

Solving Constraint Satisfaction Problems through Belief Propagation-guided decimation

Andrea Montanari, Federico Ricci-Tersenghi and Guilhem Semerjian

Abstract—Message passing algorithms have proved surprisingly successful in solving hard constraint satisfaction problems on sparse random graphs. In such applications, variables are fixed sequentially to satisfy the constraints. Message passing is run after each step. Its outcome provides an heuristic to make choices at next step. This approach has been referred to as ‘decimation,’ with reference to analogous procedures in statistical physics.

The behavior of decimation procedures is poorly understood. Here we consider a simple randomized decimation algorithm based on belief propagation (BP), and analyze its behavior on random k -satisfiability formulae. In particular, we propose a tree model for its analysis and we conjecture that it provides asymptotically exact predictions in the limit of large instances. This conjecture is confirmed by numerical simulations.

I. INTRODUCTION

An instance of a constraint satisfaction problem [1] consists of n variables $\underline{x} = (x_1, \dots, x_n)$ and m constraints among them. Solving such an instance amounts to finding an assignment of the variables that satisfies all the constraints, or proving that no such assignment exists. A remarkable example in this class is provided by k -satisfiability, where variables are binary, $x_i \in \{0, 1\}$, and each constraint requires k of the variables to be different from a specific k -uple. Explicitly, the a -th constraint (clause), $a \in [m] \equiv \{1, \dots, m\}$ is specified by k variable indexes $i_1(a), \dots, i_k(a) \in [n]$, and k bits $z_1(a), \dots, z_k(a) \in \{0, 1\}$. Clause a is satisfied by assignment \underline{x} if and only if $(x_{i_1(a)}, \dots, x_{i_k(a)}) \neq (z_1(a), \dots, z_k(a))$.

A constraint satisfaction problem admits a natural factor graph [2] representation, cf. Fig. 1. Given an instance, each variable can be associated to a variable node, and each constraint to a factor node. Edges connect factor node $a \in F \equiv [m]$ to those variable nodes $i \in V \equiv [n]$ such that the a -th constraint depends in a non-trivial way on variable x_i . For instance, in the case of k -satisfiability, clause a is connected to variables $i_1(a), \dots, i_k(a)$. If the resulting graph is sparse, fast message passing algorithms can be defined on it.

Although constraint satisfaction problems are generally NP-hard, a large effort has been devoted to the development of efficient heuristics. Recently, considerable progress has been achieved in building efficient ‘incomplete solvers’ [3]. These are algorithms that look for a solution but, if they do not find one, cannot prove that the problem is unsolvable. A

particularly interesting class is provided by *message passing-guided decimation* procedures. These consist in iterating the following steps:

- 1) Run a message passing algorithm.
- 2) Use the result to choose a variable index $i \in V$, and a value x_i^* for the corresponding variable.
- 3) Replace the constraint satisfaction problem with the one obtained by fixing x_i to x_i^* .

The iteration may stop for two reasons. In the first case a contradiction is produced: the same variable x_i appears in two constraints whose other arguments have already been fixed, and that are satisfied by distinct values of x_i . If this does not happen, the iteration stops only when all the variables are fixed and a solution is found. Notice that earlier algorithms, such as unit clause propagation (UCP) [4], [5] did not use message passing in step 2, and were not nearly as effective.

Random constraint satisfaction problems are a useful testing ground for new heuristics. For instance, *random k -satisfiability* is the distribution over k -SAT formulae defined by picking a formula uniformly at random among all the ones including m clauses over n variables. Decimation procedures of the type sketched above proved particularly successful in this context. In particular *survey propagation-guided decimation* [6], [7] outperformed the best previous heuristics based on stochastic local search [3]. More recently *belief propagation-guided decimation* was shown empirically to have good performances as well [8].

Unfortunately, so far there exists no analysis of message-passing guided decimation. Our understanding almost entirely relies on simulations, even for random instances. Consequently the comparison among different heuristics, as well as the underpinnings of their effectiveness are somewhat unclear. In this paper we define a simple class of randomized message passing-guided decimation algorithms, and present a technique for analyzing them on random instances. The technique is based on the identification of a process on infinite trees that describes the evolution of the decimation algorithm. The tree process is then analyzed through an appropriate generalization of density evolution [14]. Our approach is close in spirit to the one of [9]. While it applies to a large class of random constraint satisfaction problems (including, e.g. coloring of random graphs), for the sake concreteness, we will focus on random k -SAT.

We expect the tree process to describe exactly the algorithm behavior in the limit of large instances, $n \rightarrow \infty$. While we could not prove this point, numerical simulations convincingly support this conjecture. Further, non-rigorous

A. Montanari is with Departments of Electrical Engineering and Statistics, Stanford University, montanari@stanford.edu, F. Ricci-Tersenghi is with Dipartimento di Fisica, Università di Roma La Sapienza, G. Semerjian is with Laboratoire de Physique Théorique de l’Ecole Normale, Paris

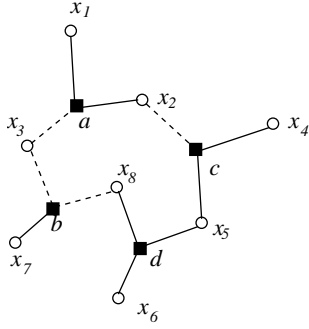


Fig. 1. Factor graph of a small 3-SAT instance. Continuous edges correspond to $z_j(a) = 0$, and dashed ones to $z_j(a) = 1$. The corresponding Boolean formula reads $(x_1 \vee x_2 \vee \bar{x}_3) \wedge (\bar{x}_2 \vee x_4 \vee x_5) \wedge (x_5 \vee x_6 \vee x_8) \wedge (\bar{x}_3 \vee x_7 \vee \bar{x}_8)$.

predictions based on tree calculations have been repeatedly successful in the analysis of random k -satisfiability. This approach goes under the name of ‘cavity method’ in statistical mechanics [6].

The paper is organized as follows. Section II contains some necessary background and notation on random k -SAT as well as a synthetic discussion of related work. In Section III we define the decimation procedure that we are going to analyze. We further provide the basic intuition behind the definition of the tree model. The latter is analyzed in Section IV, and the predictions thus derived are compared with numerical simulations in Section V. Finally, some conclusions and suggestions for future work are presented in Section VI. Proofs of several auxiliary lemmas are omitted from this extended abstract and deferred to technical appendices.

II. RANDOM k -SAT AND MESSAGE PASSING: BACKGROUND AND RELATED WORK

As mentioned above, *random k -SAT* refers to the uniform distribution over k -SAT instances with m constraints over n variables. More explicitly, each constraint is drawn uniformly at random among the $2^k \binom{n}{k}$ possible ones. We are interested here in the limit $n, m \rightarrow \infty$ with $m/n = \alpha$ fixed.

Consider the factor graph G of a random k -SAT formula, endowed with the graph-theoretic distance. Namely, the distance of two variable nodes $d(i, j)$ is the length of the shortest path leading from i to j on G . It is well known [10] that, in the large size limit, any finite neighborhood of a random node i converges in distribution to a well defined random tree. This observation will be the basis of our tree analysis of the decimation process, and is therefore worth spelling it out in detail. Let $B(i, \ell)$ be the subgraph induced by all the vertices $j \in G$, such that $d(i, j) \leq \ell$. Then $B(i, \ell) \xrightarrow{d} T(\ell)$ as $n \rightarrow \infty$, where $T(\ell)$ is the random rooted (factor) tree defined recursively as follows. For $\ell = 0$, $T(\ell)$ is the graph containing a unique variable node. For any $\ell \geq 1$, start by a single variable node (the root) and add $l \stackrel{d}{=} \text{Poisson}(\alpha k)$ clauses, each one including the root and $k - 1$ new variables (first generation variables). If $\ell \geq 2$, generate an independent copy of $T(\ell - 1)$ for each variable node in the first generation and attach it to them. The values $z_j(a)$ that violate clause a are independently chosen in $\{0, 1\}$

with equal probability. It is easy to see that the limit object $T(\infty)$ is well defined and is an infinite tree with positive probability if $\alpha > 1/k(k - 1)$.

We let $\alpha_s(k)$ be the largest value of α such that random k -SAT instances admit with high probability a solution. It is known [11] that $\alpha_s(k) = 2^k \log 2 - O(k)$. A sharp conjecture on the value of $\alpha_s(k)$ has been put forward in [6] on the basis of statistical physics calculations, implying $\alpha_s(k) \approx 4.267, 9.93, 21.12$ for (respectively) $k = 3, 4, 5$ and $\alpha_s(k) = 2^k \log 2 - \frac{1}{2}(1 + \log 2) + O(2^{-k})$ for large k [12].

Simple heuristics have been analyzed thoroughly [5] and proved to find a solution with probability bounded away from 0 if $\alpha \leq \text{const } 2^k/k$. Here the proportionality constant depends on the specific heuristic.

To the best of our knowledge, the first application of message passing algorithms to k -satisfiability is reported in [13]. In this early study BP was mostly applied in a *one-shot fashion* (as in iterative decoding of sparse graph codes [14]), without decimation. By this we mean that belief propagation is run, and resulting marginal probabilities are used to guess the values of all variables at once. However the probability of success of the one-shot algorithm is exponentially small: there are $\Theta(n)$ isolated constraints, whose variables have non-trivial marginal probabilities, each of them is hence violated with finite probability in the one-shot assignment.

Statistical mechanics methods allowed to derive a very precise picture of the solution set [15], [6], [8]. This inspired a new message passing algorithm dubbed *survey propagation* [7]. In conjunction with decimation, this algorithm allowed to solve random instances of unprecedentedly large sizes, in difficult regimes of α and k .

A natural way of introducing belief propagation for k -satisfiability is to consider the uniform distribution over solutions (assuming their existence). Let us denote by $\partial a = \{i_1(a), \dots, i_k(a)\}$ the set of variable nodes on which the a -th constraint effectively depends, for any subset U of the variable nodes their partial assignment $\underline{x}_U = \{x_i | i \in U\}$, and $w_a(\underline{x}_{\partial a}) = \mathbb{I}\{(x_{i_1(a)}, \dots, x_{i_k(a)}) \neq (z_1(a), \dots, z_k(a))\}$ the indicator function of the event ‘clause a is satisfied.’ The uniform distribution over the solutions can thus be written

$$\mu(\underline{x}) = \frac{1}{Z} \prod_{a \in F} w_a(\underline{x}_{\partial a}). \quad (1)$$

In [16] it was proved that for $\alpha \leq (2 \log k)/k [1 + o(1)]$, BP computes good approximations of the marginals of μ , irrespective of its initialization. It is clear from empirical studies [17], [18] that the ‘worst case’ argument used in this estimate (and in other papers on belief propagation [19], [20]) is far too pessimistic.

In Ref. [21] a simple message passing algorithm, warning propagation (see below), was analyzed for a modified (‘planted’) ensemble of random formulae. The algorithm was proved to converge and find solutions for large enough density α (see also [22], [23]). Both the ensemble and the algorithm are quite different from the ones treated in this paper.

Further, the definition and analysis of a ‘Maxwell decoder’ in [24], [25], is closely related to the approach in this paper. Let us recall that the Maxwell decoder was a (mostly conceptual) algorithm for implementing maximum likelihood decoding of LDPC codes over the erasure channel. The treatment in [24], [25] applies almost verbatim to a simple constraint satisfaction problem known as XORSAT. The generalization in the present paper is analogous to the one from the erasure to a general binary memoryless symmetric channel.

Finally, let us mention that BP decimation can be an interesting option in engineering applications, as demonstrated empirically in the case of lossy source coding [26], [27].

III. A SIMPLE DECIMATION PROCEDURE

A. Belief propagation

Let us recall the definition of BP for our specific setup ([2], [28] are general references). BP is a message passing algorithm: at each iteration messages are sent from variable nodes to neighboring clause nodes and vice versa. To describe the message update equations, we need some more notation. As in the case of factor nodes, we shall call ∂i the set of factors that depends on the variable x_i . If $i \in \partial a$, say $i = i_l(a)$, we denote $z(i, a) = z_l(a)$ the value of x_i which does not satisfy the a -th clause. For a pair of adjacent variable (i) and factor (a) nodes (i.e. $i \in \partial a$), let us call $\partial_+ i(a)$ (resp. $\partial_- i(a)$) the set of factor nodes adjacent to i , distinct from a , that agrees (resp. disagrees) with a on the satisfying value of x_i . In formulae, $\partial_+ i(a) = \{b \in \partial i \setminus a \mid z(i, b) = z(i, a)\}$ and $\partial_- i(a) = \{b \in \partial i \mid z(i, b) = 1 - z(i, a)\}$.

It is convenient to use log-likelihood notations for messages as is done in iterative decoding [14], with two caveats: (1) We introduce a factor 1/2 to be consistent with physics notation; (2) The message from variable node i to factor node a corresponds to the log-likelihood for x_i to satisfy/not-satisfy clause a (rather than to be 0/1).

Let $\{h_{i \rightarrow a}^{(r)}\}$, $\{u_{a \rightarrow i}^{(r)}\}$ denote the messages that are passed at time r along the directed edges $i \rightarrow a$ and $a \rightarrow i$, for $i \in V$, and $a \in F$. The update equations read

$$h_{i \rightarrow a}^{(r+1)} = \sum_{b \in \partial_+ i(a)} u_{b \rightarrow i}^{(r)} - \sum_{b \in \partial_- i(a)} u_{b \rightarrow i}^{(r)}, \quad (2)$$

$$u_{a \rightarrow i}^{(r)} = f(\{h_{j \rightarrow a}^{(r)}; j \in \partial a \setminus i\}), \quad (3)$$

where we define the function $f : \mathbb{R}^{k-1} \rightarrow \mathbb{R}$ as

$$f(h_1, \dots, h_{k-1}) = -\frac{1}{2} \log \left\{ 1 - \frac{1 - \epsilon}{2^{k-1}} \prod_{i=1}^{k-1} (1 - \tanh h_i) \right\}, \quad (4)$$

with $\epsilon = 0$ (this parameter is introduced for the discussion in Sec. V).

For $i \in V$, let $\partial_+ i$ be the subset of clauses that are satisfied by $x_i = 0$, and $\partial_- i$ the subset satisfied by $x_i = 1$. Then the BP estimate for the marginal of x_i under the measure $\mu(\cdot)$

is $\nu_i^{(r)}(x_i)$, where

$$\begin{aligned} \nu_i^{(r)}(0/1) &= \frac{1 \pm \tanh h_i^{(r)}}{2}, \\ h_i^{(r)} &= \sum_{a \in \partial_+ i} u_{a \rightarrow i}^{(r)} - \sum_{a \in \partial_- i} u_{a \rightarrow i}^{(r)}. \end{aligned} \quad (5)$$

B. Unit clause and warning propagation

During the decimation procedure a subset U of the variables are *fixed* to specific values, collectively denoted as \underline{x}_U^* . This has some *direct implications*. By this we mean that for some other variables x_j , $j \in V \setminus U$, it follows from ‘unit clause propagation’ (UCP) that they take the same value in all of the solutions compatible with the partial assignment \underline{x}_U^* . We will say that these variables are directly implied by the condition $\underline{x}_U = \underline{x}_U^*$. Let us recall that unit clause propagation corresponds to the following deduction procedure. For each of the fixed variables x_i , and each of the clauses a it belongs to, the value x_i^* can either satisfy clause a , or not. In the first case clause a can be eliminated from the factor graph. In the second a smaller clause with one less variable is implied. In both cases variable x_i is removed. It can happen that the size of a clause gets reduced to 1, through this procedure. In this case the only variable belonging to the clause must take a definite value in order to satisfy it. We say that such a variable is directly implied by the fixed ones. Whenever a variable is directly implied, its value can be substituted in all the clauses it belongs to, thus allowing further reductions.

The process stops for one of two reasons: (1) All the fixed or directly implied variables have been pruned and no unit clause is present in the reduced formula. In this case we refer to all variables that appeared at some point in a unit clause as *directly implied* variables. (2) Two unit clauses imply different values for the same variable. We will say that a *contradiction* was revealed in this case: no solution \underline{x} of the formula can verify the condition $\underline{x}_U = \underline{x}_U^*$.

A key element in our analysis is the remark that UCP admits a message passing description. The corresponding algorithm is usually referred to as *warning propagation* (WP) [29]. The WP messages (to be denoted as $h_{i \rightarrow a}^{(r)}$, $u_{a \rightarrow i}^{(r)}$) take values in $\{I, 0\}$. The meaning of $u_{a \rightarrow i}^{(r)} = I$ (respectively $u_{a \rightarrow i}^{(r)} = 0$) is: ‘variable x_i is (resp. is not) directly implied by clause a to satisfy it.’ For variable-to-factor messages, the meaning of $h_{i \rightarrow a}^{(r)} = I$ (respectively $h_{i \rightarrow a}^{(r)} = 0$) is: ‘variable x_i is (resp. is not) directly implied, through one of the clauses $b \in \partial i \setminus a$, not to satisfy clause a .’

We want to apply WP to the case in which a part of the variables have been fixed, namely $x_i = x_i^*$ for any $i \in U \subseteq V$. In this case the WP rules read

$$h_{i \rightarrow a}^{(r+1)} = \begin{cases} I & \text{if } \exists b \in \partial_- i(a) \text{ s.t. } u_{b \rightarrow i}^{(r)} = I \\ & \text{or } i \in U \text{ and } x_i^* = z(i, a), \\ 0 & \text{otherwise,} \end{cases} \quad (6)$$

$$u_{a \rightarrow i}^{(r)} = \begin{cases} I & \text{if } h_{j \rightarrow a}^{(r)} = I \forall j \in \partial a \setminus i, \\ 0 & \text{otherwise.} \end{cases} \quad (7)$$

BP-Decimation (k -SAT instance G)	
1:	initialize BP messages $\{h_{i \rightarrow a} = 0, u_{a \rightarrow i} = 0\}$;
2:	initialize WP messages $\{h_{i \rightarrow a} = 0, u_{a \rightarrow i} = 0\}$;
3:	initialize $U = \emptyset$;
4:	for $t = 1, \dots, n$
5:	run BP until the stopping criterion is met;
6:	choose $i \in V \setminus U$ uniformly at random;
7:	compute the BP marginal $\nu_i(x_i)$;
8:	choose x_i^* distributed according to ν_i ;
9:	fix $x_i = x_i^*$ and set $U \leftarrow U \cup \{i\}$;
10:	run WP until convergence;
11:	if a contradiction is found, return FAIL;
12:	end
13:	return \underline{x}^* .

TABLE I

THE BELIEF PROPAGATION-GUIDED DECIMATION ALGORITHM.

In the following we shall always assume that WP is initialized with $h_{i \rightarrow a}^{(0)} = 0, u_{a \rightarrow i}^{(0)} = 0$ for each edge $(ia) \in E$. It is then easy to prove that messages are monotone in the iteration number (according to the ordering $0 < \mathbb{I}$). In particular the WP iteration converges in at most $O(n)$ iterations. We denote $\{u_{a \rightarrow i}^{(\infty)}\}$ the corresponding fixed point messages, and say that $i \in V \setminus U$ is *WP-implied* by the fixed variables if there exist $a \in \partial i$ such that $u_{a \rightarrow i}^{(\infty)} = \mathbb{I}$. Then the equivalence between UCP and WP can be stated in the form below.

Lemma 1. *Assume a partial assignment \underline{x}_U^* to be given for $U \subseteq V$. Then*

- 1) *The fixed point WP messages $\{u_{a \rightarrow i}^{(\infty)}\}$ do not depend on the order of the WP updates (as long as any variable is updated an a priori unlimited number of times).*
- 2) *$i \in V \setminus U$ is directly implied iff it is WP-implied.*
- 3) *UCP encounters a contradiction iff there exists $i \in V, a \in \partial_+ i, b \in \partial_- i$ such that $u_{a \rightarrow i}^{(\infty)} = u_{b \rightarrow i}^{(\infty)} = \mathbb{I}$.*

For the clarity of what follows let us emphasize the terminology of *fixed* variables (those in U) and of *directly implied* variables (not in U , but implied by \underline{x}_U^* though UCP or WP). Finally, we will call *frozen* variables the union of fixed and directly implied ones, and denote the set of frozen variables by $W \subseteq V$.

C. Decimation

The BP-guided decimation algorithm is defined by the pseudocode of Table I. There are still a couple of elements we need to specify. First of all, how the BP equations (2), (3) are modified when a non-empty subset U of the variables is fixed. One option would be to eliminate these variables from the factor graph, and reduce the clauses they belong to accordingly. A simpler approach consists in modifying Eq. (2) when $i \in U$. Explicitly, if the chosen value x_i^* satisfies clause a , then we set $h_{i \rightarrow a}^{(r+1)} = +\infty$. If it does not, we set $h_{i \rightarrow a}^{(r+1)} = -\infty$.

Next, let us stress that, while WP is run until convergence, a not-yet defined ‘stopping criterion’ is used for BP. This will

be precised in Section V. Here we just say that it includes a maximum iteration number r_{\max} , which is kept of smaller order than $O(n)$.

The algorithm complexity is therefore naively $O(n^3 r_{\max})$. It requires n cycles, each involving: (1) at most r_{\max} BP iterations of $O(n)$ complexity and (2) at most n WP iterations of complexity $O(n)$. It is easy to reduce the complexity to $O(n^2 r_{\max})$ by updating WP in sequential (instead of parallel) order, as in UCP. Finally, natural choice (corresponding to the assumption that BP converges exponentially fast) is to take $r_{\max} = O(\log n)$, leading to $O(n^2 \log n)$ complexity.

In practice WP converges after a small number of iterations, and the BP updates are the most expensive part of the algorithm. This could be reduced further by using the fact that fixing a single variable should produce only a small change in the messages. Ref. [7] uses this argument for a similar algorithm to argue that $O(n \log n)$ time is enough.

D. Intuitive picture

Analyzing the dynamics of BP-decimation seems extremely challenging. The problem is that the procedure is not ‘myopic’ [5], in the sense that the value chosen for variable x_i depends on a large neighborhood of node i in the factor graph. By analogy with myopic decimation algorithms one expects the existence of a critical value of the clause density $\alpha_{\text{BPd}}(k)$ such that the algorithm finds a solution with probability bounded away from 0 for $\alpha < \alpha_{\text{BPd}}(k)$, while it is unsuccessful with high probability for $\alpha > \alpha_{\text{BPd}}(k)$. Notice that, if the algorithm finds a solution with positive probability, restarting it a finite number of times should¹ yield a solution with probability arbitrarily close to 1.

We shall argue in favor of this scenario and present an approach to analyze the algorithm evolution for α smaller than a *spinodal point* $\alpha_{\text{spin}}(k)$. More precisely, our analysis allows to compute the asymptotic fraction of ‘directly implied’ variables after any number of iterations. Further, the outcome of this computation provides a strong indication that $\alpha_{\text{spin}}(k) \leq \alpha_{\text{BPd}}(k)$. Both the analysis, and the conclusion that $\alpha_{\text{spin}}(k) \leq \alpha_{\text{BPd}}(k)$ are confirmed by large scale numerical simulations.

Our argument goes in two steps. In this section we show how to reduce the description of the algorithm to a sequence of ‘static’ problems. The resolution of the latter will be treated in the next section. Both parts rely on some assumptions on the asymptotic behavior of large random k -SAT instances, that originate in the statistical mechanics treatment of this problem [6], [8]. We will spell out such assumptions along the way.

As a preliminary remark, notice that the two message passing algorithms play different roles in the BP-decimation procedure of Table I. BP is used to estimate marginals of the uniform measure $\mu(\cdot)$ over solutions, cf. Eq. (1), in the first repetition of the loop. In subsequent repetitions, it is used to compute marginals of the conditional distribution,

¹A caveat: here we are blurring the distinction between probability with respect to the formula realization and the algorithm realization.

given the current assignment $\underline{x}_U = \underline{x}_U^*$. These marginals are in turn used to choose the values $\{x_i^*\}$ of variables to be fixed. WP is on the other hand used to check a necessary condition for the current partial assignment to be consistent. Namely it checks if it induces a contradiction on directly implied variables. In fact, it could be replaced by UCP, and, in any case, it does not influence the evolution of the partial assignment \underline{x}_U^* .

Let us introduce some notation: $(i(1), i(2), \dots, i(n))$ is the order in which variables are chosen at step 5 in the algorithm, $U_t = \{i(1), \dots, i(t)\}$ the set of fixed variables at the beginning of the $t+1$ -th repetition of the loop, and W_t the frozen variables at that time (i.e. the union of U_t and the variables directly implied by $\underline{x}_{U_t}^*$).

We begin the argument by considering an ‘idealized’ version of the algorithm where BP is replaced by a black box, that is able to return the exact marginal distribution of the measure conditioned on the previous choices, namely $\nu_i(\cdot) = \mu_{i|U}(\cdot | \underline{x}_U^*)$. Let us point out two simple properties of this idealized algorithm. First, it always finds a solution if the input formula is satisfiable (this will be the case with high probability if we assume $\alpha < \alpha_s(k)$). In fact, assume by contradiction that the algorithm fails. Then, there has been a last time t , such that the k -SAT instance has at least one solution consistent with the condition $\underline{x}_{U_{t-1}} = \underline{x}_{U_{t-1}}^*$, but no solution under the additional constraint $x_i = x_i^*$ for $i = i(t)$. This cannot happen because it would imply $\mu_{i|U_{t-1}}(x_i^* | \underline{x}_{U_{t-1}}^*) = 0$, and if this is the case, we would not have chosen x_i^* in step 8 of the algorithm.

The second consequence is that the algorithm output configuration \underline{x}^* is a uniformly random solution. This follows from our assumption since

$$\begin{aligned} \mathbb{P}\{\underline{x}^* | i(\cdot)\} &= \prod_{t=1}^n \nu_{i(t)}(x_{i(t)}^*) = \\ &= \prod_{t=1}^n \mu_{i(t)|U(t-1)}(x_{i(t)}^* | \underline{x}_{U(t-1)}^*) = \mu(\underline{x}^*). \end{aligned}$$

Therefore, the distribution of the state of the idealized algorithm after any number t of decimation steps can be described as follows. Pick a uniformly random solution \underline{x}^* , and a uniformly random subset of the variable indexes $U_t \subseteq V$, with $|U_t| = t$. Then fix the variables $i \in U_t$ to take value $x_i = x_i^*$, and discard the rest of the reference configuration \underline{x}^* (i.e. the bits x_j^* for $j \notin U_t$).

We now put aside the idealized algorithm and consider the effect of fixing the t -th variable $i(t)$ to $x_{i(t)}^*$. Three cases can in principle arise: (i) $x_{i(t)}$ was directly implied to be equal to $1 - x_{i(t)}^*$ by $\underline{x}_{U_{t-1}}^*$ and a contradiction is generated. We assume that BP is able to detect this direct implication and avoid such a trivial contradiction; (ii) $x_{i(t)}$ was directly implied to $x_{i(t)}^*$ by $\underline{x}_{U_{t-1}}^*$. The set of frozen variables remains the same, $W_t = W_{t-1}$, as this step is merely the actuation of a previous logical implication; (iii) $i(t)$ was not directly implied by $\underline{x}_{U_{t-1}}^*$. This is the only interesting case that we develop now.

Let us call $Z_t \equiv W_t \setminus W_{t-1}$ the set of newly frozen variables after this fixing step. A moment of reflection shows that Z_t contains $i(t)$ and that it forms a connected subset of V in G . Consider now the subgraph $G_t \subseteq G$ induced by Z_t (i.e. $G_t = (Z_t, F_t, E_t)$ where F_t is the set of factor nodes having at least one adjacent variable in Z_t , and E_t is the set of edges between Z_t and F_t). A crucial observation is the following:

Lemma 2. *If G_t is a tree, no contradiction can arise during step t .*

From this lemma, and since the factor graph of a typical random formula is locally tree-like, one is naturally lead to study the size of Z_t , i.e. of the cascade of newly implied variables induced by fixing the t -th variable. If this size remains bounded as $n \rightarrow \infty$, then G_t will typically be a tree and, consequently, contradictions will arise with vanishingly small probability during one step. If on the other hand the size diverges for infinitely large samples, then G_t will contain loops and opens the possibility for contradictions to appear.

In order to compute the typical size of Z_t , we notice that $|Z_t| = |W_t| - |W_{t-1}|$, and consider a t of order n , namely $t = n\theta$. If we let $\phi(\theta) \equiv \mathbb{E}[|W_{n\theta}|]/n$ denote the fraction of frozen variables when a fraction θ of variables have been fixed, then under mild regularity conditions we have

$$\lim_{n \rightarrow \infty} \mathbb{E}[|Z_{n\theta}|] = \frac{d\phi(\theta)}{d\theta}. \quad (8)$$

Of course ϕ will be an increasing function of θ . The argument above implies that, as long as its derivative remain finite for $\theta \in [0, 1]$, then the algorithm finds a solution. When the derivative diverges at some point θ_* , then the number of direct implications of a single variable diverges as well. The spinodal point $\alpha_{\text{spin}}(k)$ is defined as be the smallest value of α such that this happens.

The expectation in the definition of $\phi(\theta)$ is with respect to the choices made by the real BP algorithm in the first $n\theta$ steps, including the small mistakes it necessarily makes. Our crucial hypothesis is that the location of $\alpha_{\text{spin}}(k)$ does not change (in the $n \rightarrow \infty$ limit) if $\phi(\theta)$ is computed along the execution of the idealized decimation algorithm. In other words we assume that the cumulative effect of BP errors over n decimation steps produces only a small bias in the distribution of \underline{x}^* . For $\alpha \geq \alpha_{\text{BPd}}(k)$ this hypothesis is no longer consistent, as the real BP algorithm fails with high probability.

Under this hypothesis, and recalling the description of the state of the idealized algorithm given above, we can compute $\phi(\theta)$ as follows. Draw a random formula on n variables, a uniformly random ‘reference’ solution \underline{x}^* , a subset U of $n\theta$ variable nodes². Let $\phi_n(\theta)$ be the probability that a uniformly random variable node i is frozen, i.e. either in U or directly implied by \underline{x}_U^* . Then $\phi(\theta) = \lim_{n \rightarrow \infty} \phi_n(\theta)$. In the next Section this computation will be performed in the random tree model $T(\ell)$ of Sec. II.

²in the large n limit one can equivalently draw U by including in it each variable of V independently with probability θ .

IV. THE TREE MODEL AND ITS ANALYSIS

Let us consider a k -satisfiability formula whose factor graph is a finite tree, and the uniform measure μ over its solutions (which always exist) defined in Eq. (1). It follows from general results [2] that the recursion equations (2,3) have a unique fixed-point, that we shall denote $\{h_{i \rightarrow a}, u_{a \rightarrow i}\}$. Further the BP marginals $\nu_i(\cdot)$, cf. Eq. (5), are the actual marginals of μ . Drawing a configuration \underline{x} from the law μ is most easily done in a recursive, broadcasting fashion. Start from an arbitrary variable node i and draw x_i with distribution ν_i . Thanks to the Markov property of μ , conditional on the value of x_i , $\underline{x}_{V \setminus i}$ can be generated independently for each of the branches of the tree rooted at i . Namely, for each $a \in \partial i$, one draws $\underline{x}_{\partial a \setminus i}$ from

$$\mu(\underline{x}_{\partial a \setminus i} | x_i) = \frac{1}{z} w_a(x_i, \underline{x}_{\partial a \setminus i}) \prod_{j \in \partial a \setminus i} \nu_{j \rightarrow a}(x_j). \quad (9)$$

Here z is a normalization factor and $\nu_{i \rightarrow a}(\cdot)$ denotes the marginal of the variable x_i in the amputated factor graph where factor node a has been removed (this is easily expressed in terms of the message $h_{i \rightarrow a}$). Once all variables j at distance 1 from i have been generated, the process can be iterated to fix variables at distance 2 from i , and so on. It is easy to realize that this process indeed samples a solution uniformly at random.

Following the program sketched in the previous Section, we shall study the effect of fixing a subset of the variables to the value they take in one of the solutions. We first state the following lemma.

Lemma 3. *Suppose U is a subset of the variables of a tree formula, and let \underline{x}^* be a uniformly random solution. The probability that a variable $i \notin U$ is directly implied by \underline{x}_U^* reads*

$$\nu_i(0) \left\{ 1 - \prod_{a \in \partial_+ i} (1 - \widehat{u}_{a \rightarrow i}) \right\} + \nu_i(1) \left\{ 1 - \prod_{a \in \partial_- i} (1 - \widehat{u}_{a \rightarrow i}) \right\}, \quad (10)$$

where the new messages $\{\widehat{u}_{a \rightarrow i}, \widehat{h}_{i \rightarrow a}\}$ are solutions of

$$\widehat{h}_{j \rightarrow a} = \begin{cases} 1 & \text{if } j \in U \\ 1 - \prod_{b \in \partial_- j(a)} (1 - \widehat{u}_{b \rightarrow j}) & \text{otherwise} \end{cases} \quad (11)$$

$$\widehat{u}_{a \rightarrow i} = \prod_{j \in \partial a \setminus i} \left(\frac{1 - \tanh h_{j \rightarrow a}}{2} \widehat{h}_{j \rightarrow a} \right). \quad (12)$$

We consider now a random tree factor graph and a random set of fixed variables U .

Lemma 4. *Consider a random tree formula $\mathbb{T}(\ell)$ obtained from the construction of Section II, and a random subset U of its variable nodes defined by letting $j \in U$ independently with probability θ for each j . Finally, let \underline{x}^* be a uniformly random solution of $\mathbb{T}(\ell)$. Then the probability that the root of $\mathbb{T}(\ell)$ is frozen (either fixed or directly implied by \underline{x}_U^*) is*

$$\phi_\ell^{\text{tree}}(\theta) = \mathbb{E}_\ell \left[(1 - \tanh h) \widehat{h} \right], \quad (13)$$

where $\mathbb{E}_\ell[\cdot]$ denotes expectation with respect to the distribution of $(h, \widehat{h})_\ell$. This is a (vector) random variable defined by recurrence on ℓ as

$$(h, \widehat{h})_\ell \stackrel{\text{d}}{=} \left(\sum_{i=1}^{l_+} u_i^+ - \sum_{i=1}^{l_-} u_i^-, 1 - \zeta \prod_{i=1}^{l_-} (1 - \widehat{u}_i^-) \right), \quad (14)$$

$$(u, \widehat{u})_{\ell+1} \stackrel{\text{d}}{=} \left(f(h_1, \dots, h_{k-1}), \prod_{i=1}^{k-1} \frac{1 - \tanh h_i}{2} \widehat{h}_i \right), \quad (15)$$

with initial condition $(u, \widehat{u})_{l=0} = (0, 0)$ with probability 1. In this recursion l_+ and l_- are two independent Poisson random variables of parameter $\alpha k/2$, ζ is a random variable equal to 0 (resp. 1) with probability θ (resp. $1 - \theta$), the $\{(u_i^+, \widehat{u}_i^+), (u_i^-, \widehat{u}_i^-)\}$ and (h_i, \widehat{h}_i) are independent copies of, respectively, $(u, \widehat{u})_\ell$ and $(h, \widehat{h})_\ell$.

To obtain a numerical estimate of the function $\phi^{\text{tree}}(\theta) = \lim_{\ell \rightarrow \infty} \phi_\ell^{\text{tree}}(\theta)$ we resorted to sampled density evolution (also called ‘population dynamics’ in the statistical physics context [6]), using samples of 10^5 elements and $k = 4$ as a working example, see Fig. 2. For small values of α , $\phi^{\text{tree}}(\theta)$ is smoothly increasing and slightly larger than θ . Essentially all frozen variables are fixed ones, and very few directly implied variables appear. Moreover the maximal slope of the curve is close to 1, implying that the number of new frozen variables at each step, Z_t , remains close to 1. As α grows, $\phi^{\text{tree}}(\theta)$ becomes significantly different from θ , and the maximal slope encountered in the interval $\theta \in [0, 1]$ gets larger. At a value $\alpha_{\text{spin}}^{\text{tree}}(k)$ the curve $\phi^{\text{tree}}(\theta)$ acquires a vertical tangent at $\theta_*(\alpha_{\text{spin}}^{\text{tree}})$, signaling the divergence of the size of the graph of newly implied variables. Density evolution gives us $\alpha_{\text{spin}}^{\text{tree}}(k = 4) \approx 8.05$, with an associated value of $\theta_* \approx 0.35$. For $\alpha > \alpha_{\text{spin}}^{\text{tree}}(k)$ the curve $\phi^{\text{tree}}(\theta)$ has more than one branch, corresponding to the presence of multiple fixed points for $\theta \in [\theta_0(\alpha), \theta_*(\alpha)]$. In analogy with [25], we expect the evolution of the algorithm to be described by picking (for each θ) the lowest branch of $\phi^{\text{tree}}(\theta)$. The resulting curve has a discontinuity at $\theta_*(\alpha)$, which is a slowly decreasing function of α .

We expect the tree computation to provide the correct prediction for the actual curve $\phi(\theta)$ (i.e. $\phi^{\text{tree}}(\theta) = \phi(\theta)$) for a large range of the satisfiable regime, including $[0, \alpha_{\text{spin}}^{\text{tree}}(k)]$. As a consequence, we expect $\alpha_{\text{spin}}(k) = \alpha_{\text{spin}}^{\text{tree}}(k)$ and BP decimation to be successful up to $\alpha_{\text{spin}}(k)$. Similar tree computations are at the basis of a number of statistical mechanics computations in random k -SAT and have been repeatedly confirmed by rigorous studies.

The relation between tree and graph can be formalized in terms of Aldous [30] local weak convergence method. Fix a finite integer ℓ and consider the finite neighborhood $\mathbb{B}(\ell)$ of radius ℓ around an arbitrarily chosen variable node of an uniformly drawn factor graph G on n variables. Denote by $\mu_{\mathbb{B}(\ell), n}(\cdot)$ the law of $\underline{x}_{\mathbb{B}(\ell)}$ when \underline{x} is a uniformly random solution. We proceed similarly in the random tree ensemble. Draw a random tree $\mathbb{T}(L)$ with $L \geq \ell$, let $\mathbb{T}(\ell)$ its first ℓ generations, and $\mu_{\mathbb{T}(\ell), L}^{\text{tree}}(\cdot)$ the distribution of $\underline{x}_{\mathbb{T}(\ell)}$.

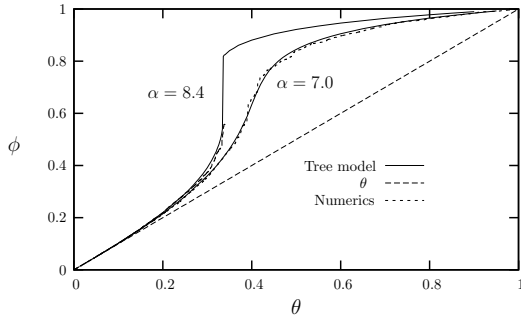


Fig. 2. Fraction of frozen variables as a function of the fraction of fixed variables. Comparison between the tree model and the algorithmic numerical results, for 4-satisfiability formulas with $n = 4000$, $\alpha = 7$ and $\alpha = 8.4$.

Considerations building on the field of statistical mechanics of disordered systems leads to the following hypothesis.

Conjecture 1. *There exists a sequence $\alpha_c(k)$ such that $\mu_{\mathbb{T}(\ell)}(\cdot) \stackrel{d}{=} \mu_{\mathbb{T}(\ell)}^{\text{tree}}(\cdot)$ for all $\alpha < \alpha_c(k)$, i.e. $(\mathbb{B}(\ell), \mu_{\mathbb{B}(\ell), n}(\cdot))$ and $(\mathbb{T}(\ell), \mu_{\mathbb{T}(\ell), L}^{\text{tree}}(\cdot))$ have the same weak limit. A precise determination of $\alpha_c(k)$ was presented in [8], yielding $\alpha_c(k) \approx 3.86, 9.55, 20.80$ for, respectively, $k = 3, 4, 5$, and $\alpha_c(k) = 2^k \log 2 - \frac{3}{2} \log 2 + O(2^{-k})$ at large k .*

Local weak limits of combinatorial models on random graphs were recently considered in [31]. For a generalized conjecture in the regime $[\alpha_c(k), \alpha_s(k)]$ see [32].

A slightly stronger version of this conjecture would imply that $\phi(\theta) = \phi^{\text{tree}}(\theta)$. As a consequence (following the discussion in previous section) the tree model would correctly describe the algorithm evolution.

V. NUMERICAL SIMULATIONS

In order to test the validity of our analysis we performed numerical simulations of the pseudo-code of Table I. Let us give a few further details on its implementation. The BP messages are stored as $\{\tanh h_{i \rightarrow a}, \tanh u_{a \rightarrow i}\}$. Ambiguities in the update rule (3) arises when $\tanh u_{b \rightarrow i} = \tanh u_{c \rightarrow i} = 1$ with $b \in \partial_+ i(a)$ and $c \in \partial_- i(a)$. Because of numerical imprecisions this situation can occur even before a contradiction has been detected by WP; such ambiguities are resolved by recomputing the incoming messages $\tanh u_{b \rightarrow i}$ using the regularized version of Eq. (4) with a small positive value of ϵ (in practice we used $\epsilon = 10^{-4}$).

As for the stopping criterion used in step 5, we leave the BP iteration loop if either of the two following criteria is fulfilled: (1), $\sup_i |\tanh h_i^{(r)} - \tanh h_i^{(r-1)}| < \delta$, i.e. BP has converged to a fixed-point within a given accuracy; (2) A maximal number of iterations r_{\max} fixed beforehand has been reached. In our implementation we took $\delta = 10^{-10}$ and $r_{\max} = 200$.

A first numerical check is presented in Fig. 2. The two dashed curves represent the fraction of frozen variables along the execution of the BP guide decimation algorithm, for two formulae of the 4-sat ensemble, of moderate size ($n = 4000$). The first formula had a ratio of constraints per variable

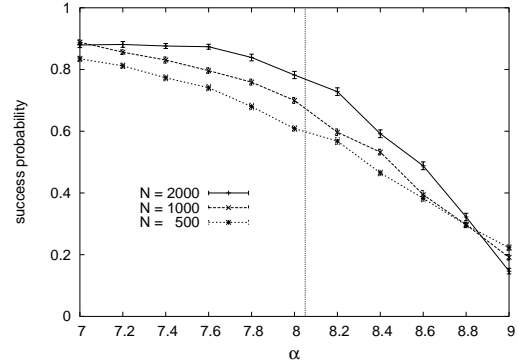


Fig. 3. Probability of success of the BP decimation algorithm as a function of the clause density α in random 4-SAT. The vertical line corresponds to the threshold $\alpha_{\text{spin}}(4)$. Our analysis indicates that BP decimation finds a solution with probability bounded away from 0 for $\alpha < \alpha_{\text{spin}}(4)$.

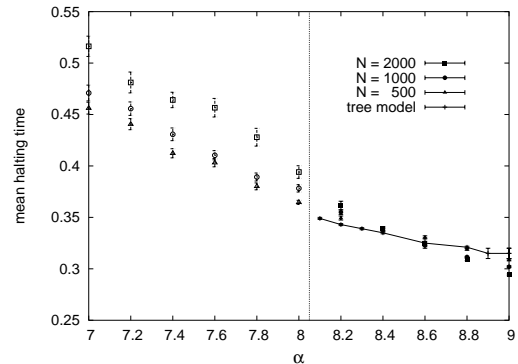


Fig. 4. Mean halting time for the BP decimation algorithm in random 4-SAT. The vertical line corresponds to the threshold $\alpha_{\text{spin}}(4)$. The mean is taken over unsuccessful runs. For $\alpha < \alpha_{\text{spin}}(4)$ a large fraction of the runs is successful and do not contribute to the mean.

$\alpha = 7 < \alpha_{\text{spin}}$. In agreement with the picture obtained from the analytical computation, the algorithm managed to find a solution of the formula (no contradiction encountered) and the measured fraction of frozen variables follows quite accurately the tree model prediction. The second formula was taken in the regime $\alpha_{\text{spin}} < \alpha = 8.4 < \alpha_c$. The algorithm halted because a contradiction was found, after roughly the fraction θ_* (computed from the tree model) of variables has been fixed. The portion of the curve before this event exhibits again a rather good agreement between the direct simulation and the model.

Figure 3 shows the probability of success of BP decimation in a neighborhood of $\alpha_{\text{spin}}(4)$ for random formulae of size $n = 500, 1000, 2000$. Each data point is obtained by running the algorithm on 1000 to 3000 formulae. The data strongly suggest that the success probability is bounded away from 0 for $\alpha < \alpha_{\text{spin}}(k)$, in agreement with our argument.

Finally, in Figure 4 we consider the number of variables t_* fixed by BP decimation before a contradiction is encountered. According to the argument in Section III-D, t_*/n should concentrate around the location θ_* of the discontinuity in $\phi(\theta)$. This is in fact the point at which the number of variables directly implied by a fixed one is no longer

bounded. The comparison is again encouraging. Notice that for $\alpha < \alpha_{\text{spin}}(k)$ we do not have any prediction, and the estimate of t_* concerns only a small fraction of the runs.

To summarize, our simulations support the claim that, for $\alpha < \alpha_{\text{spin}}(k)$ the success probability is strictly positive and the algorithm evolution follows the tree model. For $\alpha > \alpha_{\text{spin}}(k)$ the main failure mechanism is indeed related to unbounded cascades of directly implied variables, after about $n\theta_*$ steps.

VI. CONCLUSIONS AND FUTURE DIRECTIONS

Let us conclude by highlighting some features of this work and proposing some directions for future research. It is worth mentioning that, as was also found in [8], random 3-sat has a qualitatively different behavior compared to random k -sat with $k \geq 4$. In particular we did not find any evidence for the existence of a vertical tangent point in the $k = 3$ function $\phi(\theta)$ in the regime we expect to control through the tree computation, namely $\alpha < \alpha_c(3) \approx 3.86$.

Our analysis suggests that BP guided decimation is successful with positive probability for $\alpha \leq \alpha_{\text{spin}}(k)$. Further we argued that this threshold can be computed through a tree model and evaluated via density evolution. Despite these conclusions are based on several assumptions, it is tempting to make a comparison with the best rigorous results on simple decimation algorithms. For $k = 4$ the best result was obtained by Frieze and Suen [33] who proved SCB (shortest clause with limited amount of backtracking) to succeed for $\alpha < 5.54$. This is far from the conjectured threshold of BP decimation that is $\alpha_{\text{spin}}(4) \approx 8.05$. For large k , an asymptotic expansion suggests that

$$\alpha_{\text{spin}}(k) = e \frac{2^k}{k} (1 + O(k^{-1})), \quad (16)$$

whereas SCB is known from [33] to reach clause densities of $c_k 2^k/k$, with $c_k \rightarrow 1.817$ as $k \rightarrow \infty$. A rigorous version of our analysis would lead to a constant factor improvement. On the other hand, the quest for an algorithm that provably solves random k -SAT in polynomial time beyond $\alpha = O(2^k/k)$, is open.

From a practical point of view the decimation strategy studied in this paper is not the most efficient one. A seemingly slight modification of the pseudo-code of Table I consists in replacing the uniformly random choice of the variable to be fixed, privileging the ones with the most strongly biased marginals. The intuition for this choice is that these marginals are the less subject to the 'small errors' of BP. The numerical results reported in [8] suggest that this modification improves significantly the performances of the decimation algorithm; unfortunately it also makes its analysis much more difficult.

This work was partially supported by EVERGROW, integrated project No. 1935 in the complex systems initiative of the Future and Emerging Technologies directorate of the IST Priority, EU Sixth Framework.

REFERENCES

- [1] N. Creignou, S. Khanna and M. Sudan, *Complexity classifications of boolean constraint satisfaction problems*, SIAM, Philadelphia, 2001
- [2] F. R. Kschischang, B. J. Frey and H-A. Loeliger, "Factor graphs and the sum-product algorithm" (2001), *IEEE Trans. Inform. Theory* **47**, 498-519.
- [3] B. Selman, H. Levesque and D. Mitchell, "A New Method for Solving Hard Satisfiability Problems", 10th Natl Conf. on Artif. Intell., San Jose, CA, 440-446
- [4] J. Franco, "Results related to threshold phenomena research in satisfiability: lower bounds", *Theoret. Comput. Sci.* **265**, 147-157 (2001).
- [5] D. Achlioptas and G. B. Sorkin, *Proc. of the Annual Symposium on the Foundations of Computer Science*, Redondo Beach, CA, November 2000
- [6] M. Mézard, G. Parisi, and R. Zecchina, "Analytic and Algorithmic Solution of Random Satisfiability Problems", *Science* **297** (2002), 812-815.
- [7] M. Mézard and R. Zecchina, "The random K -satisfiability problem: from an analytic solution to an efficient algorithm", *Phys. Rev. E* **66** (2002), 056126.
- [8] F. Krzakala, A. Montanari, F. Ricci-Tersenghi, G. Semerjian and L. Zdeborova, "Gibbs States and the Set of Solutions of Random Constraint Satisfaction Problems," *Proc. Natl. Acad. Sci.* **104** (2007) 10318-10323.
- [9] M. Luby, M. Mitzenmacher and A. Shokrollahi, "Analysis of Random Processes via And-Or Tree Evaluation" *Proc. of the Symposium on Discrete Algorithms*, San Francisco, January 2008.
- [10] S. Janson, T. Luczak and A. Ruciński", *Random graphs*, John Wiley, New York, 2000
- [11] D. Achlioptas and Y. Peres, "The threshold for random k -SAT is $2k \log 2 - O(k)$ ", *Journal of the AMS*, **17** (2004), 947-973.
- [12] S. Mertens, M. Mézard and R. Zecchina, "Threshold values of random K -SAT from the cavity method", *Random Struct. Alg.* **28**, 340-373 (2006).
- [13] S. J. Pumphrey, "Solving the Satisfiability Problem Using Message Passing Techniques," Cambridge Physics Project Report.
- [14] T. Richardson and R. Urbanke, *Modern Coding Theory*, draft available at <http://lthcwww.epfl.ch/mct/index.php>
- [15] G. Biroli, R. Monasson and M. Weigt, "A variational description of the ground state structure in random satisfiability problems", *Eur. Phys. J. B* **14**, 551 (2000).
- [16] A. Montanari and D. Shah, "Counting good truth assignments of random k -SAT formulae," *Proc. of the Symposium on Discrete Algorithms*, New Orleans, January 2007.
- [17] E. Aurell, U. Gordon and S. Kirkpatrick, *Proc. of Neural Information Processing Symposium*, Vancouver, 2004.
- [18] E. N. Maneva, E. Mossel and M. J. Wainwright, *Proc. of the Symposium on Discrete Algorithms*, Vancouver, January 2005.
- [19] S. Tatikonda and M. Jordan, "Loopy Belief Propagation and Gibbs Measures", *Proc. Uncertainty in Artificial Intell.* **18** (2002) 493-500.
- [20] D. Gamarnik and A. Bandyopadhyay, *Proc. of the Symposium on Discrete Algorithms*, Miami, (2006).
- [21] U. Feige, E. Mossel and D. Vilenchik, "Complete convergence of message passing algorithms for some satisfiability problems" *Proc. RANDOM*, Barcelona, (2006).
- [22] A. Coja-Oghlan, M. Krivelevich and D. Vilenchik, "Why almost all k -CNF formulas are easy", *Proceedings of the 13th International Conference on Analysis of Algorithms*, to appear, (2007).
- [23] F. Altarelli, R. Monasson and F. Zamponi, "Can rare SAT formulae be easily recognized? On the efficiency of message-passing algorithms for K -SAT at large clause-to-variable ratios", *J. Phys. A: Math. Theor.* **40**, 867-886 (2007).
- [24] C. Measson, A. Montanari, T. Richardson and R. Urbanke, "Life Above Threshold: From List Decoding to Area Theorem and MSE," *IEEE Inform. Theory Workshop*, San Antonio, October 2004
- [25] C. Measson, A. Montanari and R. Urbanke, "Maxwell Construction: The Hidden Bridge between Iterative and Maximum a Posteriori Decoding," *IEEE Trans. Inform. Theory*, to be published.
- [26] E. Maneva and M. Wainwright, "Lossy source encoding via message-passing and decimation over generalized codewords of LDGM codes," *IEEE Inform. Theory Symposium*, Adelaide, September 2004
- [27] S. Ciliberti, M. Mézard and R. Zecchina, "Lossy data compression with random gates," *Phys. Rev. Lett.* **95**, 038701 (2005)
- [28] J. Pearl, "Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference", San Francisco, CA: Morgan Kaufmann, 1988.

- [29] A. Braunstein, M. Mézard and R. Zecchina, “Survey propagation: an algorithm for satisfiability”, *Random Structures and Algorithms* **27**, 201-226 (2005).
- [30] D. Aldous and J.M. Steele, “The objective method”, in *Probability on Discrete Structures*, H. Kesten ed., 1, Springer (2003).
- [31] A. Gerschenfeld and A. Montanari, *Proc. of the Annual Symposium on the Foundations of Computer Science*, Providence, RI, October 2007
- [32] G. Semerjian, “On the freezing of variables in random constraint satisfaction problems”, arXiv/0705.2147.
- [33] A. Frieze and S. Suen, “Analysis of Two Simple Heuristics on a Random Instance of k-SAT”, *Journal of Algorithms* **20**, 312-355 (1996).

APPENDIX

Proof of Lemma 1. The statement is completely analogous to the equivalence between message passing and peeling versions of erasure decoding for LDPC codes [14]. Since the proof follows the same lines as well, we will limit ourselves to sketch its main points.

1) Let $\{u_{a \rightarrow i}\}$ and $\{u'_{a \rightarrow i}\}$ be two fixed points of WP. Then $\{\min(u_{a \rightarrow i}, u'_{a \rightarrow i})\}$ is a fixed point as well. It follows that the ‘minimal’ fixed point is well defined and that it coincides with the limit of $\{u_{a \rightarrow i}^{(r)}\}$ irrespective of the order of WP updates.

2) Consider the ordering $\{i(1), i(2), \dots, i(q)\}$ according to which variables are declared as directly implied within UCP. For each $s \in \{1, \dots, q\}$ there is at least one unit clause involving only variable $i(s)$ before this was declared. Call this $a(s)$. Then use the same update order for WP, namely update, in sequence message $u_{a(s) \rightarrow i(s)}$, and all the messages $h_{i(s) \rightarrow b}$ for $b \neq a(s)$. It is immediate to show that this leads to a fixed point, and the resulting WP-implied variables coincide with the directly implied variables. The proof is completed by using point 1.

3) Consider the same ordering of variables used in point 2 above. If there exists $i \in V$, $a \in \partial_+ i$, $b \in \partial_- i$ as in the statement, then UCP must have reduced both clauses a and b to a unit clause involving x_i and requiring it to take different values. Viceversa if UCP produces such a situation, in the WP updates $u_{a \rightarrow i}^{(r)} = u_{b \rightarrow i}^{(r)} = \text{I}$ after some time r . \square

Proof of Lemma 2. The same statement has been proved for the Maxwell decoder [25]. We therefore briefly recall the basic ideas used in that case.

First of all the only WP messages changing from step $t - 1$ to step t (call these the ‘new’ messages) are the ones on the edges of the tree G_t , and directed outwards. As a consequence, no contradiction can arise because of two contradicting new messages, because no variable node has two incoming new messages.

There could be, in line of principle, a contradiction between a new and an old message. The crucial observation is that indeed any factor node in F_t has at most two adjacent variable nodes in Z_t (because otherwise it could not ‘transmit’ an implication). If a variable node i already receives some I message at time $t - 1$ from clause a , then it cannot receive any new message at time t from a different clause b . This because the message $i \rightarrow b$ must already be I, and therefore clause b is already effectively ‘reduced’.

An alternative argument consists in considering the equivalent UCP representation. If G_t is a tree, then no variable appears twice in a unit clause, and therefore no contradiction arises. \square

Proof of Lemma 3. Since we are dealing with a tree graph, equations (11,12) admit a unique solution, determined from the boundary condition $\hat{h}_{i \rightarrow a} = 1$ (resp. $\hat{h}_{i \rightarrow a} = 0$) if i is a leaf in U (resp. a leaf outside of U). The newly introduced messages have the following interpretation. Imagine running WP, cf. Eqs. (6), (7) to find which variables are directly implied by \underline{x}_U . Then $\hat{u}_{a \rightarrow j}$ is the probability that $u_{a \rightarrow j} = \text{I}$ when \underline{x}_U is drawn conditional on x_j satisfying a . Further, $\hat{h}_{j \rightarrow a}$ is the probability that $h_{j \rightarrow a} = \text{I}$ when \underline{x}_U is drawn conditional on x_j not satisfying clause a .

Now, suppose x_i has been fixed to x_i^* drawn according to its marginal (hence the two terms in Eq. (10)) and a configuration \underline{x} has been generated conditional on x_i , through the broadcast construction. Then the configuration of the variables in U is retained, $\underline{x}_U = \underline{x}_U^*$, and the rest of \underline{x} is discarded. The status (directly implied or not) of x_i is read off from the values of the messages $u_{a \rightarrow i}$ it receives. It is easy to convince oneself that x_i cannot be implied to take the value opposite to the one it took at the beginning of the broadcasting: by definition \underline{x}_U is compatible with it. Equation (10) follows by computing the probability that at least one of the messages $u_{a \rightarrow i}$ is equal to I among the ones from clauses a that are satisfied by x_i^* .

Equation (11) is derived by applying the same argument to the branch of the tree rooted at j and not including factor node a . Finally, to derive Eq. (12) notice that, in order for variable x_l to be directly implied to satisfy clause a , each of the variables $j \in \partial a \setminus l$ must be implied by the corresponding subtree not to satisfy a . From the above remark, this can happen only if none of the $\{x_j^*\}$ satisfies a . The probability of this event is easily found from (9) to be

$$\prod_{j \in \partial a \setminus i} \frac{1 - \tanh h_{j \rightarrow a}}{2}. \quad (17)$$

\square

Proof of Lemma 4. Denote by ρ the root of $T(\ell)$. Conditional on the realization of the tree and of the set U , the probability of a direct implication of the root is obtained by solving (2), (3), (11), (12) for the edges directed towards the root, which leads to couples of messages $\{(h_{i \rightarrow a}, \hat{h}_{i \rightarrow a})\}$ and $\{(u_{a \rightarrow i}, \hat{u}_{a \rightarrow i})\}$ along the edges of $T(\ell)$. Since $T(\ell)$ and U are random these couples of messages are random variables as well.

We claim that for $\ell \geq 1$, the messages $(u_{a \rightarrow \rho}, \hat{u}_{a \rightarrow \rho})$ sent to the root of $T(\ell)$ by the adjacent constraint nodes are distributed as $(u, \hat{u})_\ell$. Similarly for $\ell \geq 0$, $(h, \hat{h})_\ell$ has the distribution of the messages sent from the first generation variables to their ancestor constraint node in a random $T(\ell + 1)$. This claim is a direct consequence of Eqs. (2), (3), (11), (12) and of the definition of $T(\ell)$ and U . The random variables l_\pm have, for instance, the distribution of the

cardinalities of $\partial_{\pm}i(a)$ for an arbitrary edge of the random tree, as $|\partial i \setminus a| \stackrel{d}{=} \text{Poisson}(\alpha k)$ and unsatisfying values $z(i, a)$ of the variables are chosen independently with equal probability.

Finally the expression of $\phi_{\ell}^{\text{tree}}(\theta)$ is obtained from (10) by noting that the cardinalities of $\partial_{\pm}i$ for the root of $T(\ell)$ are distributed as the ones of $\partial_{\pm}i(a)$ and using the global symmetry between 0 and 1, which implies that on average the two terms of (10) yield the same contribution. Note that the dependence on θ of $\phi_{\ell}^{\text{tree}}$ arises through the distribution of $(h, \hat{h})_{\ell}$, the bias of the coin ζ used in (14) being θ . \square

Details on the population dynamics algorithm. The numerical procedure we followed in order to determine $\phi_{\ell}^{\text{tree}}(\theta)$ amounts to approximating the distribution of the random variable $(u, \hat{u})_{\ell}$ by the empirical distribution of a large sample of couples $\{(u_j, \hat{u}_j)\}_{j=1}^N$. A sample $\{(h_j, \hat{h}_j)\}_{j=1}^N$ is then generated according to Eq. (14): for each $j \in [N]$ one draws two Poisson random variables l_+ and l_- , $l_+ + l_-$ indexes j_i^{\pm} uniformly in $[N]$, and a biased coin ζ . The j -th element of the sample is thus computed as

$$(h_j, \hat{h}_j) = \left(\sum_{i=1}^{l_+} u_{j_i^+} - \sum_{i=1}^{l_-} u_{j_i^-}, 1 - \zeta \prod_{i=1}^{l_-} (1 - \hat{u}_{j_i^-}) \right).$$

Subsequently the sample $\{(u_j, \hat{u}_j)\}$ is updated from $\{(h_j, \hat{h}_j)\}$ by a similar interpretation of Eq. (15). After ℓ iterations of these two steps, starting from the initial configuration $(u_j, \hat{u}_j) = (0, 0)$ for all $j \in [1, N]$, the estimate of $\phi_{\ell}^{\text{tree}}(\theta)$ is given by

$$\frac{1}{N} \sum_{j=1}^N (1 - \tanh h_j) \hat{h}_j. \quad (18)$$

When ℓ gets large this quantity is numerically found to converges to a limit we denoted $\phi^{\text{tree}}(\theta)$. \square

Large k argument. Consider the function $\hat{\phi}(\theta)$ defined, for $\theta \in [0, 1]$, as the smallest solution in $[0, 1]$ of the equation

$$\hat{\phi} = \theta + (1 - \theta) \left(1 - \exp \left[-\frac{\alpha k}{2^k} \hat{\phi}^{k-1} \right] \right). \quad (19)$$

It can be shown that $\hat{\phi}(\theta)$ is a smoothly increasing function of θ as long as $\alpha < \hat{\alpha}_{\text{spin}}(k)$, while for larger values of α a discontinuous jump develops in its curve. This threshold can be explicitly computed and reads

$$\hat{\alpha}_{\text{spin}}(k) = \frac{2^k}{k} \left(\frac{k-1}{k-2} \right)^{k-2}. \quad (20)$$

We believe this simple to determine function $\hat{\phi}(\theta)$ to be equivalent to the true $\phi(\theta)$ in the large k limit, up to exponentially small in k corrections. In fact (13,14,15) implies the following exact equation,

$$\mathbb{E}[\hat{h}] = \theta + (1 - \theta) \left(1 - \exp \left[-\frac{\alpha k}{2^k} \phi(\theta)^{k-1} \right] \right), \quad (21)$$

where the expectation is taken in the $\ell \rightarrow \infty$ limit. For large values of k one can show the random variable h to be exponentially close to 0, hence $\phi(\theta)$ and $\mathbb{E}[\hat{h}]$ coincide at the leading order, and by comparing (19) and (21) they also coincide with $\hat{\phi}(\theta)$. The conjecture stated in Eq. (16) was obtained by expanding (20) at the leading order. \square