THEORY OF COMPUTING, Volume 9 (19), 2013, pp. 617–651
www.theoryofcomputing.org

Complete Convergence of Message Passing Algorithms for Some Satisfiability Problems*

Uriel Feige Elchanan Mossel† Dan Vilenchik

Received January 22, 2012; Revised May 4, 2013; Published June 15, 2013

Abstract: In this paper we analyze the performance of Warning Propagation, a popular message passing algorithm. We show that for 3CNF formulas drawn from a certain distribution over random satisfiable 3CNF formulas, commonly referred to as the planted-assignment distribution, running Warning Propagation in the standard way (run message passing until convergence, simplify the formula according to the resulting assignment, and satisfy the remaining subformula, if necessary, using a simple "off the shelf" heuristic) works when the clause-variable ratio is a sufficiently large constant. We are not aware of previous rigorous analysis showing the effectiveness of message passing algorithms for satisfiability instances.

ACM Classification: F.2.2, G.3

AMS Classification: 68Q25, 90C59

Key words and phrases: constraint satisfaction, average case, SAT, statistical physics

1 Introduction

A CNF formula over the variables $x_1, x_2, \ldots, x_n$ is a conjunction of clauses $C_1, C_2, \ldots, C_m$ where each clause is a disjunction of one or more literals. Each literal is either a variable or its negation. A formula is said to be in k-CNF form if every clause contains exactly k literals. A CNF formula is satisfiable if there is a Boolean assignment to the variables such that every clause contains at least one literal which evaluates to true. 3SAT, the language of all satisfiable 3CNF formulas, is well known to be NP-complete [13].

*An extended abstract of this paper appeared in the Proceedings of RANDOM 2006 [16].

†Supported by DOD ONR grants N000141110140 and N0014-07-1-05-06.

 DOI: 10.4086/toc.2013.v009a019

The plethora of worst-case NP-hardness results for many interesting optimization problems motivates the study of heuristics that give "useful" answers for "typical" subset of the problem instances. In this paper we seek to evaluate those two measures rigorously and for that we shall use random models and average case analysis.

A random 3CNF formula consists of m clauses chosen uniformly at random from the set of all $8\binom{n}{3}$ possible ones (m is a parameter of the distribution, and m/n is referred to as the clause-variable ratio, or the density, of the random instance). This distribution shows a sharp threshold with respect to satisfiability [19]. Specifically, a random 3CNF with clause-variable ratio below the threshold is satisfiable w. h. p. (with high probability, meaning with probability tending to 1 as n goes to infinity) and one with ratio above the threshold is unsatisfiable w. h. p. This threshold is not known exactly (and not even known to be independent of n), known bounds are at least 3.52 [21] and at most 4.506 [15]. Experimental results indicate that the threshold is closer to the higher end of the interval [14], and lies roughly around 4.2.

In this paper we study random 3CNF formulas with an arbitrary density. To this end, the aforementioned distribution needs to be refined, as otherwise, at some density, most formulas stop being satisfiable. One possibility, favored by many researchers in similar settings [18, 5, 22, 23, 4, 9] is to use the planted distribution, denoted throughout by $\mathsf{P}^{\text{plant}}_{n,p}$. A random 3CNF in this distribution is obtained by first picking an assignment φ to the variables, and then including every clause satisfied by φ with probability $p = p(n)$, thus guaranteeing that the resulting instance is satisfiable.

Our contribution is analyzing the performance of Warning Propagation (WP for brevity), a popular message passing algorithm, when applied to satisfiable formulas drawn from a certain random distribution over satisfiable 3CNF formulas, commonly called the planted distribution. We show that the standard way of running message passing algorithms—run message passing until convergence, simplify the formula according to the resulting assignment, and satisfy the remaining subformula, if possible, using a simple "off the shelf" heuristic—works for planted random satisfiable formulas with a sufficiently large yet constant clause-variable ratio. The effectiveness of message passing algorithms was *experimentally* shown for "hard" formulas [7], for which other heuristics fail; a decimation process based on Belief Propagation was analyzed in [12], where it was shown that the process *fails* to find a satisfying assignment when the input is a random formula with a certain clause-variable ratio (below the satisfiability threshold).

Our result is the first to rigorously prove the *effectiveness* of a message passing algorithm for the solution of a non-trivial random SAT distribution. Let us now move to a formal description of our results.

1.1 Warning propagation

WP is a simple iterative message passing algorithm, that serves as an excellent intuitive introduction to more involved message passing algorithms such as Belief Propagation [26] and Survey Propagation [7]. These algorithms are based on the *cavity method* in which the messages that a clause (or a variable) receives are meant to reflect a situation in which a "cavity" is formed, namely, the receiving clause (or variable) is no longer part of the formula. Messages in the WP algorithm can be interpreted as "warnings," telling a clause the values that variables will have if the clause "keeps quiet" and does not announce its wishes, and telling a variable which clauses will not be satisfied if the variable does not commit to satisfying them. We now present the algorithm in a formal way.

Let F be a CNF formula. For a variable x, let $N^+(x)$ be the set of clauses in F in which x appears positively (namely, as the literal x), and $N^-(x)$ be the set of clauses in which x appears negatively. For a clause C, let $N^+(C)$ be the set of variables that appear positively in C, and respectively $N^-(C)$ for negative ones. There are two types of messages involved in the WP algorithm. Messages sent from a **variable** x_i to a **clause** C_j in which it appears:

$$x_i \to C_j = \sum_{C_k \in N^+(x_i), k \neq j} C_k \to x_i - \sum_{C_k \in N^-(x_i), k \neq j} C_k \to x_i\,.$$

If x_i appears only in C_j then we set the message to 0. The intuitive interpretation of this message should be x_i signals C_j what is currently its favorable assignment by the other clauses it appears in (a positive message means TRUE, negative one means FALSE and a 0 message means $\mathsf{UNASSIGNED}$). The second type are messages sent from a **clause** C_j to a **variable** x_i appearing in C_j:

$$C_j \to x_i = \prod_{x_k \in N^+(C_j), k \neq i} I_{<0}(x_k \to C_j) \prod_{x_k \in N^-(C_j), k \neq i} I_{>0}(x_k \to C_j)\,,$$

where $I_{<0}(b)$ is an indicator function which is "1" iff $b < 0$ (respectively $I_{>0}$). If C_j contains only x_i (which cannot be the case in 3CNF formulas) then the message is set to 1. $C_j \to x_i = 1$ can be intuitively interpreted as C_j sending a *warning* to x_i asking it to commit to satisfying C_j (as all other literals signaled C_j that currently they evaluate to FALSE). Lastly, we define the current assignment of a variable x_i to be

$$B_i = \sum_{C_j \in N^+(x_i)} C_j \to x_i - \sum_{C_j \in N^-(x_i)} C_j \to x_i\,.$$

If $B_i > 0$ then x is assigned TRUE, if $B_i < 0$ then x_i is assigned FALSE, otherwise x_i is $\mathsf{UNASSIGNED}$. Assume some order on the clause-variable messages (e. g., the lexicographical order on pairs of the form (j,i) representing the message $C_j \to x_i$). Given a vector $\alpha \in \{0,1\}^{3m}$ in which every entry is the value of the corresponding $C_j \to x_i$ message, a partial assignment $\psi \in \{\mathsf{TRUE}, \mathsf{FALSE}, \mathsf{UNASSIGNED}\}^n$ can be generated according to the corresponding B_i values (as previously explained).

Given a 3CNF formula F on n variables and m clauses, the **factor graph** of F, denoted by $FG(\mathsf{F})$, is the following graph representation of F. The factor graph is a bipartite graph, $FG(\mathsf{F}) = (V_1 \cup V_2, E)$ where $V_1 = \{x_1, x_2, \ldots, x_n\}$ (the set of variables) and $V_2 = \{C_1, C_2, \ldots, C_m\}$ (the set of clauses). $(x_i, C_j) \in E$ iff x_i appears in C_j. For a 3CNF F with m clauses it holds that $\#E = 3m$, because every clause contains exactly 3 different variables. (Here and elsewhere, $\#A$ denotes the cardinality of a set A. The notation $|a|$ will denote the absolute value of a real number a.)

It would be convenient to think of the messages in terms of the corresponding factor graph. Every undirected edge (x_i, C_j) of the factor graph is replaced with 2 anti-parallel directed edges, $(x_i \to C_j)$ associated with the message $x_i \to C_j$ and respectively the edge $(C_j \to x_i)$.

Let us now formally describe the algorithm.

```
Warning Propagation(3CNF formula F):
1. construct the corresponding factor graph FG(F).
2. randomly initialize the clause-variable messages to 0 or 1.
3. repeat until no clause-variable message changed from the
```

```
   previous iteration:
   3.a randomly order the edges of FG(F).
   3.b update all clause-variable messages C_j → x_i according
       to the random edge order.
4. compute a partial assignment ψ according to the B_i messages.
5. return ψ.
```

The variable-clause message updates are implicit in line 3.b: when evaluating the clause-variable message along the edge $C \to x$, $C = (x \vee y \vee z)$, the variable-clause messages concerning this calculation ($z, y \to C$) are evaluated on-the-fly using the last updated values $C_i \to y$, $C_j \to z$ (allowing feedback from the same iteration). We allow the algorithm not to terminate (the clause-variable messages may keep changing every iteration). If the algorithm does return an assignment ψ then we say that it converged. In practice it is common to limit in advance the number of iterations, and if the algorithm does not converge by then, return a failure.

1.2 Our results

Given a 3CNF F, **simplify** F according to ψ, when ψ is a partial assignment, means: in every clause substitute every assigned variable with the value given to it by ψ. If a clause contains a literal which evaluates to true, remove the clause. From the remaining clauses, remove all literals which evaluate to false. The resulting instance is not necessarily in 3CNF form, as clauses may have any number of literals between 0 and 3. Denote by $\mathsf{F}|_\psi$ the 3CNF F simplified according to ψ. Note that $\mathsf{F}|_\psi$ may contain empty clauses, in which case it is not satisfiable. For a set of variables $A \subseteq V$, denote by $\mathsf{F}[A]$ the set of clauses in which all variables belong to A.

We call a 3CNF formula **simple**, if it can be satisfied using simple well-known heuristics (examples include very sparse random 3CNF formulas which are solvable w. h. p. using the pure-literal heuristic [8], formulas with small weight terminators—to use the terminology of [3]—solvable w. h. p. using RWalkSat, etc). This is a somewhat informal notion, but it suffices for our needs.

Theorem 1.1. *Let F be a 3CNF formula randomly sampled according to $\mathcal{P}^{\text{plant}}_{n,p}$, where $p \geq d/n^2$, d a sufficiently large constant. Then the following holds w. h. p. (the probability taken over the choice of F, and the random choices in lines 2 and 4 of the WP algorithm). There exists a satisfying assignment φ^* (not necessarily the planted one) such that:*

1. *$\mathsf{WP}(\mathsf{F})$ converges after at most $O(\log n)$ iterations.*
2. *Let ψ be the partial assignment returned by $\mathsf{WP}(\mathsf{F})$, let V_A denote the variables assigned to either TRUE or FALSE in ψ, and V_U the variables left UNASSIGNED. Then for every variable $x \in V_A$, $\psi(x) = \varphi^*(x)$. Moreover, $\#V_A \geq (1 - e^{-\Theta(d)})n$.*
3. *$\mathsf{F}|_\psi$ is a simple formula which can be satisfied in time $O(n)$.*

Remark 1.2. Theorem 1.1 relates to the planted 3SAT model, but [11] implies that it also true to the uniform random 3SAT distribution, in which a satisfiable formula with m clauses is chosen uniformly at random among all such formulas.

Proposition 1.3. *Let F be a 3CNF formula randomly sampled according to $P_{n,p}^{\text{plant}}$, where $p \geq c\log n/n^2$, with c a sufficiently large constant, and let φ be its planted assignment. Then w. h. p. after at most 2 iterations, $WP(F)$ converges, and the returned ψ equals φ.*

It is worth noting that formulas in $\mathsf{P}_{n,p}^{\text{plant}}$, with n^2p some large constant, are not known to be simple (in the sense that we alluded to above). For example, it is shown in [3] that RWalkSat is very unlikely to hit a satisfying assignment in polynomial time when running on a random $\mathsf{P}_{n,p}^{\text{plant}}$ instance in the setting of Theorem 1.1. Nevertheless, planted 3CNF formulas with sufficiently large (constant) density were shown to be solvable w. h. p. in [18] using a spectral algorithm. Though in our analysis we use similar techniques to [18] (which relies himself on [4]), our result is conceptually different in the following sense. In [4, 9, 18], the starting point is the planted distribution, and then one designs an algorithm that works well under this distribution. The algorithm may be designed in such a way that makes its analysis easier. In contrast, our starting point is a given algorithm, WP, and then we ask for which input distributions it works well. We cannot change the algorithm in ways that would simplify the analysis. Another difference between our work and that of [4, 9, 18] is that unlike the algorithms analyzed in those other papers, WP is a randomized algorithm, a fact which makes its analysis more difficult. We could have simplified our analysis had we changed WP to be deterministic (for example, by initializing all clause-variable messages to 1 in step 2 of the algorithm), but there are good reasons why WP is randomized. For example, it can be shown that (the randomized version) WP converges with probability 1 on 2CNF formulas that form one cycle of implications, but might not converge if step 4 does not introduce fresh randomness in every iteration of the algorithm (details omitted).

The planted 3SAT model is also similar to LDPC codes in many ways. Both constructions are based on random factor graphs. In codes, the received corrupted codeword provides noisy information on a single bit or on the parity of a small number of bits of the original codeword. In $\mathsf{P}_{n,p}^{\text{plant}}$, φ being the planted assignment, the clauses containing a variable x_i provide noisy information on the polarity of $\varphi(x_i)$. Comparing our results with the coding setting, the effectiveness of message passing algorithms for amplifying local information in order to decode codes close to channel capacity was recently established in a number of papers, e. g., [25, 27]. Our results are similar in flavor, however the combinatorial analysis provided here allows to recover an assignment satisfying *all* clauses, whereas in the random LDPC codes setting, message passing allows to recover only $1-o(1)$ fraction of the codeword correctly. In [24] it is shown that for the erasure channel, all bits may be recovered correctly using a message passing algorithm, however in this case the LDPC code is designed so that message passing works for it.

1.3 Paper's organization

The remainder of the paper is structured as follows. Section 2 provides an overview that may help the reader follow the more technical parts of the proofs. In Section 3 we discuss some properties that a typical instance in $\mathsf{P}_{n,p}^{\text{plant}}$ possesses. Using these properties, we prove in Section 4 Theorem 1.1 and Proposition 1.3. In Section 6 we summarize our results and discuss potentially interesting lines for further research.

2 Proof outline

Let us first consider some possible fixed points of the Warning Propagation (WP) algorithm. The *trivial* fixed point is the one in which all messages are **0**. One may verify that this is the unique fixed point in some cases when the underlying 3CNF formula is very easy to satisfy, such as when all variables appear only positively, or when every clause contains at least two variables that do not appear in any other clause. A *local maximum* fixed point is one that corresponds to a strict local maximum of the underlying MAX-3SAT instance, namely to an assignment τ to the variables in which flipping the truth assignment of any single variable causes the number of satisfied clauses to strictly decrease. The reader may verify that if every clause C sends a **1** message to a variable if no other variable satisfies C under τ, and a **0** message otherwise, then this is indeed a fixed point of the WP algorithm. Needless to say, the WP algorithm may have other fixed points, and might not converge to a fixed point at all.

Recall the definition of $\mathsf{P}_{n,p}^{\text{plant}}$. First a truth assignment φ to the variables $V = \{x_1, x_2, \ldots, x_n\}$ is picked uniformly at random. Next, every clause satisfied by φ is included in the formula with probability p (in our case $p \geq d/n^2$, d a sufficiently large constant). There are $(2^3-1)\cdot\binom{n}{3}$ clauses satisfied by φ, hence the expected size of F is $p\cdot 7\cdot\binom{n}{3} = 7dn/6 + o(n)$ (when d is constant, then this is linear in n, and therefore such instances are sometimes referred to as *sparse* 3CNF formulas). To simplify the presentation, we may assume due to symmetry that the planted assignment φ is the all-one vector.

To aid intuition, we list some (incorrect) assumptions and analyze the performance of WP on a $\mathsf{P}_{n,p}^{\text{plant}}$ instance under these assumptions.

1. In expectation, a variable appears in $4\binom{n}{2}p = 2d + o(1)$ clauses positively, and in $3d/2 + o(1)$ clauses negatively. Our first assumption is that for every variable, its number of positive and negative appearances is equal to these expectations.

2. We say that a variable *supports* a clause with respect to the planted assignment (which was assumed without loss of generality to be the all **1** assignment) if it appears positively in the clause, and the other variables in the clause appear negatively. Hence the variable is the only one to satisfy the clause under the planted assignment. For every variable in expectation there are roughly $d/2$ clauses that it supports. Our second assumption is that for every variable, the number of clauses that it supports is equal to this expectation.

3. Recall that in the initialization of the WP algorithm, every clause-variable message $C \to x$ is 1 w. p. $1/2$, and 0 otherwise. Our third assumption is that with respect to every variable, half the messages that it receives from clauses in which it is positive are initialized to 1, and half the messages that it receives from clauses in which it is negative are initialized to 1.

4. Recall that in step 3.b of WP, clause-variable messages are updated in a random order. Our fourth assumption is that in each iteration of step 3, the updates are based on the values of the other messages from the previous iteration, rather than on the last updated values of the messages (that may correspond either to the previous iteration or the current iteration, depending on the order in which clause-variable messages are visited). Put differently, we assume that in step 3.b all clause-variable messages are evaluated in *parallel*.

Observe that under the first two assumptions, the planted assignment is a local maximum of the underlying MAX-3SAT instance. We show that under the third and fourth assumption, WP converges to the corresponding local maximum fixed point in two iterations. Based on the initial messages as in our third assumption, the messages that variables send to clauses are all roughly $(2d-3d/2)/2 = d/4$. Following the initialization, in the first iteration of step 3 every clause C that x supports will send x the message 1, and all other messages will be 0. Here we used our fourth assumption. (Without our fourth assumption, WP may run into trouble as follows. The random ordering of the edges in step 3 may place for some variable x all messages from clauses in which it appears positively before those messages from clauses in which it appears negatively. During the iteration, some of the messages from the positive clauses may change from 1 to 0. Without our fourth assumption, this may at some point cause x to signal to some clauses a negative rather than positive value.) The set of clause-variable messages as above will become a fixed point and repeat itself in the second iteration of step 3. (For the second iteration, the fourth assumption is no longer needed.) Hence the algorithm will terminate after the second iteration.

Unfortunately, none of the four assumptions that we made are correct. Let us first see to what extent they are violated in the context of Proposition 1.3, namely, when d is very large, significantly above $\log n$. Standard concentration results for independent random variables then imply that the first, second and third assumptions simultaneously hold for all variables, up to small error terms that do not affect the analysis. Our fourth assumption is of course never true, simply because we defined WP differently. This complicates the analysis to some extent and makes the outcome depend on the order chosen in the first iteration of step 3.a of the algorithm. However, it can be shown that for most such orders, the algorithm indeed converges to the fixed point that corresponds to the planted assignment.

The more difficult part of our work is the case when d is constant (though a sufficiently large constant), as in the case of Theorem 1.1. In this case, already our first two assumptions are incorrect. Random fluctuations with respect to expected values will w. h. p. cause a linear fraction of the variables to appear negatively more often than positively, or not to support any clause (with respect to the planted assignment). In particular, the planted assignment would no longer be a local maximum with respect to the underlying MAX-3SAT instance. Nevertheless, as is known from previous work [18], a large fraction of the variables will behave sufficiently close to expectation so that the planted assignment is a local maximum with respect to these variables. Slightly abusing notation, these set of variables are often called the *core* of the 3CNF formula. Our proof plan is to show that WP does converge, and that the partial assignment in step 4 assigns all core variables their correct planted value. Moreover, for non-core variables, we wish to show that the partial assignment does not make any unrecoverable error—whatever value it assigns to some of them, it is always possible to assign values to those variables that are left unassigned by the partial assignment so that the input formula is satisfied. The reason why we can expect such a proof plan to succeed is that it is known to work if one obtains an initial partial assignment by means other than WP, as was already done in [18, 17].

Let us turn now to our third assumption. It too is violated for a linear fraction of the variables, but is nearly satisfied for most variables. This fact marks one point of departure for our work compared to previous work [18, 17]. Our definition of the core variables will no longer depend only on the input formula, but also on the random choice of initialization messages. This adds some technical complexity to our proofs.

The violation of the fourth assumption is perhaps the technical part in which our work is most interesting. It relates to the analysis of WP on factor graphs that contain cycles, which is often a stumbling point when one analyzes message passing algorithms. Recall that when d is very large (Proposition 1.3), making the fourth assumption simplifies the proof of convergence of WP. Hence removing this assumption in that case becomes a nuisance. On the other hand, when d is smaller (as in Theorem 1.1), removing this assumption becomes a necessity. This will become apparent when we analyze convergence of WP on what we call *free cycles*. If messages in step 3.b of WP are updated based on the value of other messages in the *previous* iteration (as in our fourth assumption), then the random choice of order in step 3.a of WP does not matter, and one can design examples in which the messages in a free cycle never converge. In contrast, if messages in step 3.b of WP are updated based on the latest value of other messages (either from the previous iteration or from the current iteration, whichever one is applicable), free cycles converge with probability 1 (as we shall later show).

To complete the proof plan, we still need to show that simplifying the input formula according to the partial assignment returned by WP results in a formula that is satisfiable, and moreover, that a satisfying assignment for this sub-formula can easily be found. The existential part (the sub-formula being satisfiable) will follow from a careful analysis of the partial assignment returned by WP. The algorithmic part (easily finding an assignment that satisfies the sub-formula) is based on the same principles used in [4, 18], showing that the sub-formula breaks into small connected components.

3 Properties of a random $\mathsf{P}_{n,p}^{\text{plant}}$ instance

In this section we discuss relevant properties of a random $\mathsf{P}_{n,p}^{\text{plant}}$ instance. This section is rather technical in nature. The proofs are based on probabilistic arguments that are standard in our context. In the rest of the paper, for simplicity of presentation, we assume due to symmetry that the planted assignment is the all TRUE assignment.

3.1 Preliminaries

The following well-known concentration result (see, for example [6, Corollary A.1.14]) will be used several times in the proof. We denote by $B(n,p)$ the binomial random variable with parameters n and p.

Theorem 3.1 (Chernoff's inequality)**.** *If $X \sim B(n,p)$ and $\varepsilon > 0$, then*

$$\Pr\left(|X - \mathbb{E}[X]| \geq \varepsilon\mathbb{E}[X]\right) \leq e^{-\Omega(\varepsilon^3\mathbb{E}[X])}.$$

We shall also use the following version of the Martingale argument, which can be found in [6, Page 101]. Assume the probability space is generated by a finite set of mutually independent Yes/No choices, indexed by $i \in I$. Given a random variable Y on this space, let p_i denote the probability that choice i is Yes. Let c_i be such that changing choice i (keeping all else the same) can change Y by at most $c_i \leq c$. We call $p_i(1-p_i)c^2$ the variance of choice i. Now consider a solitaire game in which one finds the value of Y by making queries of an always truthful oracle. The choice of query can depend on previous responses. All possible questioning lines can be naturally represented in a decision tree form. A "line of questioning" is a path from the root to a leaf of this tree, a sequence of questions and responses that determine Y. The total variance of a line of questioning is the sum of the variances of the queries in it.

Theorem 3.2. *For every $\varepsilon > 0$ there exists a $\delta > 0$ so that if for every line of questioning that determines Y, the total variance is at most σ^2, then*

$$\Pr[|Y - \mathbb{E}[Y]| > \alpha\sigma] \leq 2e^{\alpha^2/(2(1+\varepsilon))},$$

for all positive α with $\alpha C < \sigma(1+\varepsilon)\delta$.

3.2 Stable variables

Definition 3.3. A variable x *supports* a clause C with respect to a partial assignment ψ, if it is the only variable to satisfy C under ψ, and the other two variables are assigned by ψ.

Proposition 3.4. *Let F be a 3CNF formula randomly sampled according to $\mathcal{P}^{\text{plant}}_{n,p}$, where $p \geq d/n^2$, d a sufficiently large constant. Let F_{SUPP} be a random variable counting the number of variables in F whose support w.r.t. φ is less than $d/3$. Then w. h. p. $F_{\text{SUPP}} \leq e^{-\Theta(d)}n$.*

Proof. Every variable is expected to support $(d/n^2) \cdot \binom{n}{2} = d/2 + O(1/n)$ clauses, thus using Chernoff's inequality the probability of a single variable supporting less than $d/3$ clauses is at most $e^{-\Theta(d)}$. Using linearity of expectation, $\mathbb{E}[F_{\text{SUPP}}] \leq e^{-\Theta(d)}n$. To prove concentration around the expected value we use Chernoff's bound once more as the support of one variable is independent of the others (since it concerns different clauses which are included independently of each other). The claim then follows. □

Following the definitions in Section 1.1, given a CNF F and a variable x, we let $N^{++}(x)$ be the set of clauses in F in which x appears positively but does not support w. r. t. φ. Let $N^s(x)$ be the set of clause in F which x supports w. r. t. φ. Let $\pi = \pi(\mathsf{F})$ be some ordering of the clause-variable message edges in the factor graph of F. For an index i and a literal ℓ_x (by ℓ_x we denote a literal over the variable x) let $\pi^{-i}(\ell_x)$ be the set of clause-variable edges $(C \to x)$ that appear before index i in the order π and in which x appears in C as ℓ_x. For a set of clause-variable edges E and a set of clauses $\mathcal{C}$ we denote by $\mathsf{E} \cap \mathsf{C}$ the subset of edges containing a clause from C as one endpoint.

Definition 3.5. Let π be an ordering of the clause-variable messages of a formula F. A variable x is *stable* in F w. r. t. π if for every location i in π that contains a message $C \to x$, $C = (\ell_x \vee \ell_y \vee \ell_z)$, the following holds:

1. $|\#\pi^{-i}(y) \cap N^{++}(y) - \#\pi^{-i}(\bar{y}) \cap N^{-}(y)| \leq d/30$,
2. $|\#N^{++}(y) - \#N^{-}(y)| \leq d/30$,
3. $\#N^s(y) \geq d/3$,

and the same holds for z.

Proposition 3.6. *Let F be a 3CNF formula randomly sampled according to $\mathcal{P}^{\text{plant}}_{n,p}$, where $p \geq d/n^2$, d a sufficiently large constant. Let π be a random ordering of the clause-variable messages, and F_{UNSTAB} be a random variable counting the number of variables in F which are not stable. Then w. h. p. $F_{\text{UNSTAB}} \leq e^{-\Theta(d)}n$.*

Proof. We start by bounding $\mathbb{E}[F_{\text{UNSTAB}}]$. Observe that given that y appears in a clause C, and does not support it, the probability of y appearing positively or negatively is equal. Hence, for a message $C \to x$ in location i in π, $C = (\ell_x \vee \ell_y \vee \ell_z)$, we have

$$\mathbb{E}[|\#\pi^{-i}(y) \cap N^{++}(y) - \#\pi^{-i}(\bar{y}) \cap N^{-}(y)|] = 0\,,$$

and the same for z. If however

$$|\#\pi^{-i}(y) \cap N^{++}(y) - \#\pi^{-i}(\bar{y}) \cap N^{-}(y)| > d/30$$

then at least one of the quantities deviates from its expectation by $d/60$.

Look at $\#\pi^{-i}(y) \cap N^{++}(y)$—this is the number of successes in draws without replacement. It is known that this quantity is more concentrated than the corresponding quantity if the draws were made with replacement [20]. In particular, since the expectation of $\#\pi^{-i}(y) \cap N^{++}(y)$ is $O(d)$ it follows from Chernoff's bound that the probability that it deviates from its expectation by more than $d/60$ is $e^{-\Theta(d)}$. A similar statement holds for $\#\pi^{-i}(\bar{y}) \cap N^{-}(y)$. Properties (2) and (3) are bounded similarly using concentration results.

The calculations above hold in particular for the first $5d$ appearances of messages involving x. As for message $5d+1$, the probability of this message causing x to become unstable is bounded by the event that x appears in more than $5d$ clauses. As x is expected to appear in $3.5d$ clauses, the latter event happens w.p. $e^{-\Theta(d)}$ (again using Chernoff's bound). To sum up,

$$\Pr[x \text{ is unstable }] \leq 5d \cdot e^{-\Theta(d)} + e^{-\Theta(d)} = e^{-\Theta(d)}\,.$$

The bound on $\mathbb{E}[F_{\text{UNSTAB}}]$ follows by linearity of expectation.

We are now left with proving that F_{UNSTAB} is concentrated around its expectation, we do so using a martingale argument. Define two new random variables, F_1 counting the number of unstable variables x s. t. there exists a clause C, containing x, and another variable y, such that y appears in more than $\log n$ clauses, and F_2 to be the unstable variables such that in all clauses in which they appear, all the other variables appear in at most $\log n$ clauses. Observe that $F_{\text{UNSTAB}} = F_1 + F_2$. To bound F_1, observe that if $F_1 \geq 1$, then in particular this implies that there exists a variable which appears in more than $\log n$ clauses in F. This happens with probability $o(1)$ since every variable is expected to appear only in $O(d)$ clauses (using Chernoff's bound). To bound F_2 we use a martingale argument in the constellation of [6], page 101. We use the clause-exposure martingale (the clause-exposure martingale implicitly includes the random ordering π, since one can think of the following way to generate the random instance—first randomly shuffle all possible clauses, and then toss the coins). The exposure of a new clause C can change F_2 by at most $6\log n$ since every variable in C appears in at most $\log n$ clauses, namely with at most $2\log n$ other variables that might become (un)stable due to the new clause. The martingale's total variance, in the terminology of Theorem 3.2, is $\sigma^2 = \Theta(dn\log^2 n)$. Theorem 3.2, with $\alpha = e^{-\Theta(d)}\sqrt{n}/\log n$, and the fact that $\mathbb{E}[F_2] \leq \mathbb{E}[F_{\text{UNSTAB}}]$, implies concentration around the expectation of F_2. $\square$

Let $\alpha \in \{0,1\}^{3\#\mathsf{F}}$ be a clause-variable message vector. For a set of clause-variable message edges E let $\mathbf{1}_\alpha(\mathcal{E})$ be the set of edges along which the value is 1 according to α. For a set of clauses C, $\mathbf{1}_\alpha(\mathsf{C})$ denotes the set of clause-variable message edges in the factor graph of F containing a clause from C as one endpoint and along which the value is 1 in α.

Definition 3.7. A variable x is *violated* by α in π if there exists a message $C \to x$, $C = (\ell_x \vee \ell_y \vee \ell_z)$, in place i in π such that one of the following holds:

1. $|\#\mathbf{1}_\alpha(\pi^{-i}(y) \cap N^{++}(y)) - \#\mathbf{1}_\alpha(\pi^{-i}(\bar{y}) \cap N^{-}(y))| > d/30$
2. $|\#\mathbf{1}_\alpha(N^{++}(y)) - \#\mathbf{1}_\alpha(N^{-}(y))| > d/30$
3. $\#\mathbf{1}_\alpha(N^{s}(y)) < d/7$.

Or one of the above holds for z.

Proposition 3.8. *Let F be a 3CNF formula randomly sampled according to $\mathcal{P}^{\text{plant}}_{n,p}$, where $p \geq d/n^2$, d a sufficiently large constant. Let X be the set of stable variables w. r. t. an arbitrary ordering π. Let α be a random clause-variable message vector. Let F_{VIO} be a random variable counting the number of violated variables in X. Then w. h. p. $F_{\text{VIO}} \leq e^{-\Theta(d)}n$.*

Proof. If $\#X \leq e^{-\Theta(d)}n$ then we are done, as $F_{\text{VIO}} \leq \#X$. Therefore, assume $\#X = e^{-\Theta(d)}n$. As in the proof of Proposition 3.6, we first bound $\mathbb{E}[F_{\text{VIO}}]$, and then prove concentration using a martingale argument. Consider a stable variable x in F w. r. t. to an ordering π of the clause-variable messages. Let α be a random assignment to the clause-variable messages. Consider a clause-variable message edge $C \to x$ at location i in π. x is stable and therefore

$$|\#\pi^{-i}(y) \cap N^{++}(y) - \#\pi^{-i}(y) \cap N^{-}(y)| \leq d/30\,.$$

Since α is a random assignment

$$\mathbb{E}[|\#\mathbf{1}_\alpha(\pi^{-i}(y) \cap N^{++}(y)) - \#\mathbf{1}_\alpha(\pi^{-i}(\bar{y}) \cap N^{-}(y))|] \leq d/60\,.$$

If however

$$|\#\mathbf{1}_\alpha(\pi^{-i}(y) \cap N^{++}(y)) - \#\mathbf{1}_\alpha(\pi^{-i}(\bar{y}) \cap N^{-}(y))| > d/30\,, \tag{3.1}$$

then at least one of the quantities in (3.1) deviated from its expectation by at least $(d/30 - d/60)/2 = d/120$. Since both quantities are binomially distributed with expectation $O(d)$, the probability of the latter happening is $e^{-\Theta(d)}$, using standard concentration results.

Properties (2) and (3) are bounded similarly using the Chernoff bound. Using the union bound as in the proof of Proposition 3.6 and the linearity of expectation, one obtains that $\mathbb{E}[F_{\text{VIO}}] \leq e^{-\Theta(d)}\#X = e^{-\Theta(d)}n$. Finally, repeat the martingale argument in Proposition 3.6 (instead of a clause-exposure martingale, we have a clause-variable message values exposure martingale). Both martingales involve $\Theta(n)$ random variables, and therefore the same exact calculation holds in the case of the current proposition as well (maybe with different constants, which are accounted for in the Θ-notation). □

3.3 Dense subformulas

The next property we discuss is analogous to a property proved in [4] for random graphs. Loosely speaking, a random graph typically does not contain a small induced subgraph with a large average degree. A similar proposition for 3SAT can also be found in [18].

Proposition 3.9. *Let $c > 1$ be an arbitrary constant. Let F be a 3CNF formula randomly sampled according to $\mathcal{P}^{\text{plant}}_{n,p}$, where $p \geq d/n^2$, d a sufficiently large constant. Then w. h. p. there exists* no *subset of variables U, such that $\#U \leq e^{-\Theta(d)}n$ and there are at least $c\#U$ clauses in F containing two variables from U.*

Proof. For a fixed set U of variables, $\#U = k$, the number of clauses containing two variables from U is

$$\binom{k}{2}(n-2)2^3 \leq 4k^2 n\,.$$

Each of these clauses is included independently w. p. d/n^2. Thus, the probability that ck of them are included is at most

$$\binom{4k^2 n}{ck}\left(\frac{d}{n^2}\right)^{ck} \leq \left(\frac{4k^2 ne}{ck}\cdot\frac{d}{n^2}\right)^{ck} \leq \left(\frac{12kd}{cn}\right)^{ck}.$$

Using the union bound, the probability there exists a "dense" set U is at most

$$\sum_{k=2}^{e^{-\Theta(d)}n}\binom{n}{k}\left(\frac{12kd}{cn}\right)^{ck} = O(d^{2c}/n^{2c-2})\,.$$

The last equality is obtained using standard calculations, and the standard estimate on the binomial coefficient: $\binom{n}{k} \leq (en/k)^k$. □

3.4 The core variables

We describe a subset of the variables, denoted throughout by H and referred to as the *core variables*, which plays a crucial role in the analysis. The notion of a stable variable is not enough to ensure that the algorithm will set a stable variable according to the planted assignment, as it may happen that a stable variable x appears in many of its clauses with unstable variables. Thus, x can be biased in the wrong direction (by wrong we mean disagreeing with the planted assignment). However, if most of the clauses in which x appears contain only stable variables, then this is already a sufficient condition to ensure that x will be set correctly by the algorithm. The set H captures the notion of such variables. There are several ways to define a set of variables with these desired properties, we present one of them, and give a constructive way of obtaining it (though it has no algorithmic implications, at least not in our context).

Formally, $\mathsf{H} = \mathsf{H}(\mathsf{F}, \varphi, \alpha, \pi)$ is constructed using the following iterative procedure:

Proposition 3.10. *If both α and π are chosen uniformly at random then with probability $1 - n^{-\Omega(d)}$, $\#H = (1 - e^{-\Omega(d)})n$.*

Proof. Let $\bar{\mathsf{H}} = V \setminus \mathsf{H}$. Set $\delta = e^{-\Theta(d)}$. Partition the variables in $\bar{\mathsf{H}}$ into variables that belong to $A_1 \cup A_2 \cup A_3$, and variables that were removed in the iterative step, $\bar{H}^{it} = H_0 \setminus \mathsf{H}$. If $\#\bar{\mathsf{H}} \geq \delta n$, then at least one of $A_1 \cup A_2 \cup A_3$, $\bar{H}^{it}$ has cardinality at least $\delta n/2$. Consequently,

$$\Pr[\#\bar{\mathsf{H}} \geq \delta n] \leq \underbrace{\Pr[\#A_1 \cup A_2 \cup A_3 \geq \delta n/2]}_{(a)} + \underbrace{\Pr[\#\bar{H}^{it} \geq \delta n/2 \;\big|\; \#A_1 \cup A_2 \cup A_3 \leq \delta n/2]}_{(b)}.$$

Let A_1 be the set of variables whose support w. r. t. φ is at most $d/3$.
Let A_2 be the set of non-stable variables w. r. t. π.
Let A_3 be the set of stable variables w. r. t. π which are violated by α.

1. *Set $H_0 = V \setminus (A_1 \cup A_2 \cup A_3)$.*
2. *While there exists a variable $a_i \in H_i$ which supports less than $d/4$ clauses in $\mathsf{F}[H_i]$ OR appears in more than $d/30$ clauses* not *in $\mathsf{F}[H_i]$ define $H_{i+1} = H_i \setminus \{a_i\}$.*
3. *Let a_m be the last variable removed in step 2. Define $\mathsf{H} = H_{m+1}$.*

Propositions 3.4, 3.6, and 3.8 are used to bound (a). To bound (b), observe that every variable that is removed in iteration i of the iterative step (step 2), supports at least $(d/3 - d/4) = d/12$ clauses in which at least another variable belongs to $\{a_1, a_2, \ldots, a_{i-1}\} \cup A_1 \cup A_2 \cup A_3$, or appears in $d/30$ clauses each containing at least one of the latter variables. Consider iteration $\delta n/2$. Assuming $\# A_1 \cup A_2 \cup A_3 \leq \delta n/2$, by the end of this iteration there exists a set containing at most δn variables, and there are at least $d/30 \cdot \delta n/2 \cdot 1/3$ clauses containing at least two variables from it (we divide by 3 as every clause might have been counted 3 times). Plugging $c = d/180$ in Proposition 3.9, (b) is bounded. Finally observe that (a) occurs with exponentially small probability, and (b) occurs with probability $n^{-\Omega(d)}$. □

3.5 The factor graph of the non-core variables

Proposition 3.10 implies that for $p = c \log n/n^2$, c a sufficiently large constant, w. h. p. H contains already all variables. Therefore the following propositions are relevant for the setting of Theorem 1.1 (namely, $p = O(1/n^2)$). In what follows we establish the typical structure of the factor graph induced on the non-core variables.

Proposition 3.11. *Let F be a 3CNF formula randomly sampled according to $\mathsf{P}^{\text{plant}}_{n,p}$, where $p \geq d/n^2$, d a sufficiently large constant, let π be the initial random ordering of the clause-variable messages, and α the initial random clause-variable message vector. Let T be the factor graph induced on the non-core variables. T enjoys the following properties:*

1. *Every connected component contains w. h. p. $O(\log n)$ variables,*
2. *every connected component contains w. h. p. at most one cycle,*
3. *the probability of a cycle of length at least k in T is at most $e^{-\Theta(dk)}$.*

A proposition of similar flavor to (1) was proven in [18] though with respect to a different notion of core. This alone would not have sufficed to prove our result, and we need (2) and (3) as well.

Corollary 3.12. *Let $f = f(n)$ be such that $f(n) \to \infty$ as $n \to \infty$. Then w. h. p. there is no cycle of length $f(n)$ in the non-core factor graph.*

The proof of Proposition 3.11 is quite long and technical. To avoid distraction, we go ahead and prove Theorem 1.1 and Proposition 1.3, and defer the proof of Proposition 3.11 to Section 5.

4 Proof of Theorem 1.1 and Proposition 1.3

We start by giving an outline of the proof of Theorem 1.1. Proposition 1.3 is derived as an easy corollary of that proof. To prove Theorem 1.1 we need to establish three properties:

1. *Convergence*: the WP algorithm converges to a fixed point.
2. *Consistency*: the partial assignment implied by this fixed point is consistent with some satisfying assignment.
3. *Simplicity*: after simplifying the input formula by substituting in the values of the assigned variables, the remaining subformula is not only satisfiable (this is handled by consistency), but also simple.

We assume that the formula F and the execution of WP are *typical* in the sense that Propositions 3.10 and 3.11 hold. First we prove that after one iteration WP sets the core variables H correctly (B_i agrees with φ in sign) and this assignment does not change in later iterations. The proof of this property is rather straightforward from the definition of a core. This establishes convergence and consistency for the core variables. From iteration 2 onwards WP is basically running on F in which variables belonging to H are substituted with their planted assignment. This subformula is satisfiable. Moreover, its factor graph contains small (logarithmic size) connected components, each containing at most one cycle. This last fact serves a dual purpose. It shows that if WP will eventually converge, the simplicity property will necessarily hold. Moreover, it will assist us in proving convergence and consistency for the subformula. Consider a connected component composed of a cycle and trees "hanging" on the cycle. Proving convergence on the trees is done using a standard inductive argument. The more interesting part is proving convergence on the cycle. The difficulty there is that messages on a cycle may have more than one fixed point to which they may possibly converge, which makes it more difficult to prove that they converge at all. Our proof starts with a case analysis that identifies those cases that have multiple fixed points. For these cases we prove that almost surely random fluctuations caused by step 3.a of the WP algorithm will lead to convergence to some fixed point. This is similar in flavor to the fact that a random-walk on a line eventually reaches an endpoint of the line (even though one cannot tell a-priori which endpoint this will be). Hand-in-hand with establishing convergence for the trees and cycle, we shall also prove consistency.

The set V_A of Theorem 1.1 is composed of all variables from H and those variables from the non-core factor graph that get assigned. The set V_U is composed of the UNASSIGNED variables from non-core factor graph. We now proceed with the formal proof.

4.1 Analysis of WP on the core factor graph

We start by proving that the messages concerning the factor graph induced by the core-variables converge to the correct value, and remain the same until the end of the execution.

We say that a message $C \to x$, $C = (\ell_x \vee \ell_y \vee \ell_z)$, is *correct* if its value is the same as it is when $y \to C$ and $z \to C$ are 1 (that is agree in sign with their planted assignment). In other words, $C \to x$ is 1 iff $C = (x \vee \bar{y} \vee \bar{z})$ (x supports C).

Proposition 4.1. *If $x_i \in H$ and all messages $C \to x_i$, $C \in F[H]$ are correct at the beginning of an iteration (line 3 in the WP algorithm), then this invariant is kept by the end of that iteration.*

Proof. By contradiction, let $C_0 \to x$ be the first wrongly evaluated message in the iteration, $C_0 = (\ell_x \vee \ell_y \vee \ell_z)$. Then at least one of y, z sent a wrong message to C_0.

$$y \to C_0 = \sum_{C \in N^+(y), C \neq C_0} C \to y - \sum_{C' \in N^-(y), C' \neq C_0} C' \to y\,.$$

Every message $C'' \to y$, $C'' \in F[\mathsf{H}] \cap \{N^{++}(y) \cup N^-(y)\}$ is 0 (since it was correct at the beginning of the iteration and that didn't change until evaluating $C_0 \to x$). On the other hand, $y \in \mathsf{H}$ and therefore it supports at least $d/4$ clauses in $\mathsf{F}[\mathsf{H}]$. Thus at least $(d/4-1)$ messages in the left hand sum are "1" (we subtract 1 as y might support C_0). y appears in at most $d/30$ clauses with non-core variables (all of which may contribute a wrong "1" message to the right hand sum). All in all, $y \to C_0 \geq (d/4 - d/30 - 1) > d/5$, which is correct (recall, we assume $\varphi = \mathbf{1}^n$). The same applies for z, contradicting our assumption. □

Proposition 4.2. *If $x_i \in \mathsf{H}$ and all messages $C \to x_i$, $C \in \mathsf{F}[\mathsf{H}]$ are correct by the end of a WP iteration, then B_i agrees in sign with $\varphi(x_i)$ by the end of that iteration.*

Proposition 4.2 follows immediately from the definition of H and the message B_i. It suffices to show then that after the first iteration all messages $C \to x_i$, $C \in \mathsf{F}[\mathsf{H}]$ are correct.

Proposition 4.3. *If F is a typical instance in the setting of Theorem 1.1, then after one iteration of WP(F), for every variable $x_i \in \mathsf{H}$, every message $C \to x_i$, $C \in \mathsf{F}[\mathsf{H}]$ is correct.*

Proof. The proof is by induction on the order of the execution in the first iteration. Consider the first message $C \to x$, $C = (\ell_x \vee \ell_y \vee \ell_z)$, $C \in \mathsf{F}[\mathsf{H}]$, to be evaluated in the first iteration. Now consider the message $y \to C$ at the time $C \to x$ is evaluated. All messages $C' \to y$, $C' \in \mathsf{F}[H]$ have their initial random value (as $C \to x$ is the first core message to be evaluated). Furthermore, $y \in \mathsf{H}$, and therefore there are at most $d/30$ messages of the form $C'' \to y$, $C'' \notin \mathsf{F}[\mathsf{H}]$. $x \in \mathsf{H}$ hence it is stable w. r. t. π and not violated by the initial clause-variable random messages. Therefore

$$y \to C \quad \geq \quad \underbrace{d/7}_{\text{property (3) in Definition 3.7}} \quad - \quad \underbrace{d/30}_{\text{property (2) in Definition 3.7}} \quad - \quad \underbrace{d/30}_{\text{non-core messages}} \quad > \quad d/14\,.$$

The same applies to z, to show that $C \to x$ is correct. Now consider a message $C \to x$ at position i, and assume all core messages up to this point were evaluated correctly. Observe that every core message $C' \to y$ that was evaluated already, if $C' \in \{N^{++}(y) \cup N^-(y)\} \cap \mathsf{F}[\mathsf{H}]$ then its value is "0" by the induction hypothesis. Since x is not violated by α, property (2) in Definition 3.7 ensures that to begin with $|\#\mathbf{1}_\alpha(N^{++}(y)) - \#\mathbf{1}_\alpha(N^-(y))| \leq d/30$. $y \in \mathsf{H}$, therefore it appears in at most $d/30$ non-core messages, all of which could have been already wrongly evaluated, changing the above difference by additional $d/30$. As for the core messages of y which were already evaluated, since they were evaluated correctly, property (1) in Definition 3.7 ensures that the above difference changes by at most additional $d/30$. All in all, by the time we evaluate $C \to x$,

$$\sum_{C' \in N^{++}(y), C' \neq C} C' \to y - \sum_{C'' \in N^-(y), C'' \neq C} C'' \to y \geq -3 \cdot d/30\,.$$

As for messages that y supports, property (3) in Definition 3.7 ensures that their contribution is at least $d/7$ to begin with. Every core message in $N^s(y)$ that was evaluated turned to "1," every non-core message was already counted in the above difference. Therefore $y \to C \geq d/7 - 3 \cdot d/30 > d/25$. The same applies to z showing that $C \to x$ is correct. ☐

To prove Proposition 1.3, observe that when $p = c \log n/n^2$, with c a sufficiently large constant, Proposition 3.10 implies $\mathsf{H} = V$. Combining this with Proposition 4.3, Proposition 1.3 readily follows.

4.2 The effect of messages that already converged

It now remains to analyze the behavior of WP on the non-core factor graph, given that the messages involving the core factor graph have converged correctly. A key observation is that once the messages in the factor graph induced by the core variables converged, we can think of WP as if running on the formula resulting from replacing every core variable with its planted assignment and simplifying (which may result in a 1-2-3CNF). The observation is made formal by the following proposition:

Proposition 4.4. *Consider a run of WP that has converged on the core. Starting at some iteration after WP has converged on the core, consider two alternative continuations of the warning propagation algorithm. WP_1 denotes continuing with WP on the original input formula. WP_2 denotes continuing with WP on the formula obtained by replacing each core variable with its planted assignment and simplifying. Then for every iteration t, the sequence of messages in the t'th iteration of WP_2 is identical to the respective subsequence in WP_1. (This subsequence includes those messages not involving the core variables, and includes messages of type $x \to C$ and of the type $C \to x$.)*

Proof. First note that all messages $x \to C$, $x \in \mathsf{H}$, do not change (sign) from the second iteration onwards (by the analysis in the proof of Proposition 4.3). Furthermore, if ℓ_x satisfies C in φ, then $x \to C$ is positive (if x is a true literal in C, or negative otherwise), and therefore all messages $C \to y$, $y \neq x$ are constantly 0. Namely, they don't affect any calculation, and this is as if we replaced ℓ_x with TRUE, and in the simplification process C disappeared. If ℓ_x is false in C under φ, then $x \to C$ is constantly negative (if $\ell_x = x$, or constantly positive if $\ell_x = \bar{x}$), and this is exactly like having ℓ_x removed from C (which is the result of the simplification process).

☐

4.3 Analysis of WP on the *non*-core factor graph

Note that to prove the convergence of the algorithm we need also to prove that messages of the sort $C \to x$ where C is not in the core and x is in the core converge. However, if we prove that all messages in the factor graph induced by the non-core variables converge, then this (with the fact that the core factor graph messages converge) immediately implies the convergence of messages of this type. Therefore, our *goal reduces to proving convergence of WP on the factor graph induced by $F|_\psi$, where ψ assigns the core variables their planted assignment, and the rest are UNASSIGNED.*

We say that WP converged correctly in a connected component C of the non-core factor graph if there exists a satisfying assignment ψ of the entire formula which is consistent with φ on the core, and with the assignment of WP to C.

Consider a connected component in the non-core factor graph consisting of a cycle with trees hanging from it. Our analysis proceeds in three steps:

1. We first prove that clause-variable and variable-clause messages of the form $\alpha \to \beta$ where $\alpha \to \beta$ lead from the trees to the cycle, converge correctly w. r. t. the planted assignment. In the case that the component has no cycles, this concludes the proof.

2. Then, using a refined case analysis, we show that the messages along the cycle also converge w. h. p, this time not necessarily to the planted assignment, but to some satisfying assignment which agrees with the already converged messages.

3. Finally, we conclude by showing that messages from the cycles to the trees converge. Moreover, this will imply that convergence will be to values consistent with the values converged to in step (1), and that hence that all messages in the connected component converge correctly according to some satisfying assignment.

Consider the factor graph F induced by the simplified formula. A *cycle* in F is a collection $x_1, C_2, x_3, C_4, \ldots, x_r = x_1$ where x_i and x_{i+2} belong to C_{i+1} for all i (in our description we consider only odd values of i) and $x_i \neq x_{i+2}$, $C_{i+1} \neq C_{i+3}$ for all i. A factor graph F is a *tree* if it contains no cycles. It is *unicyclic* if it contains exactly one cycle. Let $x \to C$ be a directed edge of F. We say that $x \to C$ *belongs* to the cycle, if both x and C belong to the cycle. For an edge $x \to C$ that does not belong to the cycle, we say that $x \to C$ *is directed towards* the cycle if x does not belong to the cycle and C lies on the simple path from x to the cycle. We say that the edge $x \to C$ is *directed away* from the cycle if C does not belong to the cycle and x lies on the simple path from the cycle to C. Similarly we define what it means for an edges $C \to x$ to belong to the cycle, to be directed towards the cycle and to be directed away from the cycle.

Proposition 4.5. *Let F be a unicyclic factor graph. Then every directed edge of the form $x \to C$ or $C \to x$ either belongs to the cycle, or is directed towards it or directed away from it.*

Proof. Recall that the factor graph is an undirected graph, and the direction is associated with the messages. Take an edge $x \to C$ (similarly for $C \to x$), if it lies on the cycle, then we are done. Otherwise, since the factor graph is connected, consider the path in the tree leading from some element of the cycle to C. This path is either contained in the path to x or contains it (otherwise there is another cycle). In the first case $x \to C$ is directed towards the cycle, and in the latter $x \to C$ is directed away from the cycle. $\square$

Our analysis proceeds in two parts: first we shall analyze WP on the trees, then WP on the cycle and connect the two (which is relevant for the uni-cyclic components).

4.4 WP on the trees

As we already mentioned before, there are two directions to consider: messages directed towards the cycle and away from the cycle. In this section we shall consider a rooted tree, and partition the messages according to messages which are oriented away from the root (they will correspond in the sequel to messages going away from the cycle) and messages towards the root (messages from the leaves towards the root—later to be identified with messages going into the cycle). The first lemma concerns messages going towards the root.

Remark 4.6. Lemma 4.7 is a special case of the known fact (see [7] for example) that for every tree induced by a satisfiable formula, WP converges and there exists a satisfying assignment ψ such that every B_i is either 0 or agrees with ψ. In Lemma 4.7 we assume that the formula is satisfiable by the all 1 assignment (the planted assignment), and consider only messages towards the root.

Lemma 4.7. *Let $C \to x$ be an edge in the non-core factor graph belonging to a connected component of size s, and in particular to a rooted tree T. If $C \to x$ is directed towards the root then the message $C \to x$ converges after at most $O(s)$ iterations. Furthermore, if $C \to x = 1$ then x appears positively in C.*

Proof. We consider the case $C = (\ell_x \vee \ell_y)$—the case $C = (\ell_x \vee \ell_y \vee \ell_z)$ where all three literals belong to non-core variables is proved similarly. For an edge (C,x) in the factor graph, we define $\texttt{level}(C,x)$ to be the number of edges in a path between C and the leaf most distant from C in the factor graph from which the edge (C,x) is removed. The lemma is now proved using induction on the level i. Namely, after the $i'th$ iteration, all messages $C \to x$ associated with an edge (C,x) at level i converge, and if $C \to x = 1$ then x appears positively in C.

The base case is an edge (C,x) at level 0. If $\texttt{level}(C,x) = 0$ then C is a unit clause containing only the variable x. By the definition of the messages, in this case $C \to x = 1$ and indeed it must be the case that x is positive in C (as the other two variables evaluate to FALSE under the planted). Now consider an edge (C,x) at level i, and consider iteration i. Since $i > 0$, it must be that there is another non-core variable y in C (or two more variables y,z). Consider an edge (C',y), $y \in C'$ (if no such C' exists that we are done as $C \to x$ will be constantly 0 in this case).

$\texttt{level}(C',y)$ is strictly smaller than i since every path from C to a leaf (when deleting the edge (C,x)) passes through some edge (C',y). By the induction hypothesis, all messages $C' \to y$ already converged, and therefore also $y \to C$ and in turn $C \to x$. It is only left to take care of the case $C \to x = 1$. In this case, there must be a clause C' such that $C' \to y = 1$ and y appears positively in C' (by the induction hypothesis). If $C \to x = 1$ it must be that y appears negatively in C and therefore x must appear positively (otherwise C is not satisfied by the planted assignment). □

Next we consider several scenarios that correspond to messages going form the root towards the leaves. Those scenarios correspond to step (3) of our analysis, referred to in Section 4.3.

Claim 4.8. *Assume that F is a unicyclic formula. Assume further that WP has converged on F. Let $x \to C$ be directed away from the cycle. Let F_C be the subformula inducing the tree rooted at C, while x itself is removed from C. This formula contains all clauses whose path to the cycle goes via x and a clause corresponding to C where the literal corresponding to the variable x is removed. Then $C \to x = 0$ in the fixed point if and only if F_C is satisfiable.*

Proof. The structure of the proof is similar to that of Lemma 4.7. For convenience we extend the definition of level above as to include edges on the cycle. We say that an edge (C,x) in the factor graph has $\texttt{level}(C,x)$ equal ∞ if (C,x) lies on a cycle and $\texttt{level}(C,x) = t < \infty$ if t is the maximal length of a path between C and a leaf in the factor graph from which the edge (C,x) is removed. The lemma is now proved using induction on t.

The base case is an edge (C,x) at level 0. If $\texttt{level}(C,x) = 0$ then C is a unit clause containing only the variable x, and then F_C is the empty formula. Indeed $C \to x = 1$ by definition and F_C is unsatisfiable (by definition again, the empty formula is not satisfiable).

Now consider an edge (C,x) at level $t > 0$. Assume $C = (x \vee \ell_y \vee \ell_z)$ (maybe only ℓ_y). Observe that every edge (C_j, y) in F_C has `level` value which is strictly smaller than t—since every path from C to a leaf (when deleting the edge (C,x)) passes through some edge (C_j, y). First we prove that if F_C is not satisfiable then it must be that $C \to x = 1$. If F_C is not satisfiable then it must be that $\ell_y = \bar{y}$ and $\ell_z = \bar{z}$ (otherwise, if one of them is positive then φ satisfies F_C). Further, observe that there must exist at least one clause C_j containing y positively, and at least one C_k containing z positively such that F_{C_i} and F_{C_j} are unsatisfiable. Otherwise, we can define φ' to be φ except that y or z are assigned FALSE (depending which of C_i or C_j does not exist). It is easy to see that φ' satisfies F_C, contradicting our assumption. By the induction hypothesis $C_i \to y = 1$ and $C_j \to z = 1$. This in turn implies that $C \to x_i = 1$ (since C_i and C_j contain y and z respectively in an opposite polarity to C and there cannot be any message $C_k \to y = 1$ where y appears negatively in C_k since F_{C_k} is satisfiable in this case). Now assume that $C \to x = 1$. The same arguments imply that C must be of the form $C = (x \vee \bar{y} \vee \bar{z})$ and there exists C_i, C_j as above. By the induction hypothesis, if one is to satisfy F_C it must be that $y = z =$ TRUE (this is the only way to satisfy F_{C_i} and F_{C_j} when inserting y back to C_i and z to C_j), but then C is not satisfied. □

Recall our definition for WP converges correctly on a subformula F' of F if there exists a satisfying assignment ψ to F which is consistent with the planted assignment φ on the core, and with the assignment of WP to F'.

Claim 4.9. *Assume that F is a unicyclic formula. Assume further that WP has converged on F. Let $C \to y$ be directed away from the cycle. Consider a subformula F_C which induces a tree rooted at a clause C. This formula contains the clause C and all other clauses whose path to the cycle goes via y. If in the fixed point for F it holds that*

- $C \to y = 1$,
- *y appears negatively in C,*
- $y \to C \geq 1$,

then WP converge correctly on F_C.

Proof. Let us track the origin of the message $y \to C$ in the tree (which is directed towards the root). For $y \to C \geq 1$ to occur, there must be a clause D_1 in the tree that contains y positively and a message $D_1 \to y = 1$ in the direction of the root (as messages in the direction of the root are only affected by other messages in that direction). Let us backtrack one more step. $D_1 = (y \vee \bar{k} \vee \bar{w})$ for some variables w, k; k and w must appear negatively in D_1 by Lemma 4.7), and the fact that $D_1 \to y = 1$, that is both k and w were issued warnings having them not satisfy D_1. Let us consider the clauses D_2 and D_3 that issues warnings to k and w respectively. $D_2 = (k \vee \ldots)$, $D_3 = (w \vee \ldots)$, $D_2 \to k = 1$ and $D_3 \to w = 1$, and both messages are directed towards the root. Obviously, one can inductively continue this backtracking procedure which terminates at the leaves of the tree (since there are not cycles the procedure is well defined and always terminates). Let us call the clauses and variables that emerge in this backtrack the *spine* of the tree (see Figure 1). The figure below illustrates this procedure.

For the subtree corresponding to the spinal variable we show that all messages that point away from the root converge in a consistent way with planted assignment. This combined with Lemma 4.7 completes that part of the proof.

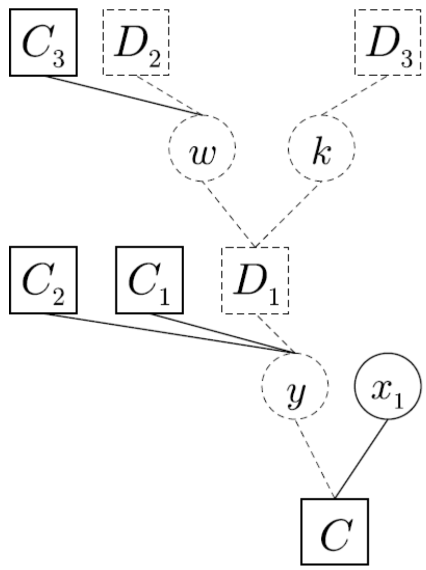

Figure 1: The spine of a tree

Let k be some variable that belongs to the spine, and let F_k be the subformula corresponding to the tree hanging from k (and we think of the messages along that tree oriented away from the cycle). We claim that $k \to C \geq 0$ for every C in F_k. This is proven via induction on the distance of the variable from y. The base case is distance 0, which is y itself. The messages that we need to verify are of the form $y \to C'$, $C' \neq D_1$, which are pointing away from the cycle. In this case $y \to C' \geq 0$ (y's message agrees with the planted); this is because the wrong message $C_{i+1} \to y$ is evened by the correct warning $D_1 \to y$, and $y \to C'$ depends only on one message which is directed away from the cycle—otherwise there is a second cycle. The induction step follows very similarly. This fact guarantees correct convergence of the variables in the trees hanging from spinal variables. Finally, for the spinal variables, there is always at most one message in the direction away from the cycle (otherwise there is more than one cycle); this message may be wrong. Using a very similar inductive argument one can show that there is always at least one correct warning (in the direction of the cycle), therefore $B_w \geq 0$ for every spinal variable w.

As for the non-spinal parts of the tree that hang on y, say a clause $M \neq D_1, C$, then $y \to M \geq 0$ since the wrong message of C is evened by the correct warning of D_1 (and there may be other correct warnings from messages in the direction of the cycle). Since there is a satisfying assignment of that subtree which is consistent with $B_y \geq 0$, Remark 4.6 can be applied to guarantee correct convergence. □

4.5 WP on cycles

We will denote a cycle by $x_1, C_2, x_3, C_4 \ldots, x_{2r-1}, C_{2r}, x_1$ where by this we mean that x_i appears in the clauses before/after it and that C_i contains the two variables before/after it. We consider two different types of cycles.

- *Biased* cycles: cycles that have at least one warning message $C \to x_i = 1$ coming into the cycle, where $C \to x_i$ directs into the cycle and the value of $C \to x_i$ is the value after the edge has converged.

- *Free* cycles: cycles that do not have such messages coming in, or all messages coming in are 0 messages.

4.5.1 Convergence of WP when the cycle is biased

First we observe that we may assume that edges that enter the cycle enter it at a variable rather than at a clause (hence that every clause on the cycle contains exactly two non-core variables). This is because of a simple argument similar to Proposition 4.4: consider an edge going into the cycle, $z \to C$, and assume that z appears positively in C. After all the edges going into the cycle have converged, if $z \to C \geq 0$ it follows that $C \to x = 0$ for cycle edges (C,x), and thus execution on the cycle is the same as if C was removed from the formula, only now we are left with a tree, for which convergence to a correct assignment is guaranteed (Remark 4.6). If $z \to C < 0$, then the execution is exactly as if z was removed from C (and C is in 2-CNF form).

Proposition 4.10. *Let $\mathcal{C}$ be a connected component of the factor graph of size s containing one cycle such that there exists an edge directed into the cycle $C \to x_i$ where x_i belongs to the cycle and such that the message converges to $C \to x_i = 1$. Then WP converges on $\mathcal{C}$ after at most $O(s)$ rounds. Moreover for the fixed point, if the message $C' \to x = 1$ then x appears positively in C'.*

Proof. A message of the cycle $C_j \to x_{j+1}$ depends only on cycle messages of the type $C_{j'} \to x_{j'+1}$, $x_{j'+1} \to C_{j'+2}$ and on messages coming into the cycle. In other words during the execution of WP the values of all messages $C_{j'} \to x_{j'-1}$, $x_{j'-1} \to C_{j'-2}$ do not affect the value of the message $C_j \to x_{j+1}$. Recall that we are in the case where there exists a message $C \to x_i = 1$ going into the cycle (after the convergence of these messages). Also x_i must appear positively in C. We consider the following cases:

- There exists a variable x_j that appears positively in both C_{j-1} and C_{j+1} (the case $j = i$ is allowed here). We note that in this case the message $x_j \to C_{j+1}$ must take the value either 0 or 1 which implies that the message $C_{j+1} \to x_{j+2}$ converges to the value 0. This in turn implies that the value of all messages $x_r \to C_{r+1}$ and $C_{r+1} \to x_{r+2}$ for $r \neq j$ will remain the same if the clause C_{j+1} is removed from the formula. However, this case reduces to the case of tree formula.

- x_i appears negatively in C_{i+1} and positively in C_{i-1}. We note that in this case the message $x_i \to C_{i+1}$ always take the value 1 which implies that the message $C_{i+1} \to x_{i+2}$ always take the value 1. Thus in this case we may remove the clause C_{i+1} from the formula and replace it by the unit clause ℓ_y where $C_{i+1} = \ell_y \vee \bar{x}_i$. Again, this reduces to the case of a tree formula.

- The remaining case is the case where x_i appears negatively in both C_{i-1} and C_{i+1} and there is no j such that x_j appears positively in both C_{j-1} and C_{j+1}. We claim that this leads to contradiction. Note that by the lemma above there exists a satisfying assignment where $x_i = 1$. Write $C_{i+1} = \bar{x}_i \vee \ell_{i+2}$. Then for the truth assignment we must have $\ell_{i+2} = 1$, similarly $\ell_{i+4} = 1$ etc. until we reach $\bar{x}_i = 1$—a contradiction.

To summarize, by Lemma 4.7 the messages going into the rooted tree at x_i converge after $O(s)$ steps, and at least one warning is issued. By the above discussion, for every clause D in the connected component it holds that $x_i \to D \geq 0$ (as x_i appears in at most one message which may be wrong—a cycle message). Since there is always a satisfying assignment consistent with x_i assigned TRUE, then after reducing the cycle to a tree we are left with a satisfiable tree. Remark 4.6 guarantees convergence in additional $O(s)$ iterations. □

4.5.2 Convergence of WP when the cycle is free

The main result of this subsection is summarized in the following claim:

Claim 4.11. *Let C be a connected component of the factor graph of size s containing one cycle of size r such that the fixed point contains no messages $C \to x = 1$ going into the cycle (the cycle is free). Then w. h. p. WP converges on C after at most $O(r^2 \cdot \log n + s)$ rounds. Moreover for the fixed point, if we simplify the formula which induces C according to the resulting B_i's, then the resulting subformula is satisfiable.*

Remark 4.12. Observe that the free case is the only one where convergence according to the planted assignment is not guaranteed. Furthermore, the free cycle case is the one that may not converge "quickly" (or not at all), though this is extremely unlikely. The proof of Proposition 4.11 is the only place in the analysis where we use the fact that in line 3.a of WP we use fresh randomness in every iteration.

We consider two cases: the easy one is the case in which the cycle contains a pure variable w. r. t the cycle (though this variable may not be pure w. r. t to the entire formula).

Claim 4.13. *If the cycle contains a variable x_i appearing in the same polarity in both C_{i+1}, C_{i-1}, then the messages $C \to x$ along cycle edges converge. Moreover for the fixed point, if $C \to x = 1$ then x satisfies C according to φ.*

The proof is essentially the first case in the proof of Proposition 4.10. We omit the details. We now move to the harder case, in which the cycle contains no pure variables (which is the case referred to in Remark 4.12).

Proposition 4.14. *Consider a free cycle of size r with no pure literal, and one of the two directed cycles of messages. Then the messages along the cycle converge w. h. p. to either all 0 or all 1 in $O(r^2 \log n)$ rounds.*

Convergence w. h. p. in polynomial time suffices due to Corollary 3.12 (which asserts that w. h. p. every cycle is of length at most $\log^{\varepsilon} n$ for every $\varepsilon > 0$). The proof of Proposition 4.14 is given in the end of this section. We proceed by analyzing WP assuming that Proposition 4.14 holds, which is the case w. h. p.

Proposition 4.15. *Suppose that the cycle messages have converged (in the setting of Proposition 4.14), then the formula resulting from substituting every x_i with the value assigned to it by B_i (according to the fixed point of WP), and simplifying, is satisfiable.*

Proof. Let F be the subformula that induces the connected component C, and decompose it according to the trees that hang on the cycle's variables and the trees that hang on the cycle's clauses. Observe that the formulas that induce these trees are variable and clause disjoint (since there is only one cycle in the C).

Let us start with the cycle clauses. The key observation is that setting the cycle variables according to one arbitrary orientation (say, set x_i to satisfy C_{i+1}) satisfies the cycle and does not conflict with any satisfying assignment of the hanging trees: if the tree hangs on a variable x_i, then since the cycle is free, the tree is satisfiable regardless of the assignment of x_i (Proposition 4.8). In the case that the tree hangs on a cycle-clause C, then the cycle variables and the tree variables are disjoint, and C is satisfied already by a cycle-variable regardless of the assignment of the tree-variables. Now how does this coincide with the result of WP. Recall that we are in the case where the cycle is free. Therefore only messages $C \to x_i$ where both C and x_i belong to the cycle affect B_i. If in the fixed point one cycle orientation is 0 and one orientation is 1, then the B_i messages of the cycle variables implement exactly this policy. If both cycle orientations converged to 1 or to 0, then the corresponding B_i messages of all cycle variables are UNASSIGNED (since the cycle is free), but then the same policy can be used to satisfy the clauses of the cycle in a manner consistent with the rest of the formula.

It remains to show that WP converges on every tree in a manner that is consistent with some satisfying assignment of the tree. We consider several cases.

Consider a tree hanging on a cycle variable x_i. Let C be some non-cycle clause that contains x_i, and F_C the subformula that induces the tree rooted at C. Observe that once the cycle has converged, then the message $x_i \to C$ does not change anymore. If $x_i \to C$ agrees with φ there are two possibilities. Either x_i satisfies C under φ, in which case C always sends 0 to F_C, and then WP executes on F_C as if C is removed. Remark 4.6 guarantees correct convergence (as $F_C \setminus C$ is satisfiable), and as for C, $B_i \geq 0$ and we can set x_i to TRUE so that it satisfies C and is consistent with the assignment of the cycle ($B_i \geq 0$ since $x_i \geq 0$ and $C \to x_i = 0$ as we are in the free cycle case). If x_i appears negatively in C, then WP executes as if x_i was deleted from C. Still F_C is satisfiable and correct convergence is guaranteed.

Now consider the case where $x_i \to C$ disagrees with φ. Recall that we assume $\varphi(x_i) = \mathsf{TRUE}$, and therefore $x \to C$ is negative in the fixed point. If x_i appears negatively in C then $C \to y = 0$ for every $y \in C$ (since x_i signals C that it satisfies it), and therefore C does not affect any calculation from this point onwards, and the correct convergence of F_C is again guaranteed by Remark 4.6 on the convergence for satisfiable trees. The more intricate case is if C contains x_i positively. Since we are in the free case, it must hold that $C \to x = 0$. Therefore using Proposition 4.8 one obtains that F_C is satisfiable (regardless of the assignment of x_i), and WP will converge as required (again Remark 4.6).

Now consider a tree hanging on a cycle clause. Namely, $C_{i+1} = (x_i \vee x_{i+2} \vee y)$, where x_i, x_{i+2} are cycle variables, and (C_{i+1}, y) is a tree edge. If one of the cycle orientations converged to 0, then $C_{i+1} \to y$ converges to 0, and then Remark 4.6 guarantees correct convergence. The same applies to the case where $C_{i+1} \to y$ converges to 1 and y is positive in C_{i+1} (since then we can remove y from the tree, use Remark 4.6 for the remaining part, then add back y, and set it to TRUE without causing any conflict with the tree assignment, but satisfying C_{i+1} according to the planted assignment).

The delicate case remains when $C_{i+1} \to y$ converges to 1 but y's polarity in C_{i+1} disagrees with φ, that is, y is negative in C_{i+1}. The key observation is that the message $y \to C_{i+1}$ (which is directed towards the cycle) must have converged to a positive value (otherwise, $C_{i+1} \to x_i$ and $C_{i+1} \to x_{i+2}$ would have converged to 0). However this complies with the scenario of Proposition 4.9, and again correct convergence is guaranteed. □

In Theorem 1.1 the unassigned variables are required to induce a "simple" formula, which is satisfiable in linear time. Observe that the factor graph induced by the UNASSIGNED variables consists of connected components whose structure is a cycle with trees hanging on it, or just a tree. A formula whose factor graph is a tree can be satisfied in linear time by starting with the leaves (which are determined uniquely in case that the leaf is a clause—namely, a unit clause, or if the leaf is a variable then it appears only in one clause, and can be immediately assigned) and proceeding recursively. Regarding the cycle, consider an arbitrary variable x on the cycle. By assigning x and simplifying accordingly, we remain with a tree. Since there are only two ways to assign x, the whole procedures is linear in the size of the connected component. This completes the proof of Theorem 1.1.

4.5.3 Proof of Proposition 4.14

Since the cycle has no pure literal it must be of the following form: $C_1 = (\ell_{x_1} \vee \overline{\ell}_{x_2}), C_2 = (\ell_{x_2} \vee \overline{\ell}_{x_3}), \ldots, C_L = (\ell_{x_L} \vee \overline{\ell}_{x_1})$.

Consider one of the directed cycles, say: $x_1 \to C_1 \to x_2 \to \cdots$ and note that when the message $x_i \to C_i$ is updated it obtains the current value of $C_{i-1} \to x_i$ and when the message $C_i \to x_{i+1}$ is updated, it obtains the current value of $x_i \to C_i$.

It thus suffices to show that the process above converges to all 0 or all 1 in time polynomial in the cycle length. This we prove in the lemma below.

Claim 4.16. *Consider the following process on $\{0,1\}^L$. Given the state of the process Γ^i at round i, the state Γ^{i+1} at round $i+1$ is defined by choosing a permutation $\sigma \in S_k$ uniformly at random. Then let $\Delta_0 = \Gamma^i$ and for $1 \le j \le L$, let Δ_j be obtained from Δ_{j-1} by setting $\Delta_j(\sigma(j-1)) = \Delta_{j-1}((\sigma(j-1)+1) \bmod L)$ and $\Delta_j(i) = \Delta_{j-1}(i)$ for all $i \neq \sigma(j-1)$. Finally, let $\Gamma^{i+1} = \Delta_L$.*

Let T be the stopping time where the process hits the state all 0 or all 1. Then

$$\Pr[T \ge 4aL^2] \le L2^{-a}. \tag{4.1}$$

for all $a \ge 1$ integer.

The proof of Proposition 4.16 is based on the fact that the process defined in this lemma is a martingale. This is established in the following two lemmas.

Lemma 4.17. *Consider the following variant $\tilde{\Gamma}^i$ of the process Γ^i defined in Proposition 4.16. In the variant, different intervals of 0/1 are assigned different colors and the copying procedure is as above. Fix one color and let X_i denote the number of elements of the cycle of that color in $\tilde{\Gamma}^i$. Then X_i is a martingale with respect to the filtration defined by $\tilde{\Gamma}^i$.*

Proof. From the definition of the copying process it is easy to see that

$$\mathbb{E}[X_{i+1} | \tilde{\Gamma}^i, \tilde{\Gamma}^{i-1}, \ldots] = \mathbb{E}[X_{i+1} | X_i].$$

We will show below that $\mathbb{E}[X_{i+1} | X_i] = 0$ and thus that X_i is a martingale with respect to the filtration $\tilde{\Gamma}^i$.

Assume that $X_i = k$. Then we may assume that the configuration $\tilde{\Gamma}^i$ consists of an interval of 1's of length k and an interval of 0's of length $L-k$. We calculate separately the expected shifts in the locations of left end-points of the 0 and 1 interval respectively. We denote the two shift random variables by L_0 and L_1. Clearly $L_0 = I_{0,1} + I_{0,2} + \cdots + I_{0,k-1}$ where $I_{0,j}$ is the indicator of the event that 0 left-end point shifted by at least j and similarly for L_1. Note that

$$\mathbb{E}[I_{0,j}] = \frac{1}{j!} - \frac{1}{(L-k+j)!}$$

and that

$$\mathbb{E}[I_{1,j}] = \frac{1}{j!} - \frac{1}{(k+j)!}\,.$$

The last equation follows from the fact that in order for the 1 interval to extend by at least j, the j copying has to take place in the correct order and it is forbidden that they all took place in the right order and the interval has become a 0 interval. The previous equation is derived similarly. Thus

$$\mathbb{E}[(X_{i+1} - X_i)|X_i] = \mathbb{E}[L_1] - \mathbb{E}[L_0] = \sum_{j=1}^{L-k}\left(\frac{1}{j!} - \frac{1}{(k+j)!}\right) - \sum_{j=1}^{k}\left(\frac{1}{j!} - \frac{1}{(L-k+j)!}\right) = 0\,.$$

This concludes the proof that X_i is a martingale. The proof follows. □

The proof of Proposition 4.16 follows by a union bound from the following lemma where the union is taken over all intervals.

Lemma 4.18. *Consider the process $\tilde{\Gamma}^i$ defined in Lemma 4.17. Fix one interval and let X_i denote its length. Let T be the stopping time where X_i equals either 0 or L. Then*

$$\Pr[T \geq 4aL^2] \leq 2^{-a}.$$

Proof. In order to bound the hitting probability of 0 and L, we need some bounds on the variance of the martingale differences. In particular, we claim that unless $k=0$ or $k=L$ it holds that

$$\mathbb{E}[(X_{t+1} - X_t)^2 | X_t = k] \geq 1/2\,.$$

If $k=1$ or $k=L-1$ this follows since with probability at least $1/2$ the end of the interval will be hit. Otherwise, it is easy to see that the probability that $X_{t+1} - X_t$ is at least 1 is at least $1/4$ and similarly the probability that it is at most -1 is at least $1/4$. This can be verified by considering the event that one end-points moves by at least 2 and the other one by at most 1.

Let T be the stopping time when X_T hits 0 or n. Then by a Wald kind of calculation we obtain:

$$\begin{aligned}
L^2 \geq \mathbb{E}[(X_T - X_0)^2] &= \mathbb{E}\big[\big(\sum_{t=1}^{\infty} 1(T \geq t)(X_t - X_{t-1})\big)^2\big] \\
&= \mathbb{E}\big[\sum_{t,s=1}^{\infty} (X_t - X_{t-1})(X_s - X_{s-1}) 1(T \geq \max\{t,s\}\big] \\
&= \mathbb{E}\big[\sum_{t=1}^{\infty} (X_t - X_{t-1})^2 1(T \geq t)\big] \geq \frac{1}{2}\sum_{t=1}^{\infty} P[T \geq t] = \mathbb{E}[T]/2\,,
\end{aligned}$$

where the first equality in the last line follows from the fact that if $s<t$ say then:

$$\begin{aligned}
&\mathbb{E}[(X_t-X_{t-1})(X_s-X_{s-1})1(T\geq \max t,s)]=\\
&=\mathbb{E}[(X_t-X_{t-1})(X_s-X_{s-1})(1-1(T<t))]=\\
&=\mathbb{E}[\mathbb{E}[(X_t-X_{t-1})(X_s-X_{s-1})(1-1(T<t))|X_1,\ldots,X_{t-1}]]=\\
&=\mathbb{E}[(X_s-X_{s-1})(1-1(T<t))E[X_t-X_{t-1}|X_1,\ldots,X_{t-1}]]=0\,.
\end{aligned}$$

We thus obtain that $\mathbb{E}[T]\leq 2L^2$. By Markov's inequality, this implies in turn that $\Pr[T\geq 4L^2]\leq 1/2$. Since X_t is a Markov chain it now follows that for all $a\in\mathbb{N}$,

$$\Pr[T\geq 4aL^2|T\geq 4(a-1)L^2]\leq 1/2\,,$$

so

$$\Pr[T\geq 4aL^2]=\prod_{b=1}^{a}\frac{\Pr[T\geq 4bL^2]}{\Pr[T\geq 4(b-1)L^2]}=\prod_{b=1}^{a}\Pr[T\geq 4bL^2|T\geq 4(b-1)L^2]\leq 2^{-a}\,.$$

The proposition follows.

□

5 Proof of Proposition 3.11

We shall start with the proof of item (2). Then we shall remark how to adjust this proof to prove (3). Item (1) was proven, though with respect to a slightly different definition of a core, in [18]. The proof of item (1) is identical to that in [18], with the adjustments made along the proof of item (2). Details of the proof of (1) are omitted.

In order to prove Proposition 3.11 (2) it suffices to prove that w. h. p. there are no two cycles with a simple path (maybe of length 0) connecting the two. To this end, we consider all possible constellations of such prohibited subgraphs and prove the proposition using a union bound over all of them.

Every simple $2k$-cycle in the factor graph consists of k variables, say $x_1,\ldots,x_k$ (all different), and k clauses $C_1,\ldots,C_k$, such that $x_i,x_{i+1}\in C_i$. The cycle itself consists of $2k$ edges.

As for paths, we have 3 different types of paths: paths connecting a clause in one cycle with a variable in the other (type 1), paths connecting two clauses (type 2), and paths connecting two variables (type 3). Clause-variable paths are always of odd length, and clause-clause, variable-variable paths are always of even length. A k-path P consists of k edges. If it is a clause-variable path, it consists of $(k-1)/2$ clauses and the same number of variables. If it is a variable-variable path, it consists of $k/2-1$ variables and $k/2$ clauses and symmetrically for the clause-clause path (we don't take into account the clauses/variables that participate in the cycle, only the ones belonging exclusively to the path).

Our prohibited graphs consist of two cycles C_1,C_2 and a simple path P connecting them. We call a graph containing exactly two simple cycles and a simple path connecting them a *bi-cycle*. The path P can be of either one of the three types described above. Similarly to the bi-cycle case, one can have a cycle C and a chord P in it. We call such a cycle a *chord-cycle*. For parameters $i,j,k\in[1,n]$, and $t\in\{1,2,3\}$, we denote by $B_{2i,2j,k,t}$ a bi-cycle consisting of a $2i$-cycle connected by a k-path of type t to a $2j$-cycle. Similarly, we denote by $B_{2i,k,t}$ a chord-cycle consisting of a $2i$-cycle with a k-path of type t as a chord.

Our goal is then to prove that w. h. p. the graph induced by the non-core variables contains no bi-cycles and no chord-cycles.

For a fixed factor graph H we let $F_H \subseteq \mathsf{F}$ be a fixed minimal set of clauses inducing H, and $V(H)$ be the set of variables in H. In order for a fixed graph H to belong to the factor graph induced by the non-core variables it must be that there exists some F_H such that $F_H \subseteq \mathsf{F}$ and that $V(H) \subseteq \bar{\mathsf{H}}$ (put differently, $V(H) \cap \mathsf{H} = \emptyset$).

Let $B = B_{2i,2j,k,t}$ (or $B = B_{2i,k,t}$ if B is a chord-cycle) be a fixed bi-cycle and F_B a fixed minimal-set of clauses inducing B. We start by bounding $\Pr[F_B \subseteq \mathsf{F}$ and $V(B) \cap \mathsf{H} = \emptyset]$ and then use the union bound over all possible bi-cycles (chord-cycles) and inducing minimal sets of clauses. As the two events—$\{F_B \subseteq \mathsf{F}\}$ and $\{V(B) \cap \mathsf{H} = \emptyset\}$—are not independent, the calculations are more involved. Loosely speaking, to circumvent the dependency issue, one needs to defuse the effect that the event $\{F_B \subseteq \mathsf{F}\}$ might have on H. To this end we introduce a set H^*, defined very similarly to H only "cushioned" in some sense to overcome the dependency issues (the "cushioning" depends on F_B). This is done using similar techniques to [4, 18].

We start by defining the new set of core variables H^* (again w. r. t. an ordering π of the clause-variable messages and an initial values vector α). The changes compared to H are highlighted in bold.

Let B_1 be the set of variables whose support w. r. t. φ is at most $d/3$.
Let B_2 be the set of non-stable variables w. r. t. π where we redefine the gap in Definition 3.5 to be $(d/30-6)$ in (1) and (2).
*Let B_3 be the set of stable variables w. r. t. π which are violated by α where we redefine the gap in Definition 3.7 to be **(d/30-6)** in (1) and (2).*
Let $\boldsymbol{J} \subseteq V(B)$ be the set of variables appearing in no more than 6 different clauses in F_B

1. *Set $H'_0 = V \setminus (B_1 \cup B_2 \cup B_3 \cup (V(F_C) \setminus J))$.*
2. *While there exists a variable $a_i \in H'_i$ which supports less than $d/4$ clauses in $\mathsf{F}[H'_i]$ OR appears in more than **(d/30-6)** clauses* not *in $\mathsf{F}[H'_i]$, define $H'_{i+1} = H'_i \setminus \{a_i\}$.*
3. *Let a_m be the last variable removed at step 2. Define $\mathsf{H}^* = H'_{m+1}. = H'_{m+1}$.*

Propositions 3.6 and 3.8 could be easily adjusted to accommodate the 6-gap in the new definition in B_2 and B_3. Therefore Proposition 3.10 can be safely restated in the context of H^*:

Proposition 5.1. *If both α and π are chosen uniformly at random then w. h. p. $\#\mathsf{H}^* \geq (1-e^{-\Theta(d)})n$.*

Proposition 5.2. *Let $b = \#V(B)$, then the set J defined above satisfies $\#J \geq b/4$.*

Proof. Observe that if F_B is minimal then $\#F_B \leq b+1$. This is because in every cycle the number of variables equals the number of clauses, and in the worst case, the path contains at most one more clause than the number of variables, and the same goes for the chord-cycle. Now suppose in contradiction that $\#J < b/4$, then there are more than $3b/4$ variables in $V(B)$, each appearing in at least 6 different clauses in F_B. Thus, $\#F_B > (6 \cdot 3b/4)/3 = 1.5b > b+1$ (we divided by three as every clause might have been counted 3 times. The last inequality is true since $b \geq 3$), contradicting $\#F_B \leq b+1$. ☐

The following proposition "defuses" the dependency between the event that a bi-cycle (chord-cycle) was included in the graph and the fact that it does not intersect the core variables. In the following proposition we fix an arbitrary π and α in the definition of H^*, therefore the probability is taken only over the randomness in the choice of F.

Proposition 5.3. $\Pr[F_B \subseteq \mathsf{F}$ *and* $V(B) \cap \mathsf{H} = \emptyset] \leq \Pr[F_B \subseteq \mathsf{F}] \cdot \Pr[J \cap \mathsf{H}^* = \emptyset]$.

Before proving Proposition 5.3, we establish the following fact.

Lemma 5.4. *For every bi-cycle (chord-cycle) B and every minimal inducing set F_B, $\mathsf{H}^*(\mathsf{F}, \varphi, \alpha, \pi) \subseteq \mathsf{H}(\mathsf{F} \cup F_B, \varphi, \alpha, \pi)$.*

Proof. The lemma is proved using induction on i (i being the iteration counter in the construction of H). For the base case $H_0'(F) \subseteq H_0(F \cup F_B)$, since every variable in $H_0'(F)$ appears in at most 6 clauses in F_B it holds that $A_i(F \cup F_B) \subseteq B_i(F)$, $i = 2, 3$. $A_1(F \cup F_B) \subseteq B_1(F)$ holds at any rate as more clauses can only increase the support, and the set J was not even considered for H_0. Suppose now that $H_i'(\mathsf{F}) \subseteq H_i(\mathsf{F} \cup F_C)$, and prove the lemma holds for iteration $i+1$. If $x \in H_{i+1}'(\mathsf{F})$ then x supports at least $d/3$ clauses in which all variables are in $H_i'(\mathsf{F})$. Since $H_i'(\mathsf{F}) \subseteq H_i(\mathsf{F} \cup F_B)$, then x supports at least this number of clauses with only variables of $H_i(\mathsf{F} \cup F_C)$. Also, x appears in at most $d/30 - 6$ clauses with some variable outside of $H_i'(\mathsf{F})$, again since $H_i'(\mathsf{F}) \subseteq H_i(\mathsf{F} \cup F_B)$ and F_B contains at most 6 clauses containing x, x will appear in no more than $d/30$ clauses each containing some variable not in $H_i(\mathsf{F} \cup F_B)$. We conclude then that $x \in H_i(\mathsf{F} \cup F_B)$. □

This lemma clarifies the motivation for defining H^*. It is not necessarily true that $\mathsf{H}(F) \subseteq \mathsf{H}(F \cup F_B)$. For example, a variable which appears in $\mathsf{H}(F)$ could disappear from $\mathsf{H}(F \cup F_B)$ since the clauses in F_B make it unstable. Loosely speaking, H^* is cushioned enough to prevent such a thing from happening.

Proof. (Proof of Proposition 5.3)

$$\Pr[F_B \subseteq \mathsf{F} \text{ and } V(B) \cap \mathsf{H} = \emptyset] \leq \Pr[F_B \subseteq \mathsf{F} \text{ and } J \cap \mathsf{H} = \emptyset] = \Pr[J \cap \mathsf{H} = \emptyset \mid F_B \subseteq \mathsf{F}] \Pr[F_B \subseteq \mathsf{F}]\,.$$

Therefore, it suffices to prove

$$\Pr[J \cap \mathsf{H} = \emptyset \mid F_B \subseteq \mathsf{F}] \leq \Pr[J \cap \mathsf{H}^* = \emptyset]\,.$$

$$\Pr[J \cap \mathsf{H}^* = \emptyset] = \sum_{F : J \cap \mathsf{H}^*(F) = \emptyset} \Pr[\mathsf{F} = F] \underbrace{\geq}_{\text{Lemma 5.4}} \sum_{F : J \cap \mathsf{H}(F \cup F_B) = \emptyset} \Pr[\mathsf{F} = F]$$

Break each set of clauses F into $F' = F \setminus F_B$ and $F'' = F \cap F_B$, and the latter equals

$$\sum_{F' : F' \cap F_B = \emptyset, J \cap \mathsf{H}(F' \cup F_B) = \emptyset} \; \sum_{F'' : F'' \subseteq F_B} \Pr[\mathsf{F} \setminus F_B = F' \text{ and } \mathsf{F} \cap F_B = F'']$$

Since the two sets of clauses, $\mathsf{F} \setminus F_B$, and $\mathsf{F} \cap F_B$, are disjoint, and clauses are chosen independently, the last expression equals,

$$\begin{aligned}
&\sum_{F':F'\cap F_B=\emptyset, J\cap \mathsf{H}(F'\cup F_B)=\emptyset} \; \sum_{F'':F''\subseteq F_B} \Pr[\mathsf{F}\setminus F_B = F']\Pr[\mathsf{F}\cap F_B = F''] = \\
&\sum_{F':F'\cap F_B=\emptyset, J\cap \mathsf{H}(F'\cup F_B)=\emptyset} \Pr[\mathsf{F}\setminus F_B = F'] \underbrace{\sum_{F'':F''\subseteq F_B} \Pr[\mathsf{F}\cap F_B = F'']}_{1} = \\
&\sum_{F':F'\cap F_B=\emptyset, J\cap \mathsf{H}(F'\cup F_B)=\emptyset} \Pr[\mathsf{F}\setminus F_B = F'] \,.
\end{aligned}$$

Since $(\mathsf{F}\setminus F_B)\cap F_B = \emptyset$, and clauses are chosen independently, the event $\{F_B \subseteq \mathsf{F}\}$ is independent of the event $\{\mathsf{F}\setminus F_B = F'\}$. Therefore, the latter expression can be rewritten as

$$\sum_{F':F'\cap F_B=\emptyset, J\cap \mathsf{H}(F'\cup F_B)=\emptyset} \Pr[\mathsf{F}\setminus F_B = F' \mid F_B \subseteq \mathsf{F}] = \Pr[J\cap \mathsf{H} = \emptyset \mid F_B \subseteq \mathsf{F}] \,. \qquad \square$$

Lemma 5.5. *Let $B = B_{2i,k,t}$ be a chord-cycle, then $\Pr[V(B)\cap H^* = \emptyset \mid \#H^* = \lambda n] \le p(i,j,k)$ where:*

1. *$p(i,k) \le \lambda^{(i+\frac{k}{2}-1)/4}$ if B consists of a $2i$-cycle and a variable-variable k-path as a chord.*
2. *$p(i,k) \le \lambda^{(i+\frac{k}{2})/4}$ if B consists of $2i$-cycle and a clause-clause k-path as a chord.*
3. *$p(i,k) \le \lambda^{(i+\frac{k-1}{2})/4}$ if B consists of $2i$-cycle and a variable-clause k-path as a chord.*

Proof. In item 1, we have $i+(k/2)-1$ variables and $i+(k/2)$ clauses. To bound the event $\{J\cap \mathsf{H}^* = \emptyset\}$, given the size of H^*, observe that F_B is fixed in the context of this event, and there is no pre-knowledge whether F_B is included in F or not. Therefore, J can be treated as a fixed set of variables, thus the choice of H^* is uniformly distributed over J. Recalling that $\#J \ge (i+(k/2)-1)/4$, it follows that

$$\Pr[J\cap \mathsf{H}^* = \emptyset \mid \#\mathsf{H}^* = \lambda n] \le \frac{\binom{n-\#\mathsf{H}^*}{\#J}}{\binom{n}{\#J}} = \frac{\binom{\lambda n}{(i+\frac{k}{2}-1)/4}}{\binom{n}{(i+\frac{k}{2}-1)/4}} \le \lambda^{(i+\frac{k}{2}-1)/4}.$$

The last inequality follows from standard bounds on the binomial coefficients. This proves item 1. In the same manner items 2, 3 are proven (just counting how many variables and clauses B contains, depending on the type of its path). $\square$

Lemma 5.6. *Let $B = B_{2i,2j,k,t}$ be a bi-cycle, then $\Pr[V(B)\cap H^* = \emptyset | \#H^* = \lambda n] \le p(i,j,k)$ where:*

1. *$p(i,j,k) \le \lambda^{(i+j+\frac{k}{2}-1)/4}$ if B consists of a $2i,2j$-cycles and a variable-variable k-path.*
2. *$p(i,j,k) \le \lambda^{(i+j+\frac{k}{2})/4}$ if B consists of $2i,2j$-cycles and a clause-clause k-path.*
3. *$p(i,j,k) \le \lambda^{(i+j+\frac{k-1}{2})/4}$ if B consists of $2i,2j$-cycles and a variable-clause k-path.*

Lemma 5.6 is proven in a similar way to Lemma 5.5.

To complete the proof of Proposition 3.11, we use the union bound over all possible bi/chord-cycles. We present the proof for the bi-cycle case with a variable-variable path; the proof for all other cases is identical. $s = s_{i,j,k} = i + j + (k/2) - 1$ (namely, $\#V(B) = s$ and $\#F_B = s+1$). We also let $p_\lambda = \Pr[\#\mathsf{H}^* = \lambda n]$. The probability of B is then at most

$$\sum_{\lambda=0}^{n} p_\lambda \cdot \sum_{i,j,\frac{k}{2}=1}^{n} \binom{n}{s_{i,j,k}} \cdot (s_{i,j,k})! \cdot (7n)^{s_{i,j,k}+1} \cdot \left(\frac{d}{n^2}\right)^{s_{i,j,k}+1} \cdot \lambda^{s_{i,j,k}/4} \leq$$
$$\sum_{\lambda=0}^{n} p_\lambda \cdot \sum_{i,j,\frac{k}{2}=1}^{n} 7d \cdot \left(\frac{7en}{s}\right)^{s} \cdot s^s \cdot n^{s+1} \cdot \left(\frac{d}{n^2}\right)^{s+1} \cdot \lambda^{s/4} \leq \sum_{\lambda=0}^{n} p_\lambda \cdot \sum_{i,j,\frac{k}{2}=1}^{n} (7e \cdot d \cdot \lambda^{1/4})^s \cdot \frac{7d}{n}.$$

Let us now break the first sum according to the different values of λ. Proposition 3.10 implies that $p_\lambda < n^{-5}$ for all $\lambda > d^{-8}$. Therefore the contribution of this part to the (double) summation is $O(1/n)$. For $\lambda < d^{-8}$, the latter simplifies to

$$\sum_{\lambda=0}^{n/d^8} p_\lambda \sum_{i,j,\frac{k}{2}=1}^{n} \left(\frac{1}{2}\right)^{s} \cdot \frac{7d}{n}. \tag{5.1}$$

Finally observe that

$$\sum_{i,j,\frac{k}{2}=1}^{n} \left(\frac{1}{2}\right)^{s} \cdot \frac{7d}{n} \leq \sum_{i+j+\frac{k}{2}\leq 4\log n} \frac{7d}{n} + \sum_{i+j+\frac{k}{2}\geq 4\log n} \left(\frac{1}{2}\right)^{s} \leq (4\log n)^3 \cdot \frac{7d}{n} + n^3 \cdot \frac{1}{n^4} = o(1)\,.$$

Therefore, (5.1) sums up to $o(1)$.

5.1 Proof of Proposition 3.11 (3)

The proof is basically the same as that of Proposition 3.11 (2). One defines the same notion of "cushioned" core H^*, and proceeds similarly. We therefore reprove only the last part—the union bound over all possible cycles.

First let us bound the number of cycles of length k. There are $\binom{n}{k}$ ways to choose the variables inducing the cycle, and $k!/2$ ways to order them on the cycle. As for the set of clauses that induces the cycle, once the cycle is fixed, we have at most $(7n)^k$ ways of choosing the third variable and setting the polarity in every clause. We also let $p_\lambda = \Pr[\#\mathsf{H}^* = \lambda n]$.

Using the union bound, the probability of a cycle of length at least k in the non-core factor graph is at most

$$\sum_{\lambda=0}^{n} p_\lambda \cdot \sum_{t=k}^{n} \binom{n}{t} \cdot t! \cdot (7n)^t \cdot \left(\frac{d}{n^2}\right)^t \cdot \lambda^{t/2} \leq \sum_{\lambda=0}^{n} p_\lambda \cdot \sum_{t=k}^{n} \left(\frac{7en}{t}\right)^t \cdot t^t \cdot n^t \cdot \left(\frac{d}{n^2}\right)^t \cdot \lambda^{t/2}$$
$$= \sum_{\lambda=0}^{n} p_\lambda \cdot \sum_{t=k}^{n} (7e \cdot d \cdot \sqrt{\lambda})^t.$$

Let us now break the first sum according to the different values of λ. Proposition 3.10 implies that there exists a constant c such that $p_\lambda < n^{-3}$ for $\lambda > e^{-cd}$. Therefore the contribution of this part to the (double) summation is $O(1/n)$. For $\lambda \le e^{-cd}$, $7e \cdot d \cdot \sqrt{e^{-cd}} = e^{-\Theta(d)}$. In this case, the last summation is simply the sum of a decreasing geometric series with quotient $e^{-\Theta(d)}$, which sums up to at most twice the first term, namely $e^{-\Theta(dk)}$. The proposition then follows.

6 Discussion

Our results show that WP is effective in solving $\mathsf{P}_{n,p}^{\text{plant}}$. Though not being the first to give an algorithm for $\mathsf{P}_{n,p}^{\text{plant}}$, our results are a first example of rigorously analyzing a message passing algorithm on a natural non-trivial random SAT distribution. We remark that our goal was to analyze WP under its most common definition, resisting attempts to modify the algorithm in ways that would simplify the analysis. One such simplification would result if in the first few iterations, messages are updated in *parallel* in two phases: clause-variable messages updates and then variable-clause messages updates.

In the non-planted case, for low density formulas (considerably below the satisfiability threshold), some algorithms were rigorously shown to find w. h. p. a satisfying assignment efficiently [3, 8]. For random k-SAT, k a sufficiently large constant, the best rigorously analyzed algorithm is given in [10]. Experimental results predict that as the density of the formula increases, more sophisticated algorithms are needed in order to find a satisfying assignment.

One possible explanation for the increasing computational hardness of finding solutions in the non-planted case is based on the geometry of the space of satisfying assignment. It is now established [2, 1] that for k-SAT just below the satisfiability threshold and $k \ge 8$, the space of solutions decomposes into an exponential number of (Hamming-distance) connected clusters such that the distance between each two is linear in the number of variables. Such complex geometry of the space of solutions poses a complex algorithmic challenge.

Our results, and similarly [18, 17], show that in the planted case (with density some large constant above the satisfiability threshold), the algorithmic task of finding a satisfying assignment is relatively easy, and in particular, the naïve WP algorithm is effective in finding satisfying assignments. Planted formulas in this regime have only one cluster of satisfying assignments (see [11] for more discussion).

We conclude with an open problem. Can our analysis be extended to show that Belief Propagation (BP) finds a satisfying assignment to $\mathsf{P}_{n,p}^{\text{plant}}$ in the setting of Theorem 1.1? Experimental results predict the answer to be positive. However, our analysis of WP does not extend as is to BP. In WP, all warnings received by a variable (or by a clause) have equal weight, but in BP this need not be the case (there is a probability level associated with each warning). In particular, this may lead to the case that messages received from non-core portions of the formula can affect the core, a possibility that our analysis managed to exclude for the WP algorithm.

Acknowledgement

We thank Eran Ofek for many useful discussions. This work was done while the authors were visiting Microsoft Research, Redmond, Washington.

References

[1] DIMITRIS ACHLIOPTAS AND AMIN COJA-OGHLAN: Algorithmic barriers from phase transitions. In *Proc. 49th FOCS*, pp. 793–802. IEEE Comp. Soc. Press, 2008. [doi:10.1109/FOCS.2008.11] 647

[2] DIMITRIS ACHLIOPTAS, AMIN COJA-OGHLAN, AND FEDERICO RICCI-TERSENGHI: On the solution-space geometry of random constraint satisfaction problems. *Random Structures & Algorithms*, 38(3):251–268, 2011. Preliminary version in STOC'06. [doi:10.1002/rsa.20323] 647

[3] MIKHAIL ALEKHNOVICH AND ELI BEN-SASSON: Linear upper bounds for random walk on small density random 3-CNFs. *SIAM J. Comput.*, 36(5):1248–1263, 2007. Preliminary version in FOCS'03. [doi:10.1137/S0097539704440107] 620, 621, 647

[4] NOGA ALON AND NABIL KAHALE: A spectral technique for coloring random 3-colorable graphs. *SIAM J. Comput.*, 26(6):1733–1748, 1997. Prelimiary version in STOC'94. [doi:10.1137/S0097539794270248] 618, 621, 624, 627, 643

[5] NOGA ALON, MICHAEL KRIVELEVICH, AND BENNY SUDAKOV: Finding a large hidden clique in a random graph. *Random Structures & Algorithms*, 13(3-4):457–466, 1998. Preliminary version in SODA'98. [doi:10.1002/(SICI)1098-2418(199810/12)13:3/4<457::AID-RSA14>3.0.CO;2-W] 618

[6] NOGA ALON AND JOEL SPENCER: *The Probabilistic Method*. Wiley-Interscience Series in Discrete Mathematics and Optimization. Wiley, 2nd edition, 2000. 624, 626

[7] ALFREDO BRAUNSTEIN, MARC MÉZARD, AND RICCARDO ZECCHINA: Survey propagation: An algorithm for satisfiability. *Random Structures & Algorithms*, 27(2):201–226, 2005. [doi:10.1002/rsa.20057] 618, 634

[8] ANDREI Z. BRODER, ALAN M. FRIEZE, AND ELI UPFAL: On the satisfiability and maximum satisfiability of random 3-CNF formulas. In *Proc. 4th Ann. ACM-SIAM Symp. on Discrete Algorithms (SODA'93)*, pp. 322–330. ACM Press, 1993. [ACM:313559.313794] 620, 647

[9] HUI CHEN AND ALAN M. FRIEZE: Coloring bipartite hypergraphs. In *Proc. 5th Ann. Conf. on Integer Programming and Combinatorial Optimization (IPCO '96)*, pp. 345–358, London, UK, 1996. Springer. [doi:10.1007/3-540-61310-2_26] 618, 621

[10] AMIN COJA-OGHLAN: A better algorithm for random k-SAT. *SIAM J. Comput.*, 39(7):2823–2864, 2010. Preliminary version in ICALP'09. [doi:10.1137/09076516X] 647

[11] AMIN COJA-OGHLAN, MICHAEL KRIVELEVICH, AND DAN VILENCHIK: Why almost all k-CNF formulas are easy. In *Proc. 13th Internat. Conf. on Analysis of Algorithms (AofA'07)*, pp. 89–102. DMTCS, 2007. DMTCS. 620, 647

[12] AMIN COJA-OGHLAN AND ANGELICA Y. PACHON-PINZON: The decimation process in random k-SAT. *SIAM J. Discrete Math.*, 26(4):1471–1509, 2012. Preliminary versions in ICALP'11 and SODA'11. [doi:10.1137/110842867] 618

[13] STEPHEN A. COOK: The complexity of theorem-proving procedures. In *Proc. 3rd STOC*, pp. 151–158. ACM Press, 1971. [doi:10.1145/800157.805047] 617

[14] JAMES M. CRAWFORD AND LARRY D. AUTON: Experimental results on the crossover point in random 3-SAT. *Artif. Intell.*, 81(1-2):31–57, 1996. Preliminary version in AAAI'93. [doi:10.1016/0004-3702(95)00046-1] 618

[15] OLIVIER DUBOIS, YACINE BOUFKHAD, AND JACQUES MANDLER: Typical random 3-SAT formulae and the satisfiability threshold. In *Proc. 11th Ann. ACM-SIAM Symp. on Discrete Algorithms (SODA'00)*, pp. 126–127. ACM Press, 2000. [ACM:338243] 618

[16] URIEL FEIGE, ELCHANAN MOSSEL, AND DAN VILENCHIK: Complete convergence of message passing algorithms for some satisfiability problems. In *Proc. 10th Internat. Workshop on Randomization and Computation (RANDOM'06)*, pp. 339–350. Springer, 2006. [doi:10.1007/11830924_32] 617

[17] URIEL FEIGE AND DAN VILENCHIK: A local search algorithm for 3SAT. Technical report, The Weizmann Institute of Science, 2004. Available at author's home page. 623, 647

[18] ABRAHAM FLAXMAN: A spectral technique for random satisfiable 3CNF formulas. *Random Structures & Algorithms*, 32(4):519–534, 2008. Preliminary version in SODA'03. [doi:10.1002/rsa.20213] 618, 621, 623, 624, 627, 629, 642, 643, 647

[19] EHUD FRIEDGUT: Sharp thresholds of graph properties, and the k-SAT problem. *J. Amer. Math. Soc.*, 12(4):1017–1054, 1999. [doi:10.1090/S0894-0347-99-00305-7] 618

[20] WASSILY HOEFFDING: Probability inequalities for sums of bounded random variables. *J. Amer. Stat. Assoc.*, 58(301):13–30, 1963. JSTOR. 626

[21] ALEXIS C. KAPORIS, LEFTERIS M. KIROUSIS, AND EFTHIMIOS G. LALAS: The probabilistic analysis of a greedy satisfiability algorithm. *Random Structures & Algorithms*, 28(4):444–480, 2006. Preliminary version in ESA'02. [doi:10.1002/rsa.20104] 618

[22] ELIAS KOUTSOUPIAS AND CHRISTOS H. PAPADIMITRIOU: On the greedy algorithm for satisfiability. *Inform. Process. Lett.*, 43(1):53–55, 1992. [doi:10.1016/0020-0190(92)90029-U] 618

[23] MICHAEL KRIVELEVICH AND DAN VILENCHIK: Solving random satisfiable 3CNF formulas in expected polynomial time. In *Proc. 17th Ann. ACM-SIAM Symp. on Discrete Algorithms (SODA'06)*, pp. 454–463. ACM Press, 2006. [doi:10.1145/1109557.1109608] 618

[24] MICHAEL LUBY, MICHAEL MITZENMACHER, AMIN SHOKROLLAHI, AND DANIEL A. SPIELMAN: Efficient erasure correcting codes. *IEEE Trans. Inform. Theory*, 47(2):569–584, 2001. [doi:10.1109/18.910575] 621

[25] MICHAEL LUBY, MICHAEL MITZENMACHER, AMIN SHOKROLLAHI, AND DANIEL A. SPIELMAN: Improved low-density parity-check codes using irregular graphs. *IEEE Trans. Inform. Theory*, 47(2):585–598, 2001. Preliminary versions in ISIT'98 and STOC'98. [doi:10.1109/18.910576] 621

[26] JUDEA PEARL: Reverend Bayes on inference engines: A distributed hierarchical approach. In *Proc. 2nd Nat. Conf. on Artificial Intelligence (AAAI'82)*, pp. 133–136, 1982. AAAI. 618

[27] THOMAS J. RICHARDSON, AMIN SHOKROLLAHI, AND RÜDIGER L. URBANKE: Design of capacity-approaching irregular low-density parity-check codes. *IEEE Trans. Inform. Theory*, 47(2):619–637, 2001. [doi:10.1109/18.910578] 621

AUTHORS

Uriel Feige
Professor
The Weizmann Institute of Science, Rehovot, Israel
uriel.feige@weizmann.ac.il
http://www.wisdom.weizmann.ac.il/~feige/

Elchanan Mossel
Professor
U.C. Berkeley, CA, USA
mossel@stat.berkeley.edu
http://www.stat.berkeley.edu/~mossel/

Dan Vilenchik
Postdoctoral fellow
The Weizmann Institute of Science, Rehovot, Israel
dan.vilenchik@weizmann.ac.il
http://www.wisdom.weizmann.ac.il/~vilenchi/

ABOUT THE AUTHORS

URIEL FEIGE is a professor at the Weizmann Institute, though this work was done while he was a member of the theory group at Microsoft Research. His main research interests involve studying the borderline between P and NP as it manifests itself in approximation algorithms, heuristics, and exact algorithms that are not necessarily polynomial time.

ELCHANAN MOSSEL is a Professor of Statistics and Computer Science at U. C. Berkeley. He received his Ph. D. in mathematics from Hebrew University in 2000. After his Ph. D., he was a post-doc at the Theory Group of Microsoft Research, Redmond and a Miller fellow in Statistics and Computer Science here at U. C. Berkeley. His research interests include Combinatorial Statistics, Discrete Fourier Analysis and Influences, Randomized Algorithms, Computational Complexity, MCMC, Markov Random Fields, Social Choice, Game Theory and Evolution.

DAN VILENCHIK graduated from Tel Aviv University in 2009; his advisor was Michael Krivelevich. His thesis focused on average-case analysis of optimization problems, in particular k-colorability and k-SAT. His interests include average-case analysis and combinational optimization.