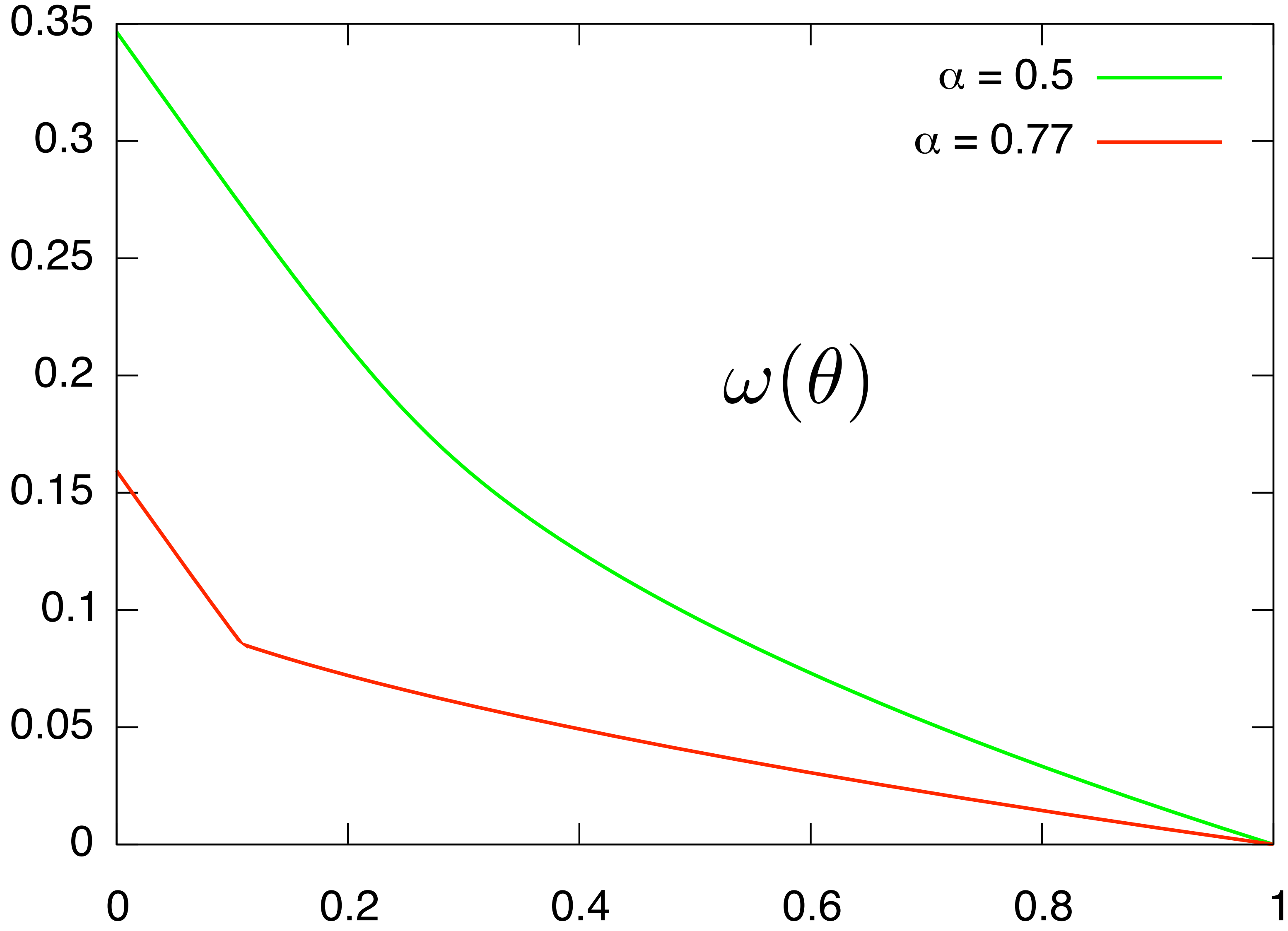


Results for random 3-XORSAT

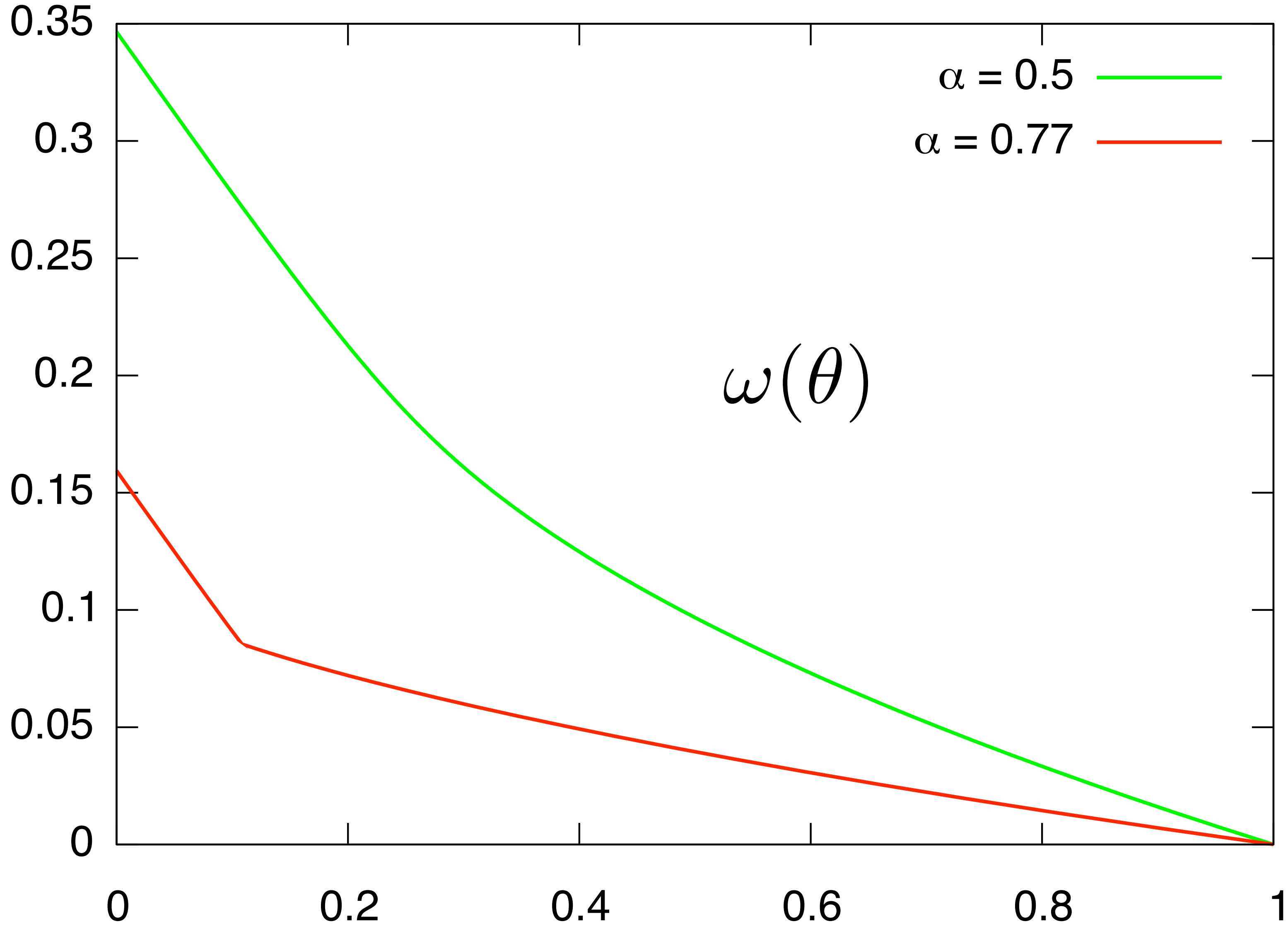
Phase transition for $\alpha > \alpha_a = \frac{1}{k}\left(\frac{k-1}{k-2}\right)^{k-2}$

Jump in $\phi(\theta)$
and
cusp in $\omega(\theta)$

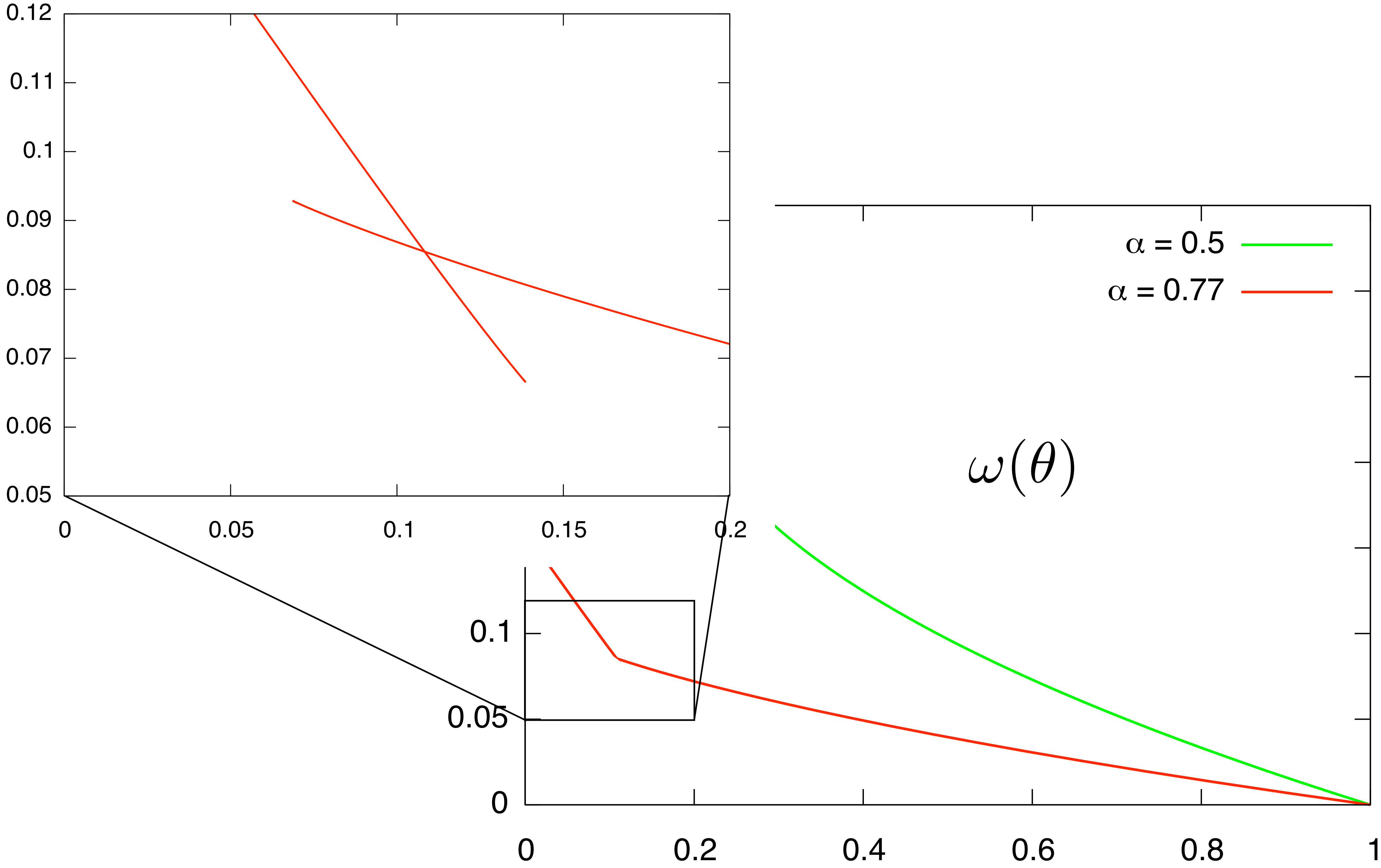
$$\alpha_a(k=3) = \frac{2}{3}$$



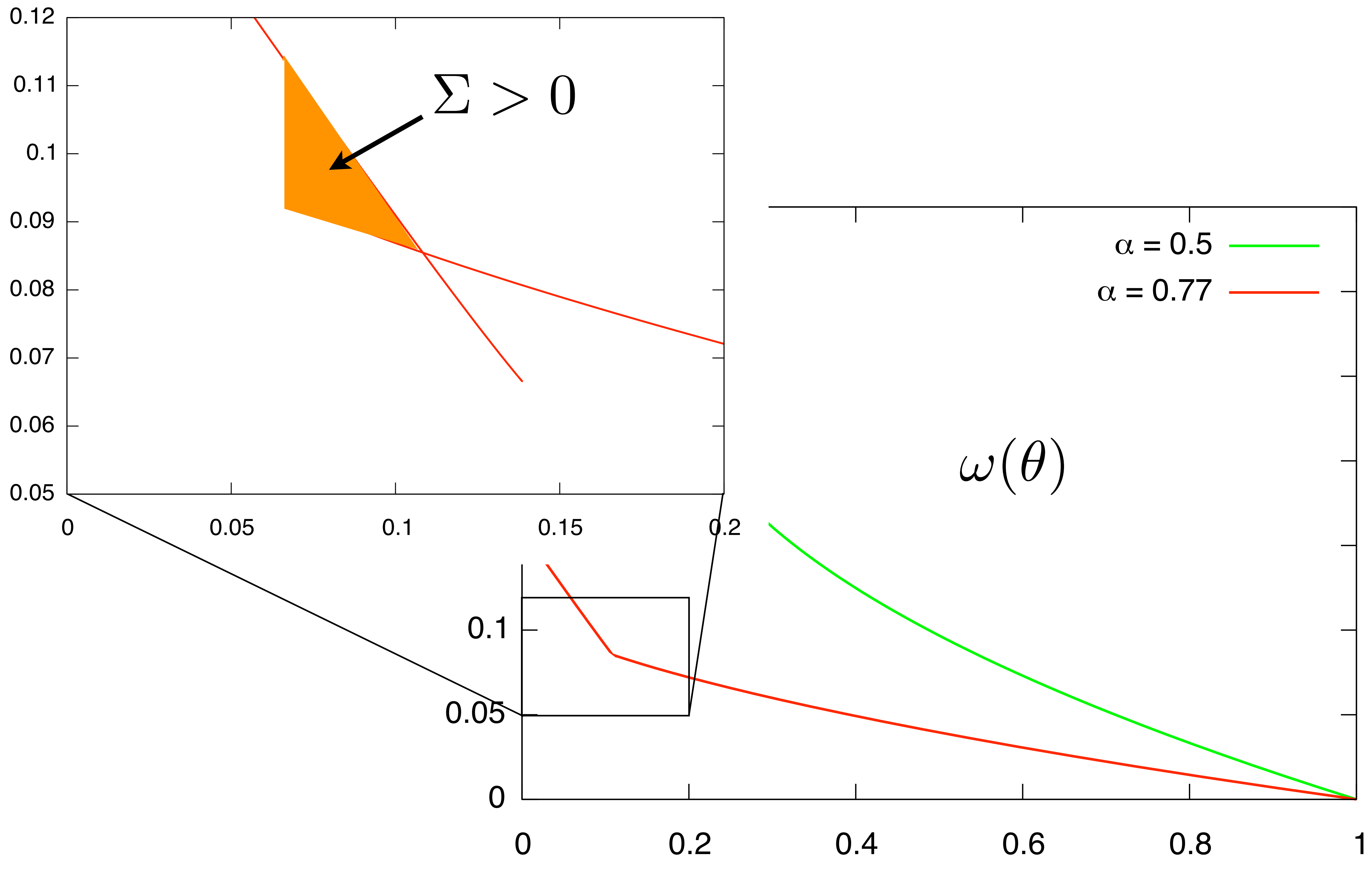
Results for random 3-XORSAT



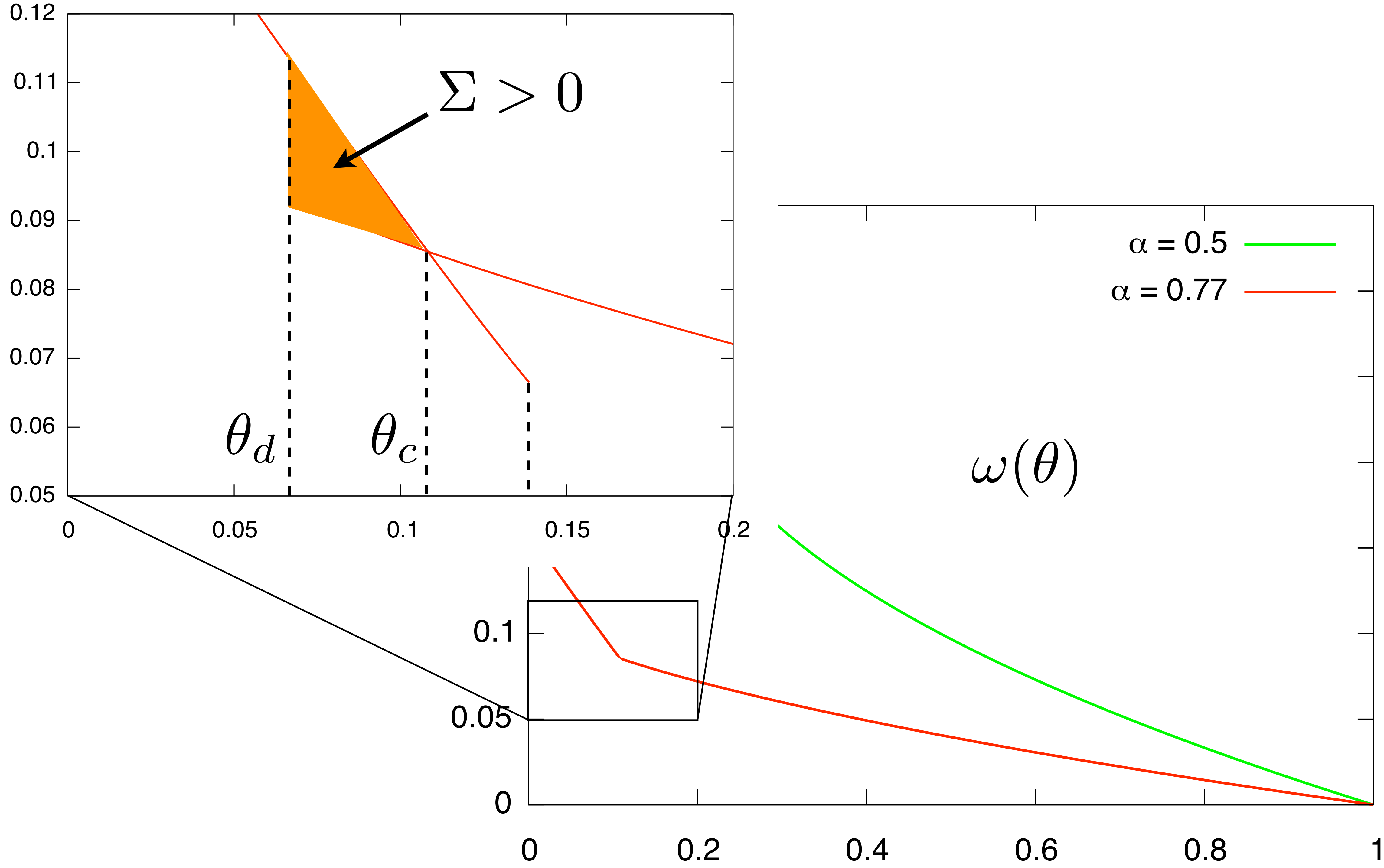
Results for random 3-XORSAT



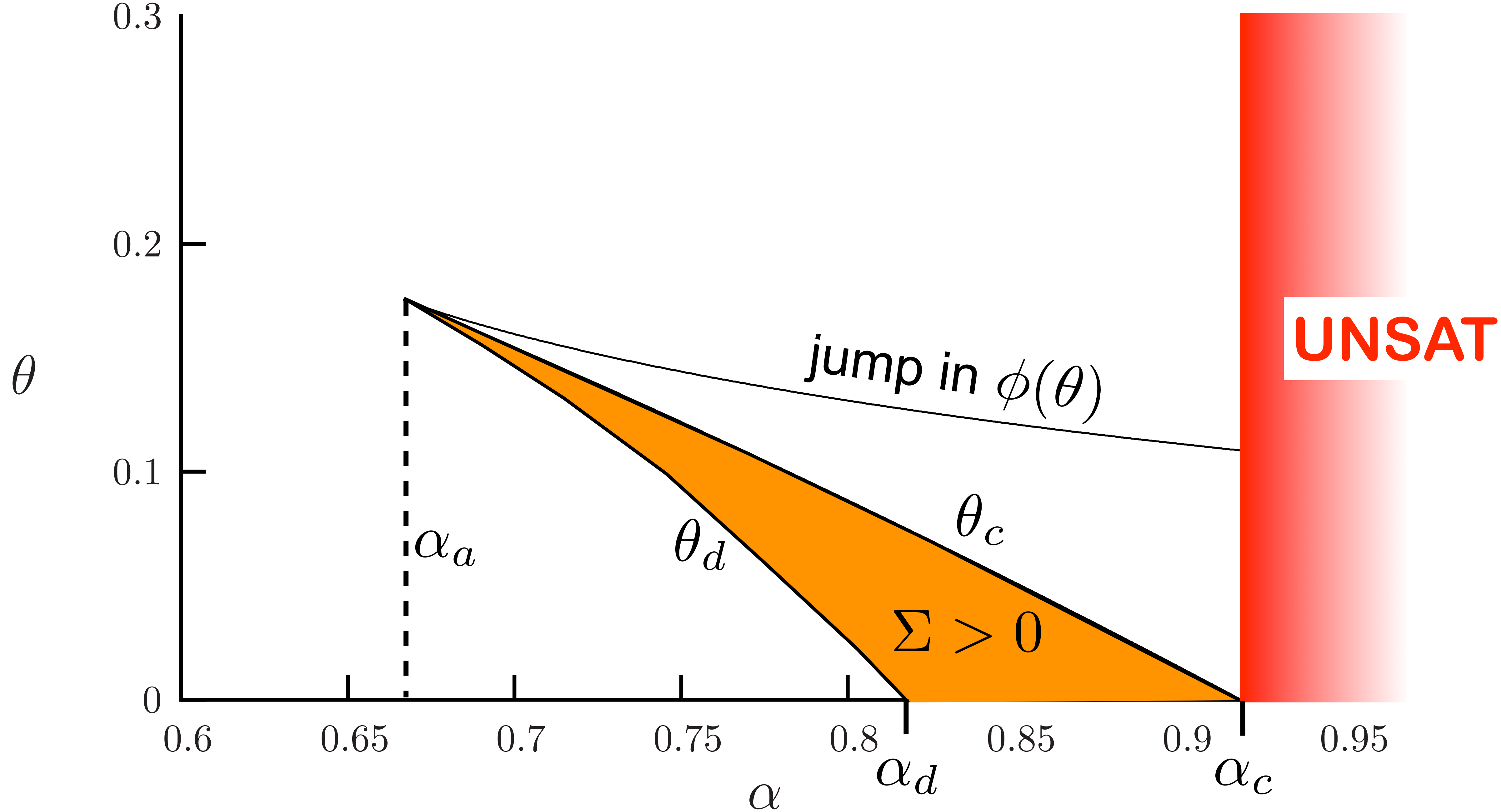
Results for random 3-XORSAT



Results for random 3-XORSAT



Phase diagram for random 3-XORSAT

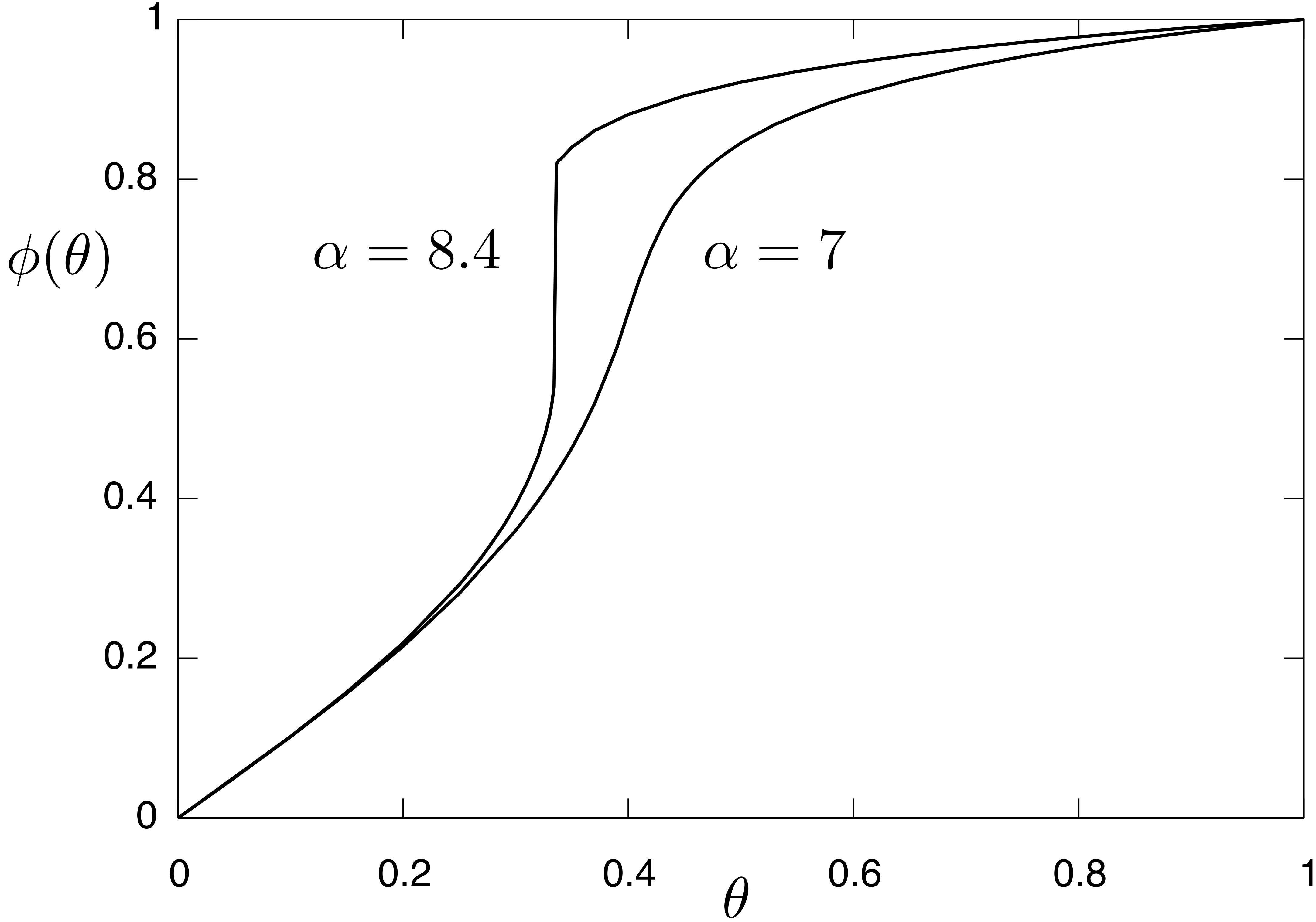


Numerics for random k-SAT

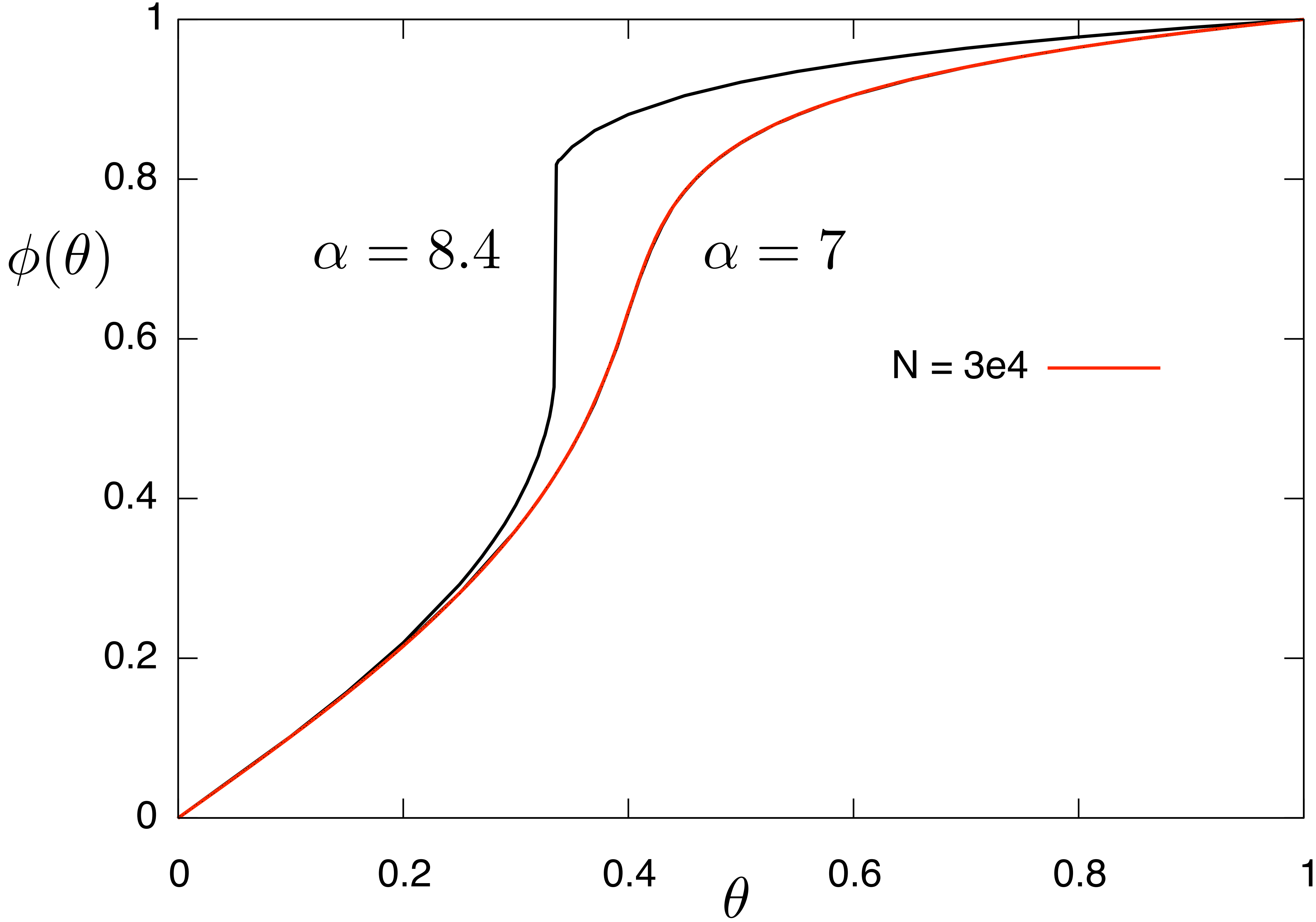
$$\boxed{\begin{aligned}\alpha_d &= 9.38\\ \alpha_c &= 9.55\\ \alpha_s &= 9.93\end{aligned}}$$

- $k = 4$, N = 1e3, 3e3, 1e4, 3e4
- Run WP
  - integer variables, no approximation
- Run BP
  - much care for dealing with quasi-frozen variables
  - slow convergence (damping and restarting trick)
  - maximum number of iterations (1000)

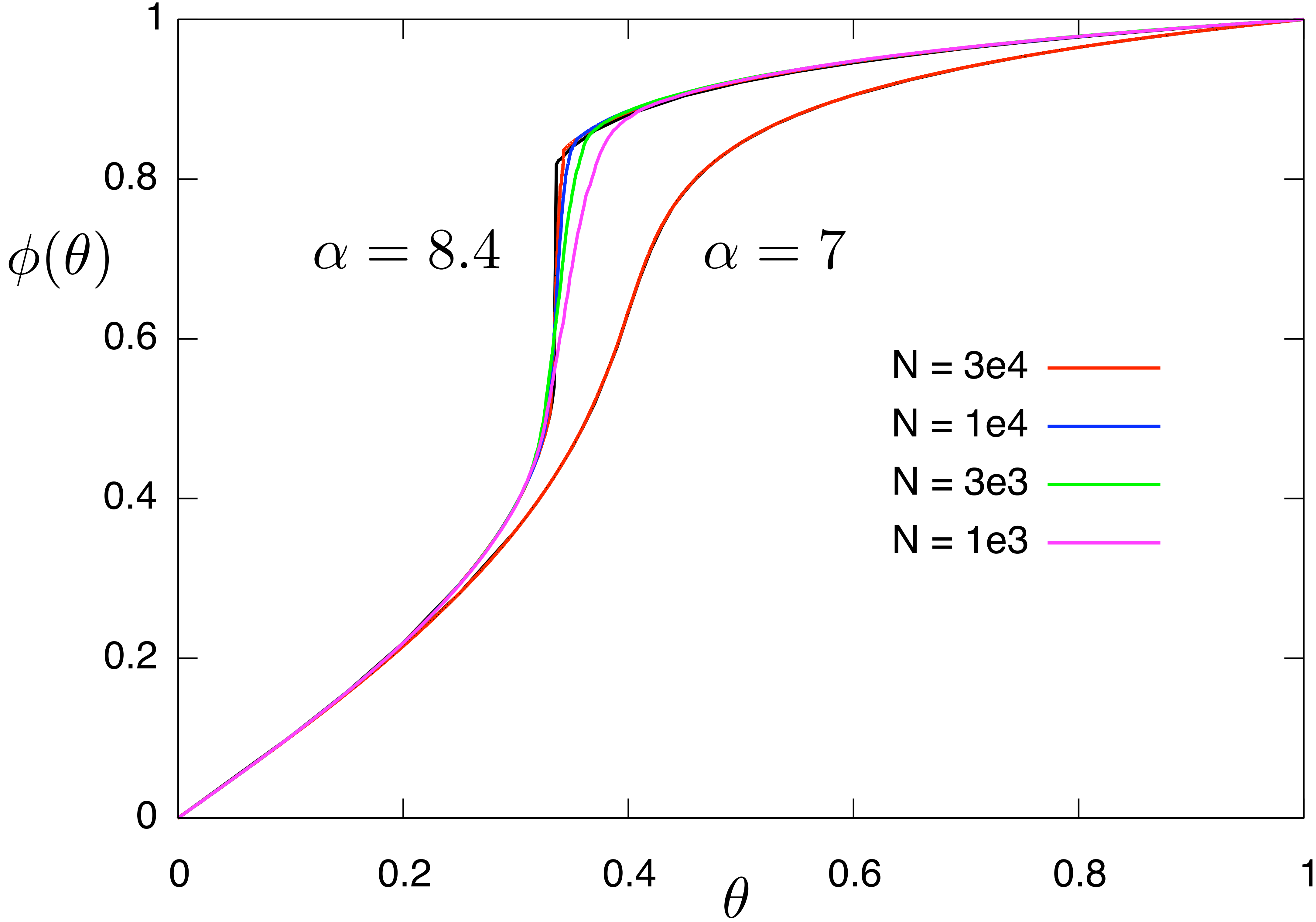
Results for random 4-SAT



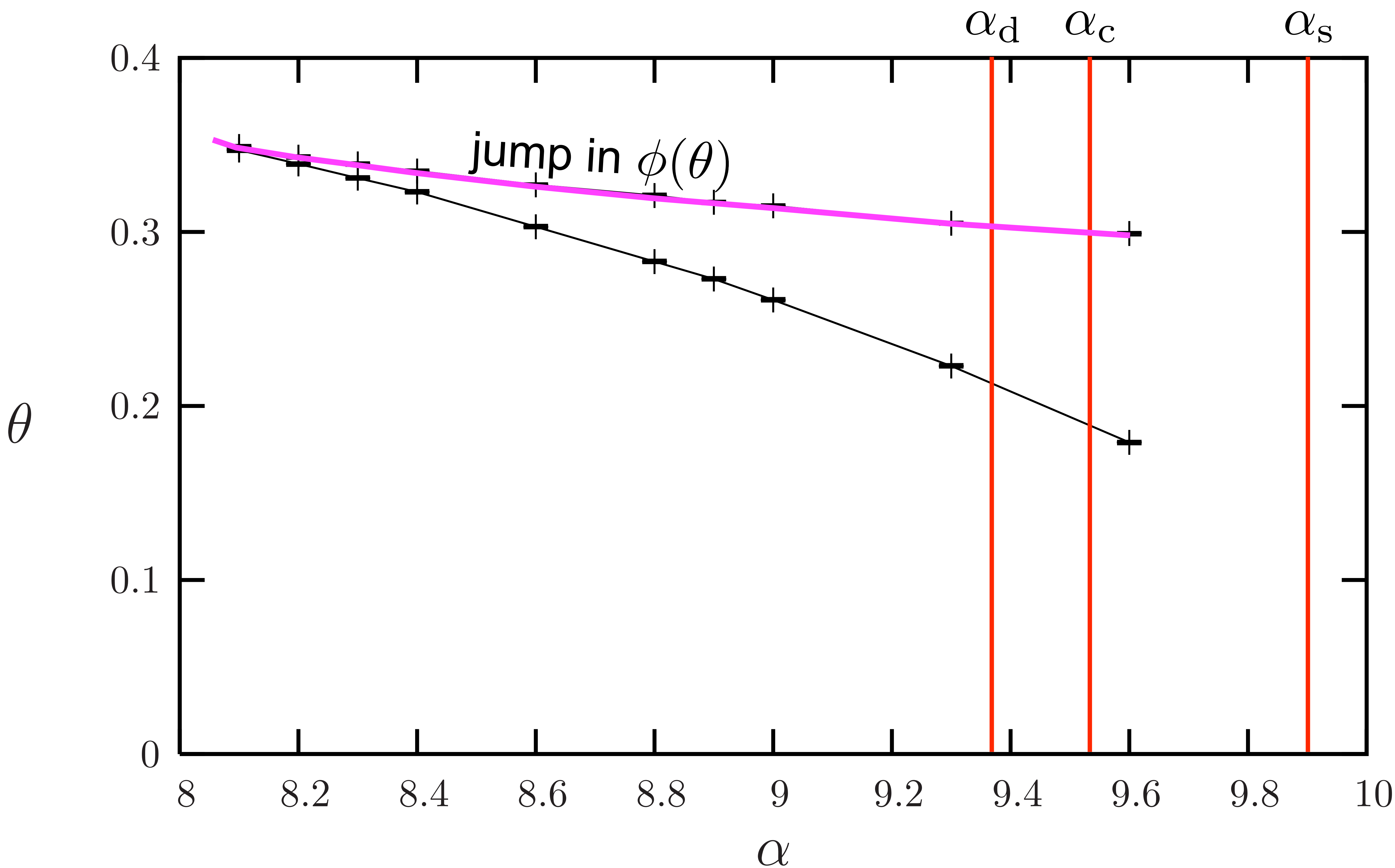
Results for random 4-SAT



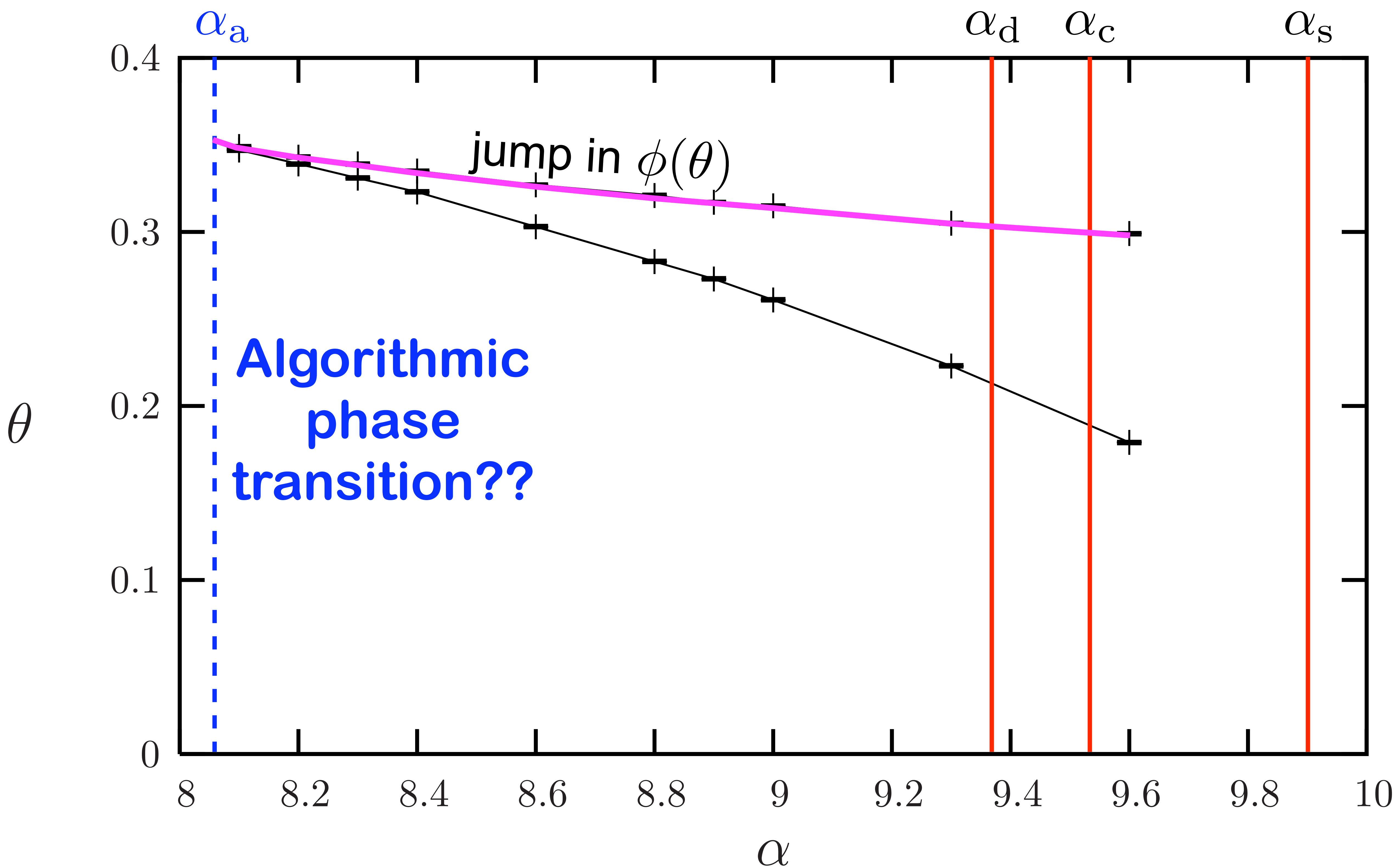
Results for random 4-SAT



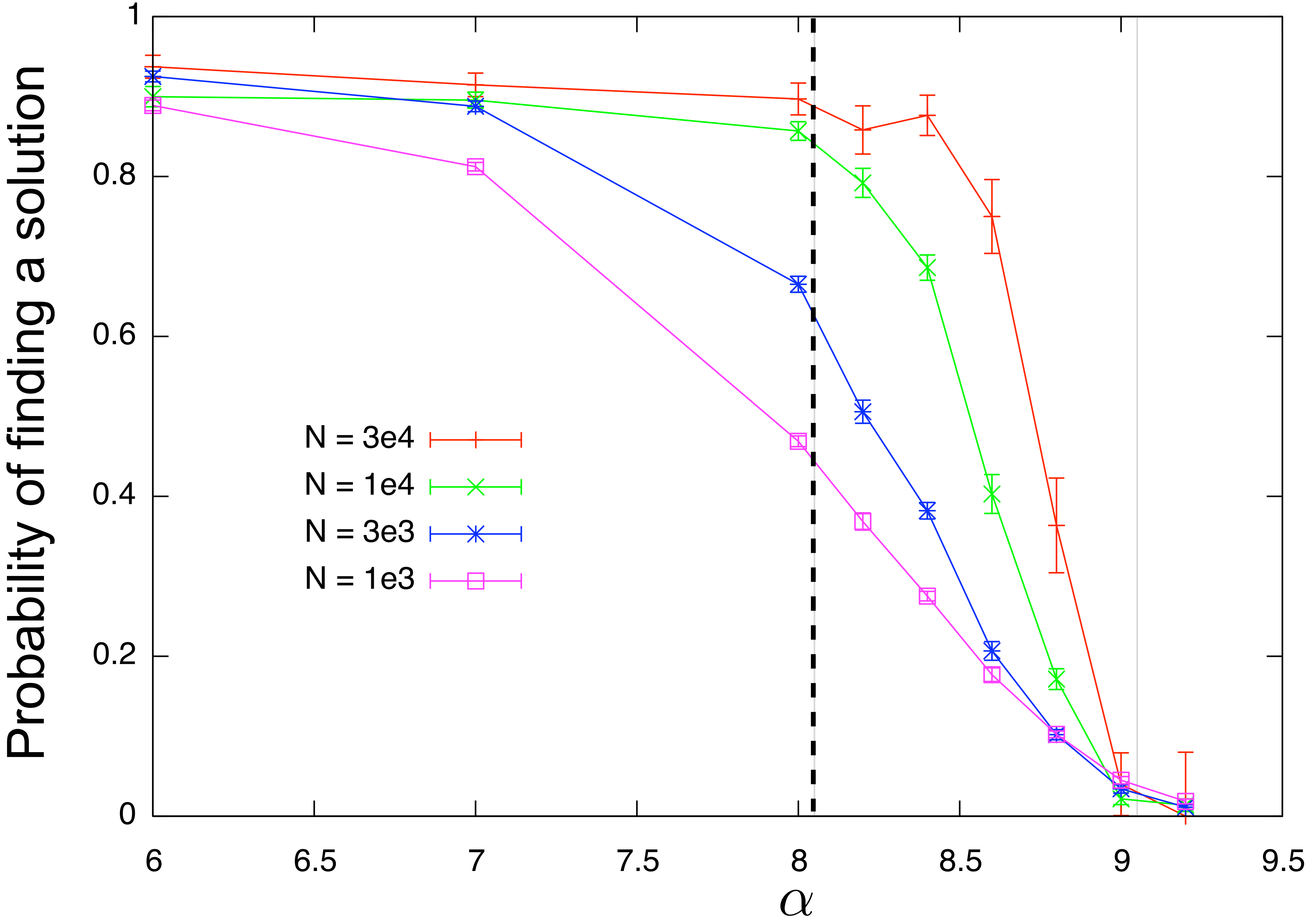
Results for random 4-SAT



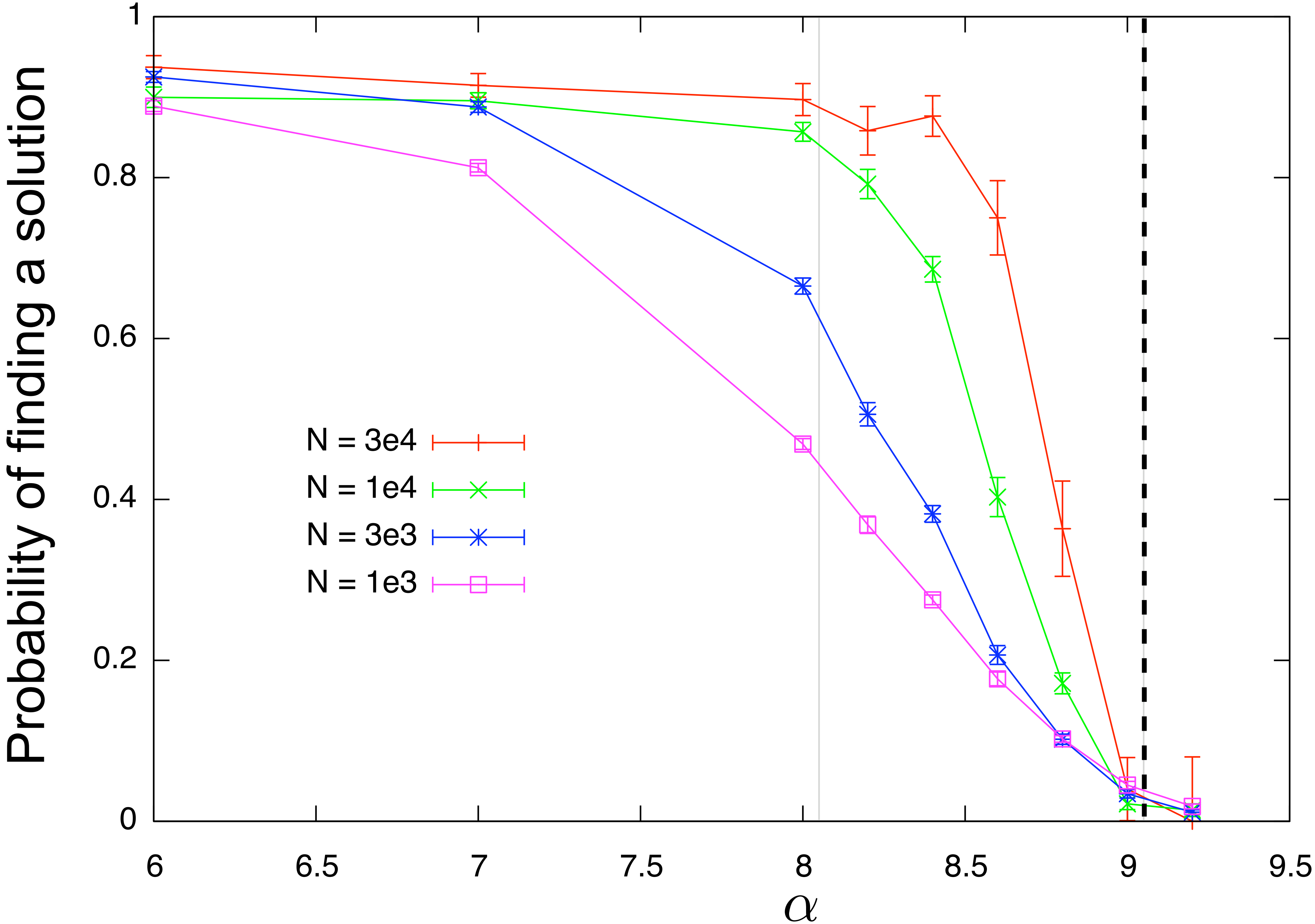
Results for random 4-SAT



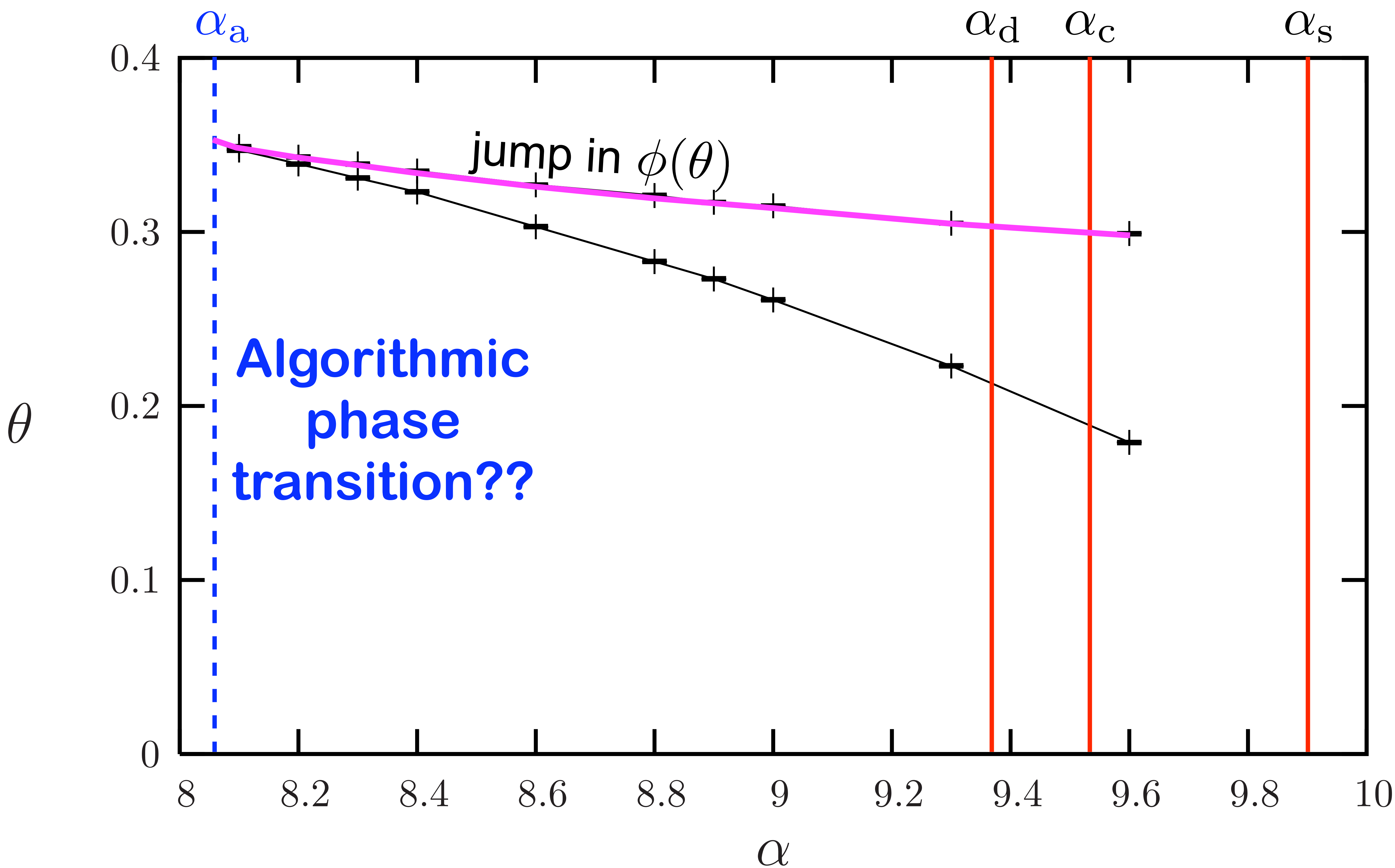
Results for random 4-SAT



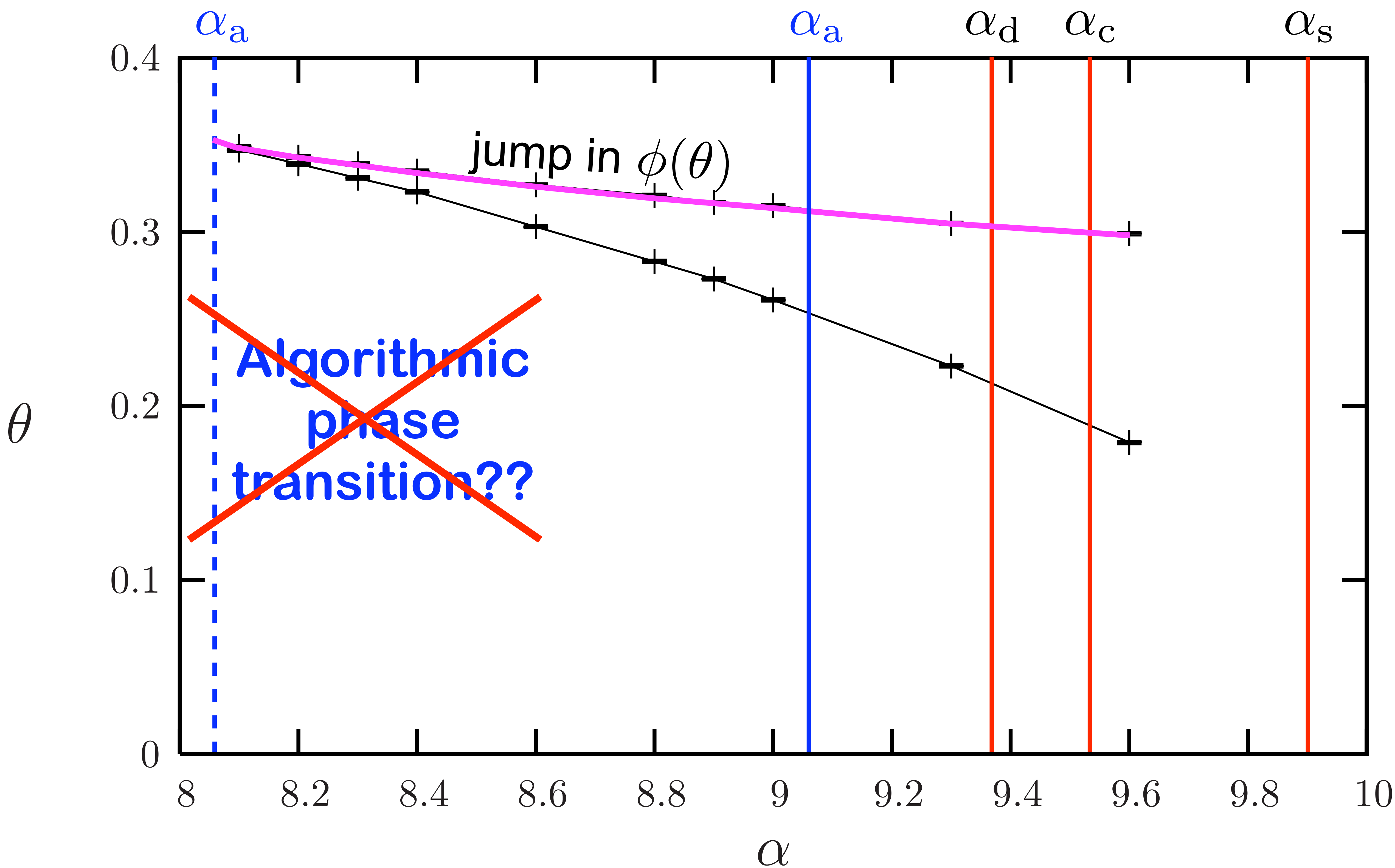
Results for random 4-SAT



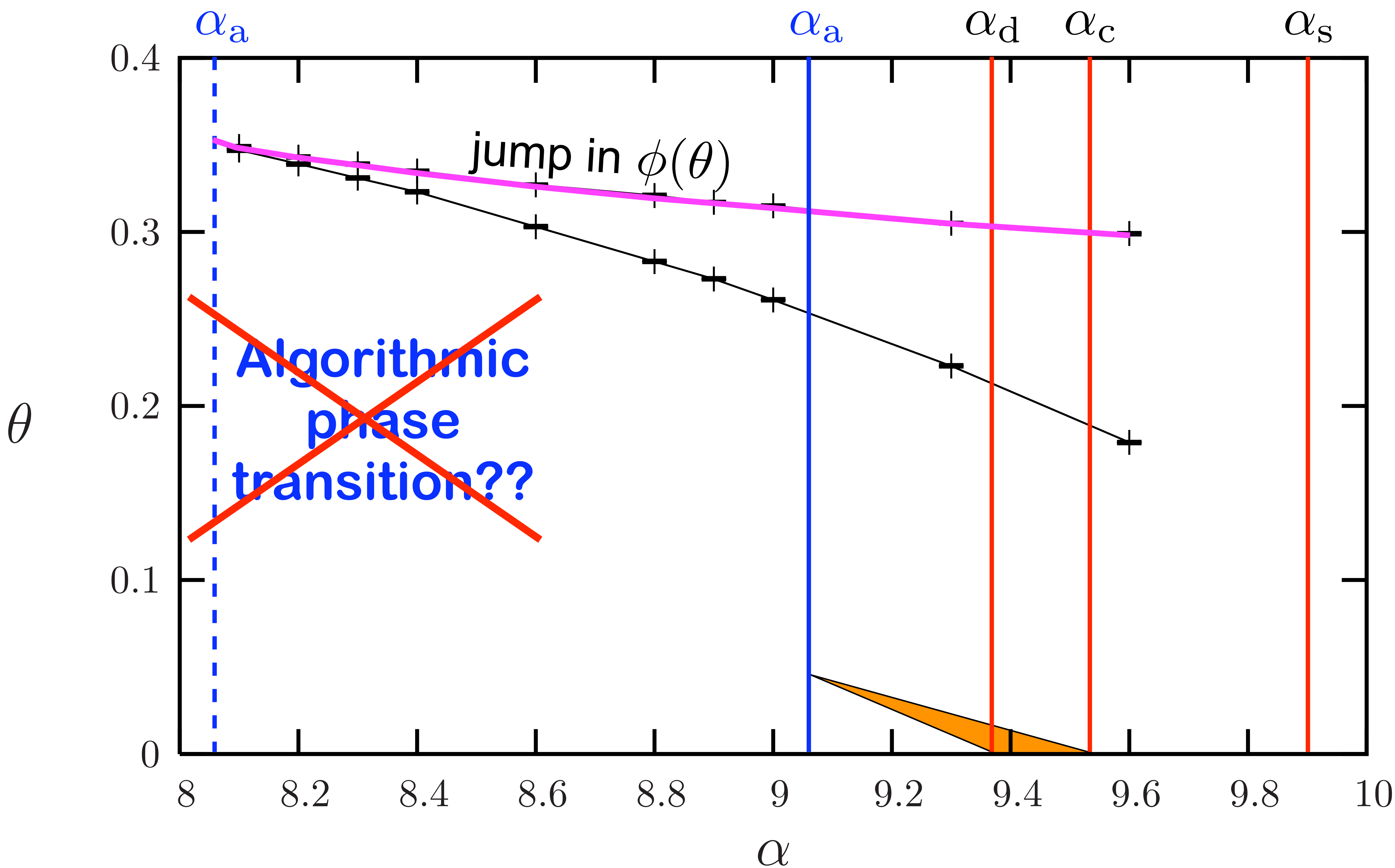
Results for random 4-SAT



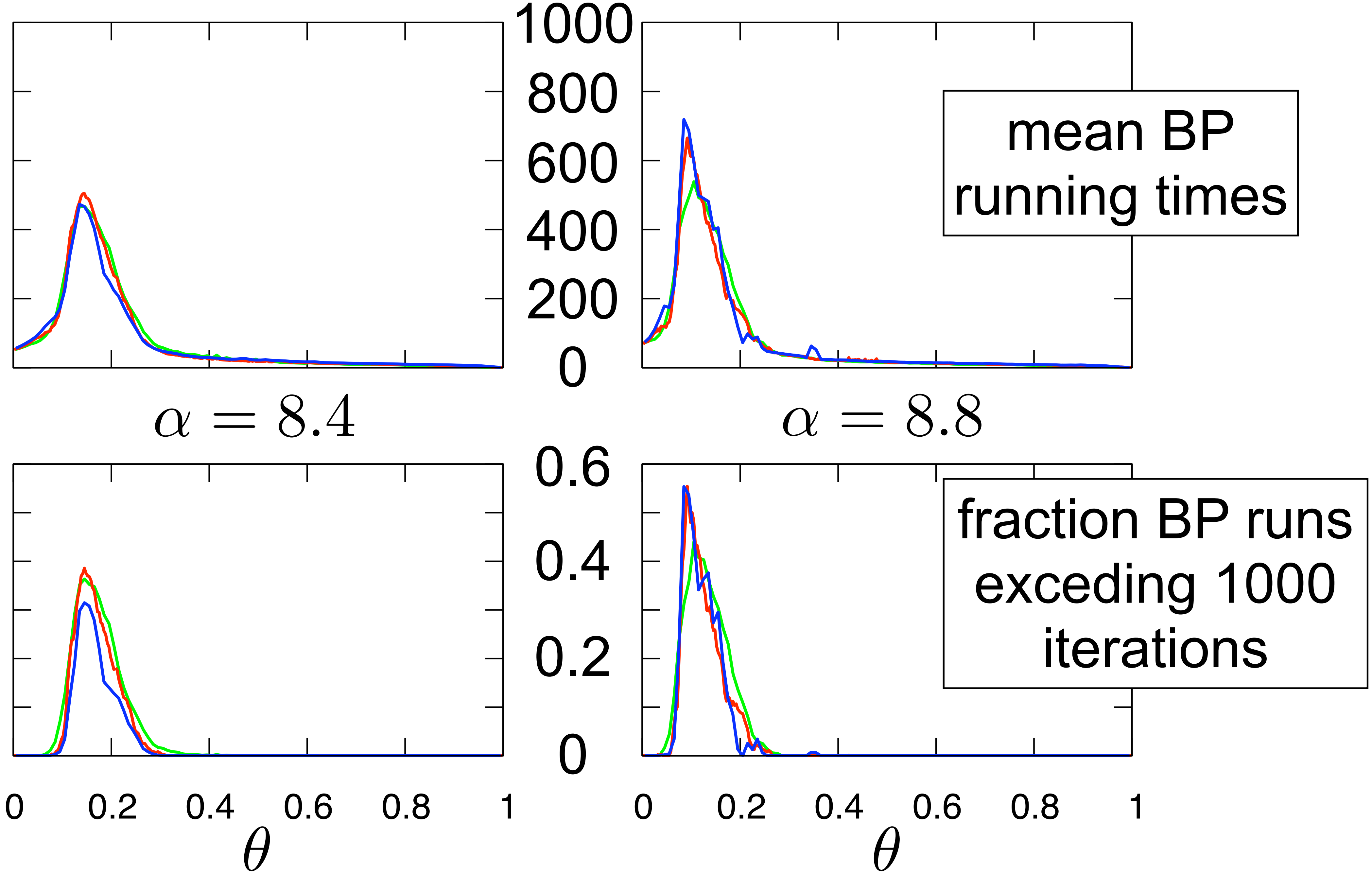
Results for random 4-SAT



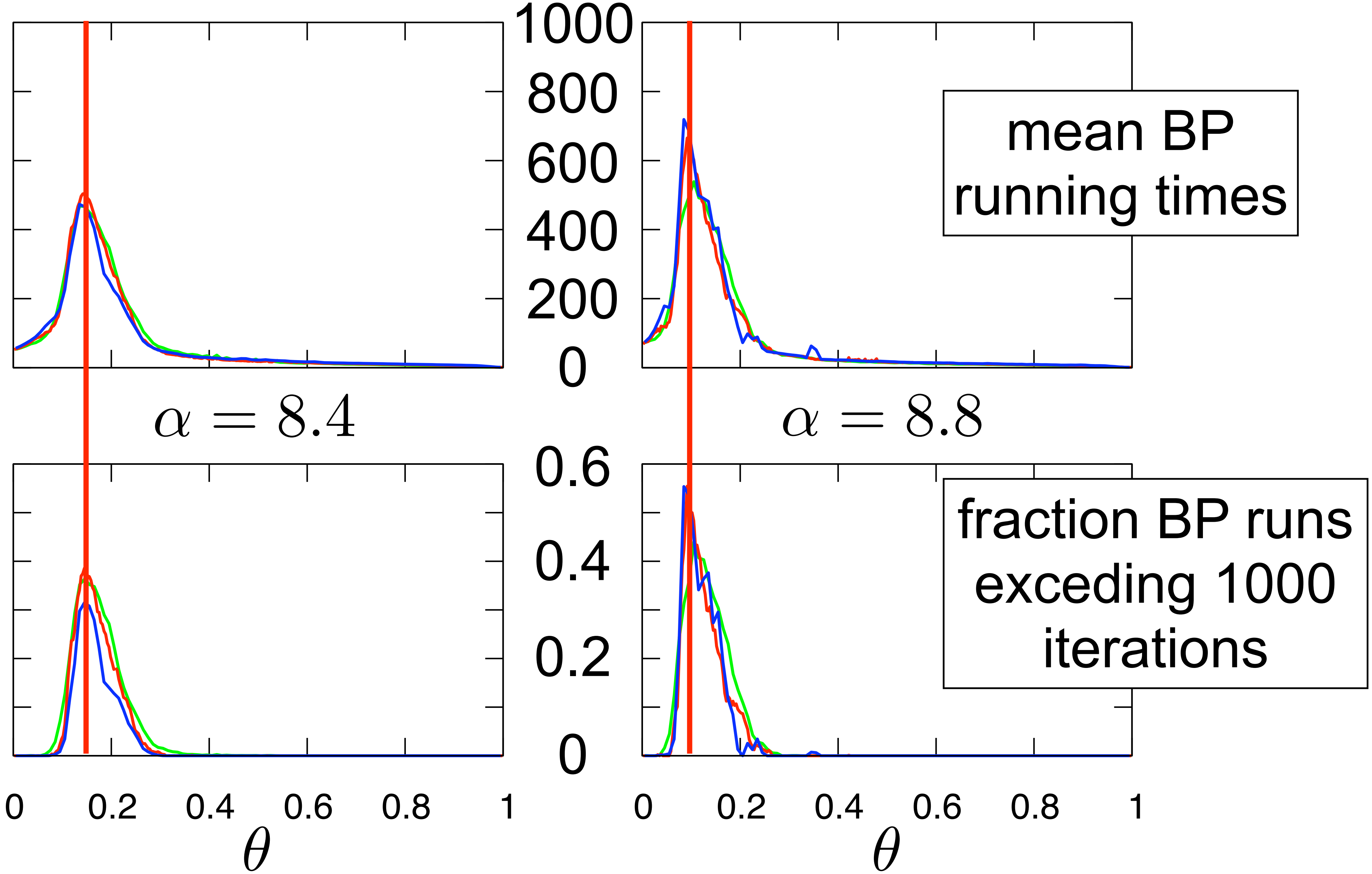
Results for random 4-SAT



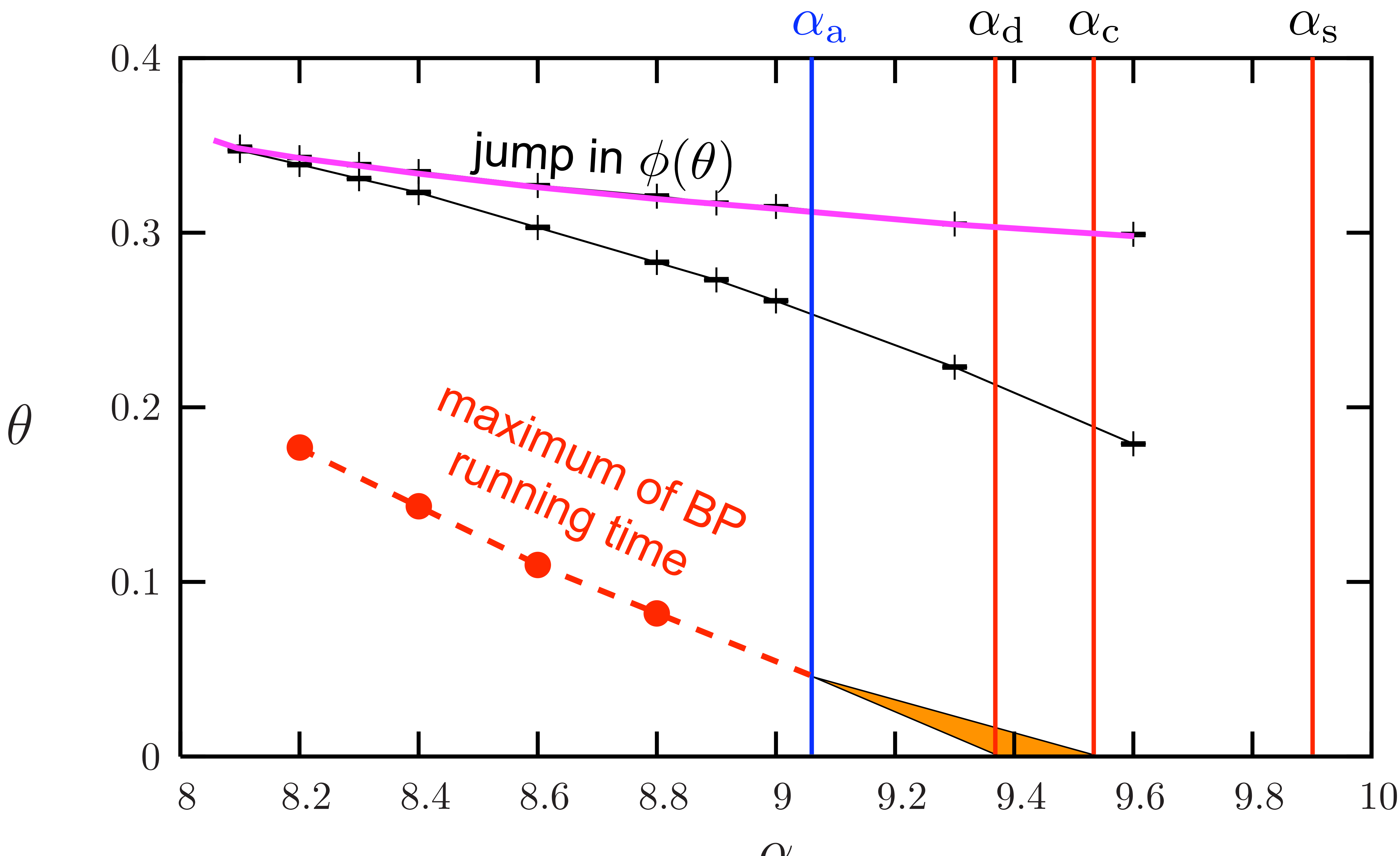
Results for random 4-SAT



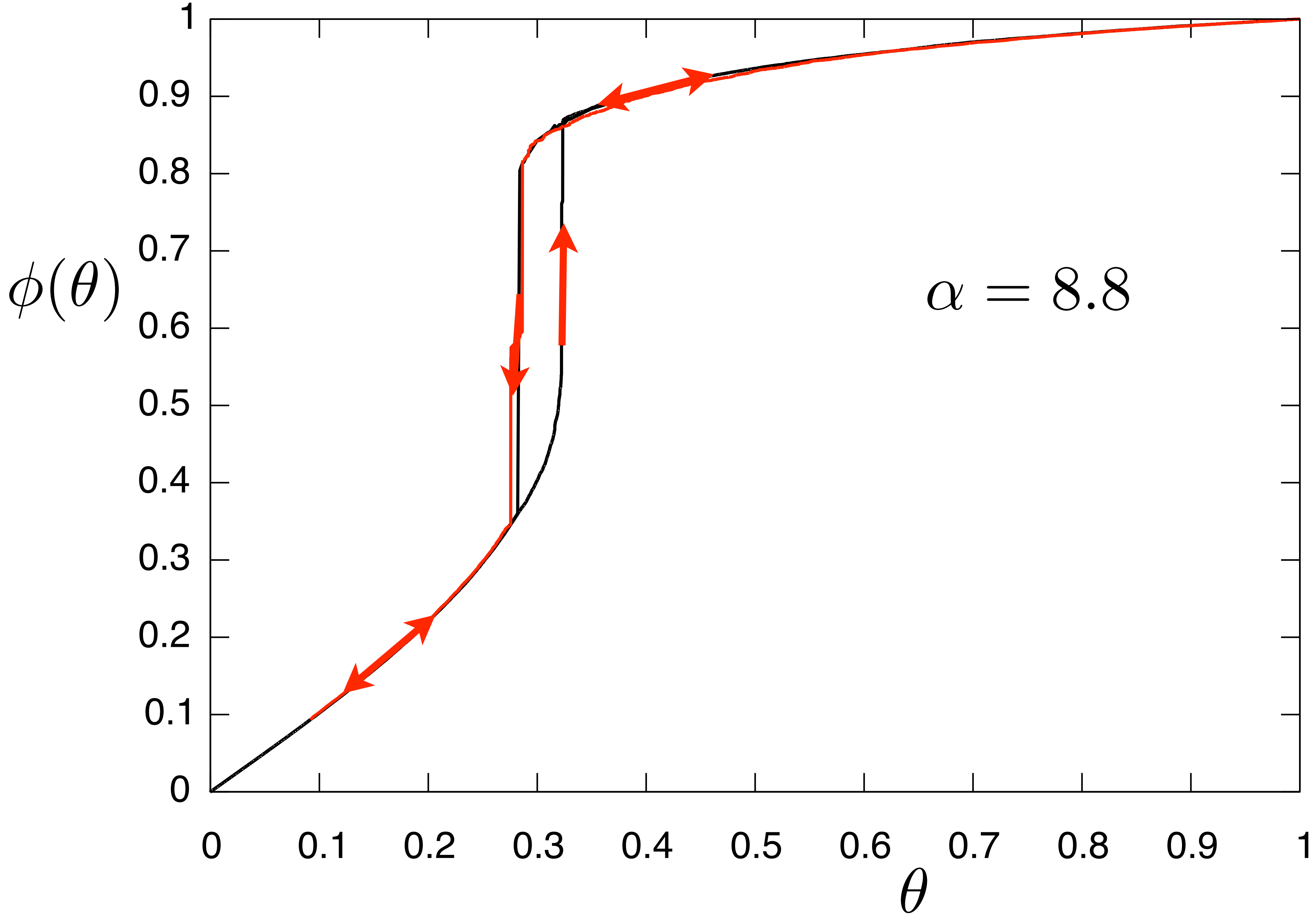
Results for random 4-SAT



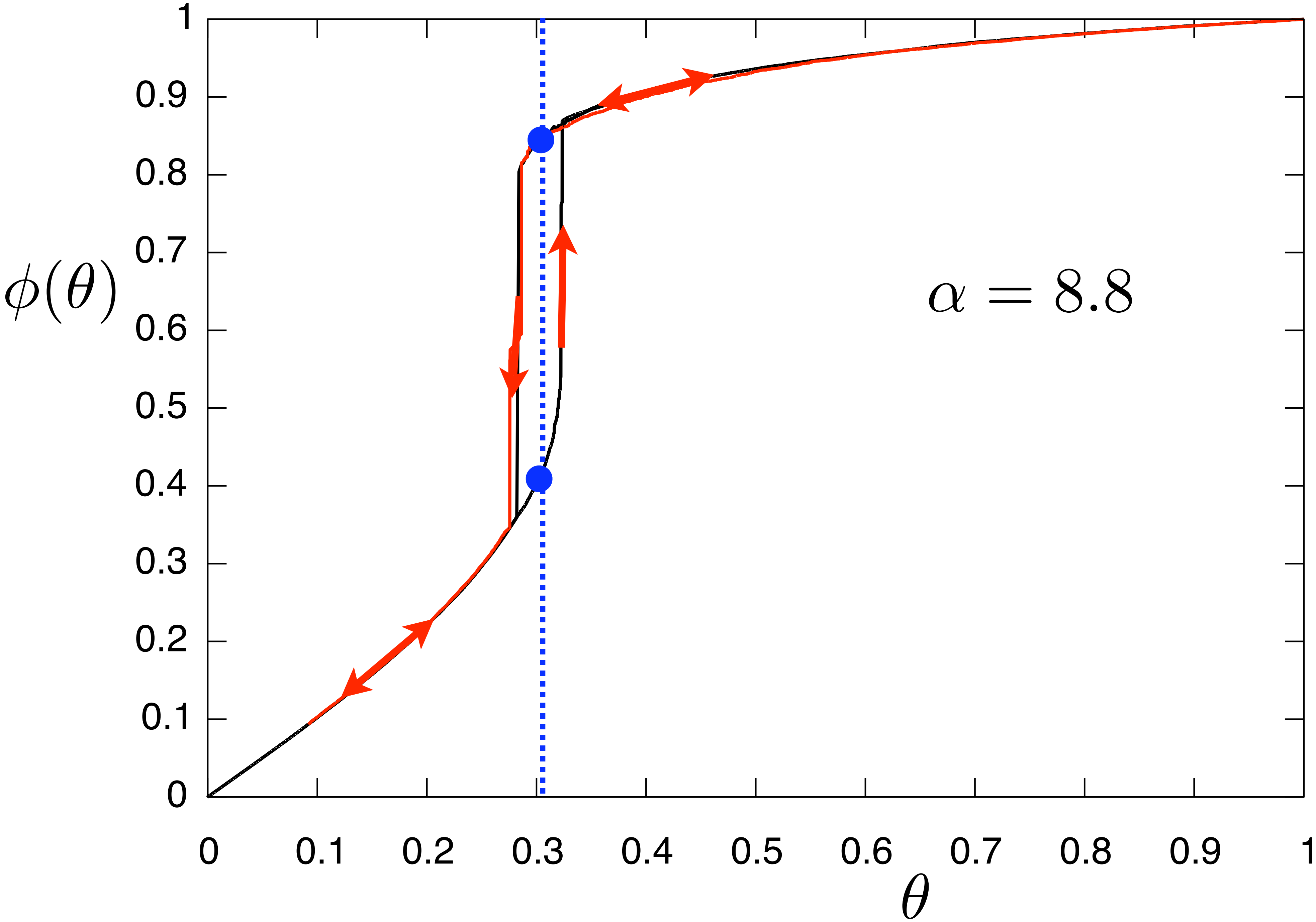
Results for random 4-SAT



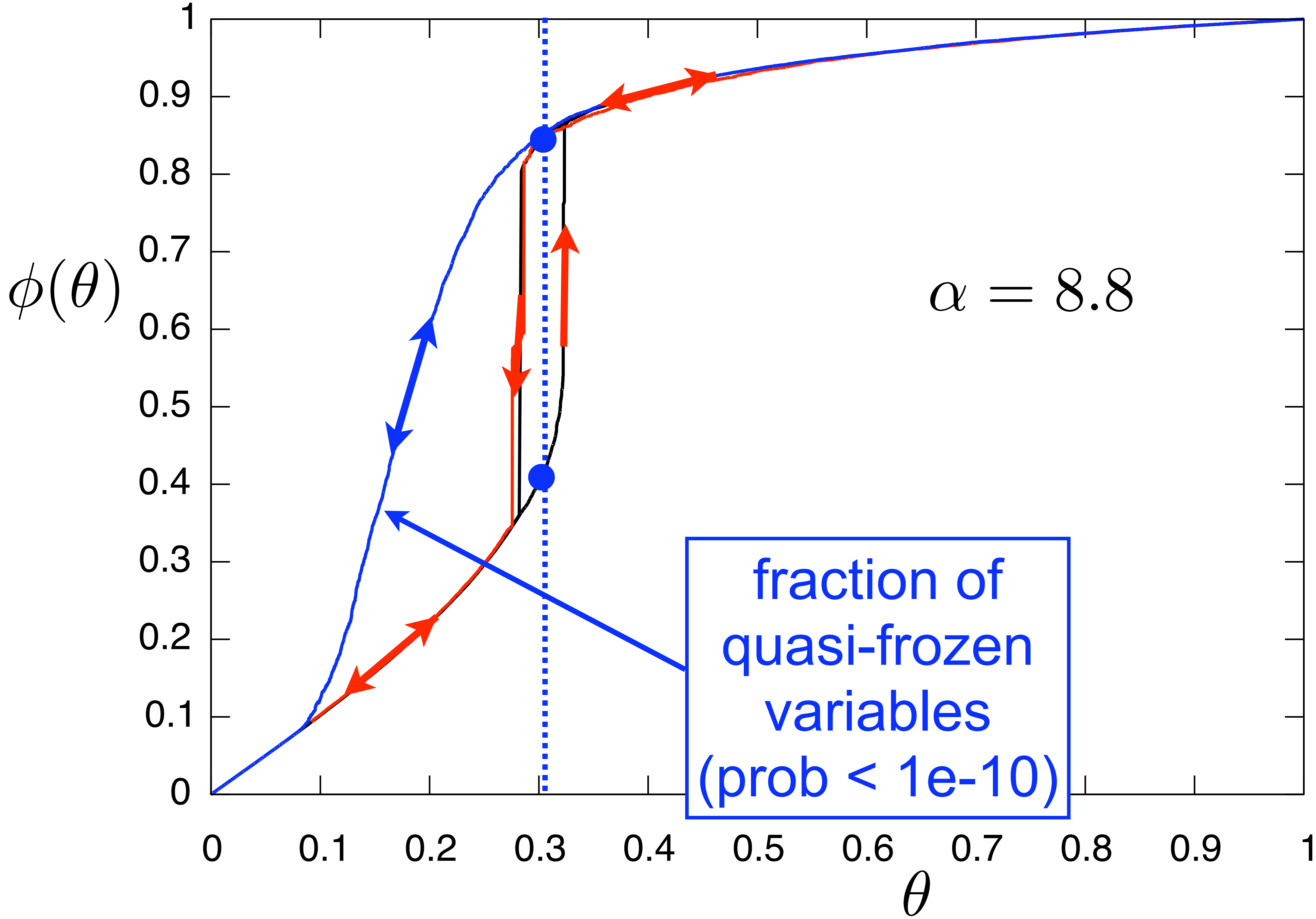
Results for random 4-SAT



Results for random 4-SAT



Results for random 4-SAT



Conclusions & open problems

- Analytically solvable algorithm for finding solutions in rCSP
- Works up to close the dynamical threshold α_d
- For large *k* we have $\alpha_d \sim \frac{\log(k)}{k} 2^k$

$$\alpha_c \sim \alpha_s \sim 2^k$$

- An algorithm working up to the condensation threshold α_c?
- Rigorous proof of BP convergence and correctness?