Complete convergence of message passing algorithms for some satisfiability problems

Uriel Feige* Elchanan Mossel† Dan Vilenchik‡

April 2006

Abstract

Experimental results show that certain message passing algorithms, namely, *survey propagation*, are very effective in finding satisfying assignments in random satisfiable 3CNF formulas. In this paper we make a modest step towards providing rigorous analysis that proves the effectiveness of message passing algorithms for random 3SAT. We analyze the performance of Warning Propagation, a popular message passing algorithm that is simpler than survey propagation. We show that for 3CNF formulas generated under 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-to-variable ratio is a sufficiently large constant. We are not aware of previous rigorous analysis of message passing algorithms for satisfiability instances, though such analysis was performed for decoding of Low Density Parity Check (LDPC) Codes. We discuss some of the differences between results for the LDPC setting and our results.

*Micorosoft Research and The Weizmann Institute. urifeige@microsoft.com.

†U.C. Berkeley. E-mail: mossel@stat.berkeley.edu. Supported by a Sloan fellowship in Mathematics, by NSF Career award DMS-0548249 and NSF grants DMS-0528488 and DMS-0504245

‡Tel-Aviv University. E-mail: vilenchi@post.tau.ac.il.

1 Introduction and Results

In this paper we analyze the performance of Warning Propagation (WP for brevity), a popular message passing algorithm, when applied to random satisfiable formulas generated under the planted distribution with a constant clause-variable ratio. 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 with clause-variable ratio below (yet rather close to) the conjectured satisfiability threshold, ~ 4.2 [6]. 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. We proceed with formal definitions, statement of the main result and a survey of related work.

1.1 Different SAT Distributions

A CNF formula over the variables $x_1, x_2, ..., x_n$ is a conjunction of clauses $C_1, C_2, ..., C_m$ 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 s.t. every clause contains at least one literal which evaluates to true. 3SAT is the language of all satisfiable 3CNF formulas. Although 2SAT is known to be in P, 3SAT is one of the most famous NP-complete problems [9]. In [18] it is proved that it is NP-hard to approximate MAX-3SAT (the problem of finding an assignment that satisfies as many clauses as possible) within a ratio better than 7/8. Given the difficulty of designing algorithms that work well in the worst case, we consider the average case performance of algorithms. One possibility for rigorously modelling average-case instances is to use random models.

Algorithmic theory of random structures has been the focus of extensive research in recent years (see [16] for a survey). As part of this trend, uniformly random 3CNFs (generated by selecting at random $m = m(n)$ clauses over the variables $\{x_1, ..., x_n\}$) have attracted much attention. Random 3SAT is known to have a sharp satisfiability threshold in the clause-to-variable ratio [15]. Namely, a random 3CNF with clause-to-variable ratio below the threshold is satisfiable **whp** (with high probability, meaning with probability tending to 1 as n goes to infinity) and one with ratio above the threshold is unsatisfiable **whp**. This threshold is not known exactly (and not even known to be independent of n). The threshold is known to be at least 3.52 [21] and at most 4.506 [11]. Experimental results indicate that the threshold is closer to the higher end of the interval [10].

In this work we mainly consider formulas with large clause-variable ratio. At such ratios almost all 3CNF formulas are not satisfiable, therefore more refined definitions are due. More formally, we consider three distributions on 3SAT instances. The first, analogous to the well known random graph model $G_{n,p}$, is the distribution in which every clause, out of $2^3\binom{n}{3}$ possible clauses, is included with probability $p = p(n)$. We denote this distribution by $\mathcal{P}_{n,p}$. The second distribution is obtained from $\mathcal{P}_{n,p}$ by conditioning on satisfiability, namely $\mathcal{P}^{\text{sat}}_{n,p}[F] = \mathcal{P}_{n,p}[F|S]$ where S is the event that F is satisfiable. Lastly, we consider the planted distribution, $\mathcal{P}^{\text{plant}}_{n,p}$, which is obtained form $\mathcal{P}_{n,p}$ by conditioning on satisfiability by a specific "planted" assignment φ. Equivalently, in $\mathcal{P}^{\text{plant}}_{n,p}$, first an assignment φ to the variables is picked uniformly at random. Then, every clause satisfied by φ is included with probability $p = p(n)$. Throughout, we use φ to denote the planted assignment when the relevant instance is clear from context.

In the context of satisfiability algorithms, $\mathcal{P}^{\text{sat}}_{n,p}$ is arguably the most interesting and natural distribution to study. However, as pointed out frequently (e.g. [4, 8]), $\mathcal{P}^{\text{sat}}_{n,p}$ seems hard to tackle rigorously and experimentally. The planted 3SAT distribution is an intermediate step towards analyzing $\mathcal{P}^{\text{sat}}_{n,p}$, and is an interesting, quite natural distribution on its own right, the analog the of planted clique, planted bisection, planted coloring, and planted bipartite hypergraphs studied e.g. in [2, 3, 5, 20, 12]. The planted 3SAT distribution is also discussed e.g. in [14, 4]. Our main result (Theorem 2) relates to the planted 3SAT model, but some of our other results (such as Proposition 3 and Corollary 5) are relevant to the $\mathcal{P}^{\text{sat}}_{n,p}$ setting as well.

1.2 3SAT and Factor Graphs

Let $\mathcal{F}$ be a 3CNF formula on n variables and m clauses. The *factor graph* (e.g. [23]) of $\mathcal{F}$, denoted by $FG(\mathcal{F})$, is the following graph representation of $\mathcal{F}$. The factor graph is a bipartite multigraph, $FG(\mathcal{F}) = (V_1 \cup V_2, E)$ where $V_1 = \{x_1, x_2, ..., x_n\}$ (the set of variables) and $V_2 = \{C_1, C_2, ..., C_m\}$ (the set of clauses). $(x_i, C_j) \in E$ iff x_i appears in C_j. For a 3CNF $\mathcal{F}$ with m clauses it holds that $\#E = 3m$ (as every clause contains exactly 3 variables). To make presentation clearer, we denote by $\#A$ the size of a set A and by $|a|$ the absolute value of a real number a. For simplicity we assume that every clause contains three distinct variables, therefore FG is a graph (no parallel edges).

1.3 Warning Propagation

Warning Propagation (WP) is a simple iterative message passing algorithm, and serves as an excellent intuitive introduction to more involved message passing algorithms such as Belief Propagation [26] and Survey Propagation [6]. 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 quite" 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 $\mathcal{F}$ be a CNF formula. For a variable x, let $N^+(x)$ be the set of clauses in $\mathcal{F}$ 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 positively appearing variables and respectively $N^-(C)$ the set of negatively appearing 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 undecided). 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 satisfy 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 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 $\psi \in \{TRUE, FALSE, UNASSIGNED\}^n$ can be generated according to the corresponding B_i values (as previously explained).

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 $(C_j \to x_i)$.

Algorithm 1 : WP(CNF formula $\mathcal{F}$)

1: construct the corresponding factor graph $FG(\mathcal{F})$.
2: randomly initialize clause-variable messages to 0 or 1.
3: **repeat**
4: randomly order the edges of $FG(\mathcal{F})$.
5: update all clause-variable messages $C_j \to x_i$ according to the random edge order.
6: **until** no clause-variable message changed from the previous iteration
7: compute a partial assignment ψ according to the B_i messages.
8: **return** ψ

In the above the description it seems like no update of variable-clause messages is carried out. This however is implicit in line 5. Namely, 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 end up at all (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 didn't converge by then, return a failure.

1.4 Related Work and Techniques

The Survey Propagation [6] algorithm experimentally outperforms all known algorithms in finding satisfying assignments to $\mathcal{P}_{n,p}$ formulas with clause-variable ratio ρ close to the satisfiability threshold ($4 \leq \rho \leq 4.25$). However, theoretical understanding of Survey Propagation and other message passing algorithm for random SAT problems is still lacking. This should be compared with the success of message passing algorithms for decoding low-density-parity-check (LDPC) codes [17]. Here, the experimental success of message passing algorithms [17] was recently complemented rigourously by a large body of theoretical work, see e.g. [24, 27, 25]. Some important insights emerge from this theoretical work. In particular, it is shown that the quality of decoding improves exponentially with the number of iterations – thus all but a small constant fraction of the received codeword can be decoded correctly using a constant number of iterations. Our analysis of WP on $\mathcal{P}_{n,p}^{\text{plant}}$ shows that much of the coding picture is valid also for $\mathcal{P}_{n,p}^{\text{plant}}$ thus providing important insights as to the success of message passing algorithms for random satisfiability problems. The planted 3SAT model is similar to LDPC 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 $\mathcal{P}_{n,p}^{\text{plant}}$, φ being the planted assignment, the clauses containing a variable x_i contain noisy information on the polarity of $\varphi(x_i)$ in the following sense – each clause contains x_i in a polarity coinciding with $\varphi(x_i)$ with probability 4/7. The SAT setting is however more involved than its coding counterpart; for example a SAT instance may have many satisfying assignments (which is **whp** the case in $\mathcal{P}_{n,p}^{\text{plant}}$ with clause-variable ratio of order $o(\log n)$) whereas a transmitted codeword has a unique true solution. More discussion follows in Section 1.6.

As for relevant results in random graph theory, the seminal work of [2] paved the road towards dealing with large-constant-degree planted distributions. [2] present an algorithm that **whp** k-colors planted k-colorable graphs (the distribution of graphs generated by partitioning the n vertices into k equally-sized color classes, and including every edge connecting two different color classes with probability p; commonly denoted $G_{n,p,k}$) with a sufficiently large constant expected degree. Building upon the techniques introduced by [2], [20] present an algorithm that 2-colors sparse planted 3-uniform bipartite hypergraphs and [14], solving an open question posed in [22], presents an algorithm for satisfying large constant degree planted 3SAT instances. Though in our analysis we use similar techniques to the aforementioned works, our result is conceptually different in the following sense. In [2, 20, 14] 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 message passing 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. This is similar in spirit to the work of [1] who showed RWalkSat works on very sparse $\mathcal{P}_{n,p}$ instances (for which other simple heuristics are also known to work), and to the work in [13], where a certain version of the k-opt heuristic is shown to work on $\mathcal{P}_{n,p}^{\text{plant}}$. Another kind of interplay between algorithms and random distributions is involved in the work on the lower end of the satisfiability threshold. Much of it is based on the analysis of simple heursitics, often too simple to be of practical value (e.g., in [7] the pure-literal heuristic is used for very sparse $\mathcal{P}_{n,p}$ instances).

Another difference between our work and that of [2, 20, 14] is that unlike the algorithms analyzed in those other papers, WP is a randomized algorithm 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).

1.5 Notation

Given a 3CNF $\mathcal{F}$, **simplify** $\mathcal{F}$ 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. Next, if a clause contains a literal which evaluates to true, remove the clause. Otherwise, remove all literals which evaluate to false (the resulting instance is not necessarily in 3CNF form). Denote by $\mathcal{F}|_\psi$ the 3CNF $\mathcal{F}$ simplified according to ψ. For a set of variables $A \subseteq V$, denote by $\mathcal{F}[A]$ the set of clauses in which all variables belong to A.

Given a 3CNF formula $\mathcal{F}$, we say that a variable x is **pure** in $\mathcal{F}$ if it appears only in one polarity (namely, always appears as the literal x or always as the literal $\bar{x}$). Let P_0 be the set of pure variables in $\mathcal{F}$, and C_0 be the set of clauses containing a pure variable. Let $L_0 = \mathcal{F}$, and $L_1 = L_0 \setminus C_0$. Let P_1 be the pure variables in L_1, namely the variables that become pure after setting the pure variables in a satisfying manner and simplifying $\mathcal{F}$. Similarly, define C_1 to be the set of clauses in L_1 containing a variable from P_1. Generally, define $L_i = L_{i-1} \setminus C_{i-1}$, P_i to be the pure variables in L_i, and C_i to be the clauses in L_i containing a variable from P_i. We say that a 3CNF $\mathcal{F}$ is *r-pure* if $L_r = \emptyset$.

The following theorem is implicitly proved in [7].

Theorem 1. *Let $\mathcal{F}$ be randomly sampled according to $\mathcal{P}_{n,p}$, $p = d/n^2$, $d < 1.225$, then **whp** $\mathcal{F}$ is $O(n)$-pure.*

Note that if there exists an r s.t. $\mathcal{F}$ is r-pure then in particular $\mathcal{F}$ is satisfiable. To better understand our results it would be convenient to have the somewhat informal notion of a *simple formula* in mind. We call a CNF formula simple, if it can be satisfied using simple well-known heuristics (examples include formulas whose factor graph is tree-like and r-pure formulas – both solvable **whp** using the pure-literal heuristic [7], formulas with small weight terminators – to use the terminology of [1] – solvable **whp** using a RWalkSat, etc).

1.6 Our Results

Theorem 2. *Let $\mathcal{F}$ 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, and let φ be its planted assignment. Then the following holds with probability $1 - e^{-\Theta(d)}$ (the probability taken over the choice of $\mathcal{F}$, the random choices in line 2 of the WP algorithm, and the random order in the first time line 4 executes):*

(a) *WP($\mathcal{F}$) converges after at most $O(\log n)$ iterations.*

(b) *Let ψ be the partial assignment returned by WP($\mathcal{F}$), 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, $\#V_A \geq (1 - e^{-\Theta(d)})n$.*

(c) *$\mathcal{F}|_\psi[V_U]$ is a simple formula which can be satisfied in time $O(n)$.*

Proposition 3. *Let $\mathcal{F}$ be a 3CNF formula randomly sampled according to $\mathcal{P}^{\text{plant}}_{n,p}$, where $p \geq c\log n/n^2$, with c a sufficiently large constant, and let φ be its planted assignment. Then **whp** after at most 2 iterations WP($\mathcal{F}$) converges, and the returned ψ equals φ. (This result can be extended to $\mathcal{P}^{\text{sat}}_{n,p}$, see below.)*

Proposition 4. *Let $\mathcal{F}$ be an r-pure CNF formula. Then after at most $O(r)$ iterations of WP($\mathcal{F}$), regardless of the initial messages and the order of execution, the following holds:*

(a) *WP($\mathcal{F}$) converges.*

(b) *Let ψ be the assignment returned by WP($\mathcal{F}$). If $\psi(x) \neq UNASSIGNED$, then in every satisfying assignment x is assigned according to $\psi(x)$.*

(c) *If $\mathcal{F}$ contains no unit clauses then ψ is the all-UNASSIGNED vector.*

Corollary 5. *In the setting of Theorem 1,* **whp** *WP($\mathcal{F}$) converges after at most $O(n)$ iterations and the returned ψ is the all-UNASSIGNED vector.*

The corollary follows immediately from Theorem 1 and Proposition 4.

Proposition 6. *Let $\mathcal{F}$ be a* satisfiable *CNF formula whose corresponding factor graph contains no cycles. Then $\mathcal{F}$ is $O(n)$-pure.*

The main idea behind the proof of Theorem 2 is to show that the formula is dense enough so that **whp** there exists a large subformula forcing WP to point in the correct direction. The rest of the formula induces a factor graph containing only trees, which are also "easy" for WP. It is worth noting that formulas in $\mathcal{P}_{n,p}^{\text{plant}}$, with n^2p some large constant, are not known to be simple (in the sense that we defined above), on the contrary, "hardness" evidence could be found in works such as [1], showing that RWalkSat is very unlikely to hit a satisfying assignment in polynomial time when running on a random $\mathcal{P}_{n,p}^{\text{plant}}$ instance in the setting of Theorem 2. In the setting of Proposition 3, the formula is already dense enough so that **whp** it forces entirely WP to point to the planted assignment. Proposition 3 can be restated with $\mathcal{P}_{n,p}^{\text{sat}}$ replacing $\mathcal{P}_{n,p}^{\text{plant}}$, because these distributions are essentially the same for sufficiently dense formulas (see [4] for a proof of this last fact).

Proposition 4 combined with Proposition 6 provide a proof to the convergence of WP on trees. This is not a new fact, and was proven e.g. in [6]. Nevertheless, Proposition 4 gives an explicit characterization of the fixed point to which WP converges (which is implicit for trees in [6]). Moreover, Proposition 4 concerns a wider family of CNF formulas (which includes trees as a strict subset), while providing a different perspective for the tree setting. For lack of space, most of the proofs are given in the appendix.

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. [24, 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 [25] 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. We on the other hand take a well known SAT distribution and analyze the performance of a message passing algorithm on it. Moreover, the SAT setting is more involved, as there are many assignments satisfying the formula, while for the erasure channel there is a unique codeword satisfying the combinatorial constraints given by the message.

The remainder of the paper is structured as follows. In Section 2 we discuss some properties that a typical instance in $\mathcal{P}_{n,p}^{\text{plant}}$ possesses, and outline the proof of Theorem 2 and Proposition 3. In Section 3 we summarize our results and discuss potentially interesting lines for further research.

2 Properties of a Random $\mathcal{P}_{n,p}^{\text{plant}}$ Instance

Recall the definition of $\mathcal{P}_{n,p}^{\text{plant}}$. First a truth assignment φ to the variables $V = \{x_1, x_2, ..., x_n\}$ is picked uniformly at random. Next, every clause satisfied by φ is included in the formula with probability p (in our case $p = d/n^2$, d a sufficiently large constant). Since φ satisfies $\mathcal{F}$, it is satisfiable. There are $(2^3 - 1) \cdot \binom{n}{3}$ clauses satisfied by φ. The expected size of $\mathcal{F}$ is $p \cdot 7 \cdot \binom{n}{3} =$

$7dn/6+o(n)$. We continue by discussing relevant properties of a random $\mathcal{P}_{n,p}^{\text{plant}}$ instance. Full proofs of the propositions ahead are available in the appendix. To simplify presentation, we assume w.l.o.g. (due to symmetry) that the planted assignment φ is the all-one vector.

2.1 Stable Variables

Definition 7. *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 8. *Let $\mathcal{F}$ be as in the setting of Theorem 2 and let F_{SUPP} be a random variable counting the number of variables in $\mathcal{F}$ whose support w.r.t. φ is less than $d/3$. Then, $E[F_{SUPP}] \leq e^{-\Theta(d)}n$.*

This follows from concentration arguments as every variable is expected to support $\frac{d}{n^2} \cdot \binom{n}{2} = \frac{d}{2} + O(\frac{1}{n})$ clauses.

Following the definitions in Section 1.3, given a CNF $\mathcal{F}$ and a variable x, we let $N^{++}(x)$ be the set of clauses in $\mathcal{F}$ in which x appears positively but doesn't support w.r.t. φ. Let $N^s(x)$ be the set of clause in $\mathcal{F}$ which x supports w.r.t. φ. Let $\pi = \pi(\mathcal{F})$ be some ordering of the clause-variable message edges in the factor graph of $\mathcal{F}$. 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 $(C \rightarrow x)$ 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 $\mathcal{E}$ and a set of clauses $\mathcal{C}$ we denote by $\mathcal{E} \cap \mathcal{C}$ the subset of edges containing a clause from $\mathcal{C}$ as one endpoint.

Definition 9. *A variable x is **stable** in $\mathcal{F}$ w.r.t. an edge order π if the following holds for every clause-variable edge $C \rightarrow x$ (w.l.o.g. assume $C = (\ell_x \vee \ell_y \vee \ell_z)$, $C \rightarrow x$ is the i'th message in π):*

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

(b) $|\#N^{++}(y) - \#N^{-}(y)| \leq d/30$.

(c) $\#N^s(y) \geq d/3$

and the same holds for z.

Proposition 10. *Let $\mathcal{F}$ be as in the setting of Theorem 2, and let π be a random ordering of the clause-variable messages. Let F_{UNSTAB} be a random variable counting the number of variables in $\mathcal{F}$ which are not stable. Then, $E[F_{UNSTAB}] \leq e^{-\Theta(d)}n$.*

This follows from concentration arguments since $E[\#\pi^{-i}(y) \cap N^{++}(y) - \#\pi^{-i}(\bar{y}) \cap N^{-}(y)] = 0$, $E[\#N^{++}(y) - \#N^{-}(y)] = 0$, and since every variable is expected to appear in at most $O(d)$ clauses.

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

Definition 11. *A variable x is **violated** by α in π if there exists a message $C \rightarrow x$, $C = (\ell_x \vee \ell_y \vee \ell_z)$, in place i in π s.t. one of the following holds:*

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

(b) $|\#\mathbf{1}_\alpha(N^{++}(y)) - \#\mathbf{1}_\alpha(N^{-}(y))| > d/30$

(c) $\#\mathbf{1}_\alpha(N^{s}(y)) < d/7$.

Or one of the above holds for z.

Proposition 12. *Let $\mathcal{F}$ be as in the setting of Theorem 2, and let X be a 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, $E[F_{VIO}] \leq e^{-\Theta(d)}\#X$.*

The proof again uses concentration arguments.

2.2 Dense Subformulas

The next property we discuss is analogous to a property proved in [2] for random graphs. Loosely speaking, [2] prove that **whp** a random graph doesn't contain a small induced subgraph with a large average degree. A similar proposition for 3SAT can also be found in [14].

Proposition 13. *Let $c \gneqq 1$ be an arbitrary constant. Let $\mathcal{F} \in \mathcal{P}^{\text{plant}}_{n,p}$ be as in the setting of Theorem 2. Then **whp** there exists* no *subset of variables U, s.t. $\#U \leq e^{-\Theta(d)}n$ and there are at least $c\#U$ clauses in $\mathcal{F}$ containing two variables from U.*

The proof involves standard first moment calculations, details in the appendix.

2.3 The Core Variables

We describe a subset of the variables, denoted throughout by $\mathcal{H}$ and referred to as the *core variables*, which plays a crucial role in the analysis. Loosely speaking, a variable is considered "safe" if it is stable w.r.t. the initial random order π, and it is not violated by the initial clause-variable message assignments α. If in addition, a safe variable x_i supports many clauses w.r.t. φ (whose corresponding message is '1' in α), then its corresponding B_i value will agree with $\varphi(x_i)$ after the first iteration. This invariant needs to be preserved however in later iterations. The set $\mathcal{H}$ captures the notion of such variables with a self-preserving quality. There are several ways to obtain these desired properties. Formally, $\mathcal{H} = \mathcal{H}(\mathcal{F}, \varphi, \alpha, \pi)$ is constructed using the following iterative procedure:

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.

(a) *Set $H_0 = V \setminus (A_1 \cup A_2 \cup A_3)$.*

(b) *While there exists a variable $a_i \in H_i$ which supports less than $d/4$ clauses in $\mathcal{F}[H_i]$ OR appears in more than $d/30$ clauses* not *in $\mathcal{F}[H_i]$ define $H_{i+1} = H_i \setminus \{a_i\}$.*

(c) *Let a_m be the last variable removed in step 2. Define $\mathcal{H} = H_{m+1}$.*

Proposition 14. *If both α and π are chosen uniformly at random then **whp** $\#\mathcal{H} \geq (1 - e^{-\Theta(d)})n$.*

The main idea of the proof is to observe that to begin with we eliminate very few variables (using the discussion in Section 2.1 to bound $\#A_1 \cup A_2 \cup A_3$). If too many variables were removed in the iterative step then a small but dense subformula exists. Proposition 13 bounds the probability of the latter occurring. Similar arguments can also be found in [2, 14].

2.4 The Factor Graph of the Non-Core Variables

Proposition 14 implies that for $p = c \log n / n^2$, c a sufficiently large constant, **whp** $\mathcal{H}$ contains already all variables. Therefore the following analysis is relevant for the setting of Theorem 2.

Proposition 15. ***Whp*** *every connected component in the factor graph induced by the non-core variables contains $O(\log n)$ variables.*

A proposition of similar flavor was proven in [14] though with respect to a different notion of core. Proposition 15 will not suffice to prove Theorem 2, and we need a further characterization of the non-core factor graph. Using similar techniques to [2] and [14] we prove:

Proposition 16. *With probability $1 - e^{-\Theta(d)}$, there exists no cycle in the factor graph induced by the non-core variables.*

2.5 Outline of Proof of Theorem 2 and Proposition 3

We start with Theorem 2 and derive Proposition 3 as an easy corollary of the analysis. The outline of the proof is as follows. We assume that the formula $\mathcal{F}$ and the run of WP are *typical* in the sense that Propositions 14, 15 and 16 hold. First we prove that after one iteration WP sets the core variables $\mathcal{H}$ correctly (B_i agrees with φ in sign) and this assignment does not change in later iterations. Therefore from iteration 2 and onwards WP is basically running on $\mathcal{F}$ in which variables belonging to $\mathcal{H}$ are substituted with their planted assignment. This subformula is satisfiable and its factor graph is a forest (namely, composed of disjoint trees). Therefore, convergence is guaranteed. The set V_A of Theorem 2 is composed of all variables from $\mathcal{H}$ and those variables from the forest that get assigned. The set V_U is composed of the UNASSIGNED variables from the forest.

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 $z \to C$ agree in sign with their planted assignment (in other words, $C \to x$ is 1 iff x supports C w.r.t. φ).

Proposition 17. *If $x_i \in \mathcal{H}$ and all messages $C \to x_i$, $C \in \mathcal{F}[\mathcal{H}]$ 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.*

Proposition 18. *If $x_i \in \mathcal{H}$ and all messages $C \to x_i$, $C \in \mathcal{F}[\mathcal{H}]$ 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 18 follows immediately from the definition of $\mathcal{H}$ and the message B_i. It suffices to show then that after the first iteration all messages $C \to x_i$, $C \in \mathcal{F}[\mathcal{H}]$ are correct.

Proposition 19. *If $\mathcal{F}$ is a typical instance in the setting of Theorem 2, then after one iteration of WP(F), for every variable $x_i \in \mathcal{H}$, every message $C \to x_i$, $C \in \mathcal{F}[\mathcal{H}]$ is correct.*

Proposition 20. *Let $\mathcal{F}$ be a typical instance in the setting of Theorem 2, then for every variable $x_j \in V \setminus \mathcal{H}$, after $O(\log n)$ iterations either $B_j = 0$ or B_j agrees in sign with $\varphi(x_j)$.*

Proof.(sketch) After one iteration, the core variables signal their correct assignment. Every clause containing a TRUE core literal will constantly signal 0 to all non-core variables, and therefore can be ignored. Every clause containing FALSE core literals can be reduced to a 2-SAT (or a 1-SAT) clause containing only the non-core variables. Therefore after the first iteration, WP basically runs on $\mathcal{F}$ simplified according to the planted assignment of $\mathcal{H}$. Since $\mathcal{F}|_{\varphi(\mathcal{H})}$ is satisfiable, the proposition then follows from Propositions 4, 6, 15 and 16. ■

As for satisfying the set of unassigned variables in time $O(n)$. Propositions 6 and 16 imply that the pure-literal procedure [7] solves the subformula induced by the unassigned variables in linear time. Theorem 2 then follows.

To prove Proposition 3, observe that when $p = c \log n/n^2$, with c a sufficiently large constant, Proposition 14 implies $\mathcal{H} = V$. Combing this with Proposition 19, Proposition 3 readily follows.

3 Discussion

Our results show that WP is effective in solving $\mathcal{P}^{\text{plant}}_{n,p}$. Though not being the first to give an algorithm for $\mathcal{P}^{\text{plant}}_{n,p}$, our results are a first example of rigourously 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 messages are updated in *parallel* in two phases: clause-variable messages updates and then variable-clause messages updates.

We conclude with an open problem. Can our analysis be extended to show that Belief Propagation (BP) finds a satisfying assignment to $\mathcal{P}^{\text{plant}}_{n,p}$ in the setting of Theorem 2? 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 effect the core, a possibility that our analysis managed to exclude for the WP algorithm.

Acknowledgements

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

References

[1] M. Alekhnovich and E. Ben-Sasson. Linear upper bounds for random walk on small density random 3-cnf. In *Proc. 44th IEEE Symp. on Found. of Comp. Science*, page 352, 2003.

[2] N. Alon and N. Kahale. A spectral technique for coloring random 3-colorable graphs. *SIAM J. on Comput.*, 26(6):1733–1748, 1997.

[3] N. Alon, M. Krivelevich, and B. Sudakov. Finding a large hidden clique in a random graph. *Random Structures and Algorithms*, 13(3-4):457–466, 1998.

[4] E. Ben-Sasson, Y. Bilu, and D. Gutfreund. Finding a randomly planted assigment in a random $3CNF$. *manuscript*, 2002.

[5] A. Blum and J. Spencer. Coloring random and semi-random k-colorable graphs. *J. of Algorithms*, 19(2):204–234, 1995.

[6] A. Braunstein, M. Mezard, and R. Zecchina. Survey propagation: an algorithm for satisfiability. *Random Structures and Algorithms*, 27:201–226, 2005.

[7] A. Z. Broder, A. M. Frieze, and E. Upfal. On the satisfiability and maximum satisfiability of random 3-cnf formulas. In *Proc. 4th ACM-SIAM Symp. on Discrete Algorithms*, pages 322–330, 1993.

[8] H. Chen. An algorithm for sat above the threshold. In *6th International Conference on Theory and Applications of Satisfiability Testing*, pages 14–24, 2003.

[9] S. A. Cook. The complexity of theorem-proving procedures. In *Proc. 3rd ACM Symp. on Theory of Computing*, pages 151–158, 1971.

[10] J. M. Crawford and L. D. Auton. Experimental results on the crossover point in random 3-SAT. *Artificial Intelligence*, 81(1-2):31–57, 1996.

[11] O. Dubois, Y. Boufkhad, and J. Mandler. Typical random 3-sat formulae and the satisfiability threshold. In *Proc. 11th ACM-SIAM Symp. on Discrete Algorithms*, pages 126–127, 2000.

[12] U. Feige and R. Krauthgamer. Finding and certifying a large hidden clique in a semirandom graph. *Random Structures and Algorithms*, 16(2):195–208, 2000.

[13] U. Feige and D. Vilenchik. A local search algorithm for 3SAT. Technical report, The Weizmann Institute of Science, 2004.

[14] A. Flaxman. A spectral technique for random satisfiable 3CNF formulas. In *Proc. 14th ACM-SIAM Symp. on Discrete Algorithms*, pages 357–363, 2003.

[15] E. Friedgut. Sharp thresholds of graph properties, and the k-sat problem. *J. Amer. Math. Soc.*, 12(4):1017–1054, 1999.

[16] A. M. Frieze and C. McDiarmid. Algorithmic theory of random graphs. *Random Structures and Algorithms*, 10(1-2):5–42, 1997.

[17] T. G. Gallager. Low-density parity-check codes. *IRE. Trans. Info. Theory*, IT-8:21–28, January 1962.

[18] J. Håstad. Some optimal inapproximability results. *J. ACM*, 48(4):798–859, 2001.

[19] W. Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58:13–30, 1963.

[20] C. Hui and A. M. Frieze. Coloring bipartite hypergraphs. In *Proceedings of the 5th International IPCO Conference on Integer Programming and Combinatorial Optimization*, pages 345–358, London, UK, 1996. Springer-Verlag.

[21] A. C. Kaporis, L. M. Kirousis, and E. G. Lalas. The probabilistic analysis of a greedy satisfiability algorithm. In *Proc. 10th Annual European Symposium on Algorithms*, volume 2461 of *Lecture Notes in Comput. Sci.*, pages 574–585. Springer, Berlin, 2002.

[22] E. Koutsoupias and C. H. Papadimitriou. On the greedy algorithm for satisfiability. *Info. Process. Letters*, 43(1):53–55, 1992.

[23] F. R. Kschischang, B. J. Frey, and H. A. Loeliger. Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory*, 47(2):498–519, 2001.

[24] M. Luby, M. Mitzenmacher, M. A. Shokrollahi, and D. Spielman. Analysis of low density parity check codes and improved designs using irregular graphs. In *Proceedings of the 30th ACM Symposium on Theory of Computing*, pages 249–258, 1998.

[25] M. Luby, M. Mitzenmacher, M. A. Shokrollahi, and D. Spielman. Efficient erasure correcting codes. *IEEE Trans. Info. Theory*, 47:569–584, February 2001.

[26] J. Pearl. *Probabilistic reasoning in intelligent systems: networks of plausible inference.* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1988.

[27] T. Richardson, A. Shokrollahi, and R. Urbanke. Design of capacity-approaching irregular low-density parity check codes. *IEEE Trans. Info. Theory*, 47:619–637, February 2001.

A Proof of Proposition 4

The analysis is divided into three phases: the first r iterations, iteration $r+1$, and the last r iterations (all in all $2r+1$ iterations). Each phase plays a different role in the analysis. Recall the definitions in Section 1.5 regarding pure variables. Formally we prove,

Lemma 21. *Let r be s.t. $\mathcal{F}$ is r-pure and $P = \bigcup_{i \le r} P_i$. Then after at most r iterations of the* WP *algorithm and for the rest of the execution, regardless of the initial messages, the order of execution, or if the execution is sequential or parallel, for every variable $x \in P$ all messages $x \to C$ are either 0, positive if $x \in P_i$ appears only positively in L_i, or negative if x appears only negatively in L_i.*

Proof. The lemma is easily proven by induction on the P_i's. The base case, $i = 0$, is immediate from the definition of a pure variable and the definition of the message $x \to C$. Suppose the lemma holds for $\bigcup_{j<i} P_j$ and consider the i'th iteration of the WP. Let $x \in P_i$, and w.l.o.g. assume x is a pure positive literal in L_i. Therefore, every clause C_j containing $\bar{x}$ (if such exists) also contains a variable from $\bigcup_{j<i} P_j$ in its pure polarity. By the induction hypothesis, this variable either signals '0' to C_j or notifies it that it will be assigned in a satisfying manner. Therefore the message $C_j \to \bar{x}$ is surely 0. Thus, looking at the definition of the message $x \to C$ one immediately sees that the latter implies $x \to C \ge 0$. ■

Lemma 22. *In the setting of Lemma 21, in iteration at most $r+1$ and for the rest of the execution, for all non-pure variables $x \in V \setminus P$, all messages $C \to x$ equal 0.*

The lemma is an immediate corollary of Lemma 21 and observing that every clause containing a non-pure variable must also contain at least one variable $x_i \in P_j$ in its pure form.

Lemma 23. *In the setting of Lemma 22, after additional r iterations, for every pure variable $x \in P_i$, either $C \to x$ is 1 (and stays so for the rest of the execution), in which case x appears in its pure from in C and in* all *satisfying assignments x satisfies C. Or $C \to x = 0$ (and stays so for the rest of the execution).*

Proof. The lemma is proven by a reverse induction on the P_i's. For the base case, $i = r$, consider $x \in P_r$ and a message $C \to x$. If $C = (\ell_x)$ then always $C \to x = 1$ by definition and indeed in all satisfying assignments x satisfies C. If C contains a non-pure variable then Lemma 22 implies $C \to x = 0$, which stays that way for the rest of the execution. If C contains only pure variables, then it cannot be the case that the other pure variables appear in polarity which is opposite their pure one (otherwise, since r is the last level they cannot be pure). Therefore, Lemma 21 implies that $C \to x = 0$. For the inductive step, consider a message $C \to x$, $x \in P_i$. Again if $C = (\ell_x)$ or contains non-pure variables we are done. If C contains only pure variables, then the only case $C \to x = 1$ is if the other pure variable(s), z, appears in opposite polarity to its pure one, and z received a warning. In this case $z \in \bigcup_{j>i} P_j$, and by the induction hypothesis in all satisfying assignments z will evaluate to FALSE in C, therefore x must satisfy C. In either case the message $C \to x$ will not change. ■

In the case $\mathcal{F}$ contains no unit clauses, Lemmas 22 and 23 show that every message $C \to x$ is 0, therefore all B_i's are 0 (and the corresponding assignment is the all-UNASSIGNED vector).

B Proof of Proposition 6

Let $\mathcal{F}$ be a satisfiable CNF. It suffices to prove that if the factor graph corresponding to $\mathcal{F}$ is a tree then $\mathcal{F}$ contains at least one pure variable (if so, remove this variable and all clauses it appears in, the factor graph remains a forest, and repeat the argument).

Lemma 24. *Let $\mathcal{F}$ be a satisfiable CNF whose factor graph is a tree, then $\mathcal{F}$ contains at least one pure variable.*

Proof. If one of the tree leaves is a variable, then it is pure (as it is contained in only one clause). Otherwise, there exists a unit clause C_0 which is a leaf, w.l.o.g. assume $C_0 = (x_0)$. If $\mathcal{F}$ contains a pure variable we are done. Otherwise, there exists a clause C_1 containing $\bar{x}_0$ ($C_1 \neq C_0$ as $\bar{x}_0 \notin C_0$). Let φ be a satisfying assignment to $\mathcal{F}$. It must be that $\varphi(x_0) = TRUE$, therefore C_1 must contain at least one additional literal, say x_1 s.t. $\varphi(x_1) = TRUE$ (C_1 cannot contain x_0 as we assume $\mathcal{F}$ contains no clauses of the form $(x \vee \bar{x})$). Since x_1 is not pure either, there must exist a clause $C_2 \neq C_1$ containing $\bar{x_1}$, and another satisfying literal z. If $z \in \{x_0\}$ we found a cycle which contradicts the assumption. Therefore $z = x_2$. Since the number of variables is at most n, eventually we will run out of new variables and find a cycle, contradicting our assumption. ■

C Proof of Proposition 10

Consider a clause-variable message edge $C \to x$ in location i in π, $C = (\ell_x \vee \ell_y \vee \ell_z)$. Now consider location $j \leq i$. The probability of an edge $C' \to \bar{y}$ in location j is $\left(3\binom{n}{2}\right) / \left(3 \cdot 7\binom{n}{3}\right) = \frac{3}{7n} + O(\frac{1}{n})$ which is exactly the probability of an edge $C'' \to y$, $C'' \in N^{++}(y)$. This implies

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

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 success in draws without replacement. It is known that this quantity is more concentrated than the corresponding quantity if the draws were made with replacement [19]. 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 mean by more than $d/60$ is $e^{-\Theta(d)}$. A similar statement holds for $\#\pi^{-i}(\bar{y}) \cap N^{-}(y)$. The proof follows.

Properties (b) and (c) 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 standard concentration results). To sum up,

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

The proposition follows by linearity of expectation.

D Proof of Proposition 12

Consider a stable variable x in $\mathcal{F}$ 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

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

If however

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

then at least one of the quantities in (D.1) deviated from its expectation by at least $(d/30-d/60)/2$. 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 (b) and (c) are bounded similarly using tight concentration results. Using the union bound as in the proof of Proposition 10 and the linearity of expectation the proposition follows.

E Proof of Proposition 13

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. $\frac{d}{n^2}$. 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.

F Proof of Proposition 14

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

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

Propositions 8, 10, and 12 and Azuma's inequality for example 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 $\{a_1, a_2, ..., a_{i-1}\} \cup A_1 \cup A_2 \cup A_3$, 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 13, (b) is bounded.

G Proof of Proposition 16

Every k-cycle in the factor graph consists of k variables, w.l.o.g. say $x_1, ..., x_k$, and k clauses, $C_1, ..., C_k$, s.t. $x_i, x_{i+1} \in C_i$. Let C be a fixed k-cycle and let $F_C \subseteq \mathcal{F}$ be a fixed minimal set of clauses inducing C. Let $V(C)$ be the variables in C. In order for C to belong to the factor graph induced by non-core variables it must be that $F_C \subseteq F$ and that $V(C) \subseteq \bar{\mathcal{H}}$ (put differently, $V(C) \cap \mathcal{H} = \emptyset$). Let us start by bounding $Pr[F_C \subseteq \mathcal{F} \bigwedge V(C) \cap \mathcal{H} = \emptyset]$ and then use the union bound over all possible cycles and inducing sets of clauses. As the two events are not independent, the calculations are more involved. Loosely speaking, to circumvent the dependency issue, one needs to defuse the effect the event $\{F_C \subseteq \mathcal{F}\}$ might have on $\mathcal{H}$. To this end we introduce a set $\mathcal{H}'$, defined very similarly to $\mathcal{H}$ only "cushioned" in some sense to overcome the dependency issues. This is done using similar techniques to [2] and [14].

We start by defining the new set of core variables $\mathcal{H}'$ (again w.r.t. an ordering π of the clause-variable messages and an initial values vector α). The changes are highlighted in bold.
Propositions 10 and 12 could be easily adjusted to accommodate the 6-gap in the new definition in B_2 and B_3. Therefore Proposition 14 can be safely restated in the context of $\mathcal{H}'$.

Proposition 25. *If the both α and π are chosen uniformly at random then **whp** $\#\mathcal{H}' \geq (1 - e^{-\Theta(d)})n$.*

Proposition 26. *The set J defined above satisfies $\#J \geq k/2$*

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 9 to be **(d/30-6)** in (a) and (b).*
*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 11 to be **(d/30-6)** in (a) and (b).*
Let $\boldsymbol{J} \subseteq V(F_C)$ be the set of variables appearing in no more than 6 different clauses in F_C

(a) *Set $H'_0 = V \setminus (B_1 \cup B_2 \cup B_3 \cup (V(F_C) \setminus J))$.*

(b) *While there exists a variable $a_i \in H'_i$ which supports less than $d/4$ clauses in $\mathcal{F}[H'_i]$ OR appears in more than **(d/30-6)** clauses* not *in $\mathcal{F}[H'_i]$, define $H'_{i+1} = H'_i \setminus \{a_i\}$.*

(c) *Let a_m be the last variable removed at step 2. Define $\mathcal{H}' = H'_{m+1}$.*

Proof. Suppose $\#J < k/2$, then there are more than $k/2$ variables in $V(F_C)$, each appearing in at least 6 different clauses in F_C. Thus, $\#F_C > (6 \cdot k/2)/3 = k$ (we divided by three as every clause might have been counted 3 times), contradicting $\#F_C = k$. ∎

The following proposition allows us to bound the event of a "bad" cycle.

Proposition 27. *$Pr[F_C \subseteq \mathcal{F} \bigwedge V(C) \cap \mathcal{H} = \emptyset] \leq Pr[F_C \subseteq \mathcal{F}] \cdot Pr[J \cap \mathcal{H}' = \emptyset]$.*

Corollary 28. *$Pr[F_C \subseteq \mathcal{F} \bigwedge V(C) \cap \mathcal{H} = \emptyset] \leq (d/n^2)^k \cdot \lambda^{k/2}$, where $\lambda = 1 - \#\mathcal{H}'/n$*

Proof.(Corollary 28) As the clauses are chosen independently,

$$Pr[F_C \subseteq \mathcal{F}] \leq (d/n^2)^k.$$

Since C is a fixed cycle and the choice of $\mathcal{H}'$ is independent of J, it follows that

$$Pr[J \cap \mathcal{H}' = \emptyset] \leq \frac{\binom{n-\#\mathcal{H}'}{k/2}}{\binom{n}{k/2}} = \frac{\binom{\lambda n}{k/2}}{\binom{n}{k/2}} \leq \lambda^{k/2}.$$

The last inequality follows from standard bounds on the binomial coefficients. ∎

Corollary 29. *The probability of a cycle in the factor graph induced by $\bar{\mathcal{H}}$ is at most $e^{-\Theta(d)}$.*

Proof. Let us bound the number of cycles of length k. There are $\binom{n}{k}$ ways to choose the variables inducing the cycles, and $k!/2$ ways to order them on the cycle. As for F_C, once the cycle is fixed, we have at most $(7n)^k$ ways of choosing the third variable and setting the polarity in every clause. Using the union bound, the probability of a cycle in the non-core factor graph is at most

$$\sum_{k=1}^{\lambda n} \binom{n}{k} \cdot k! \cdot (7n)^k \left(\frac{d}{n^2}\right)^k \cdot \lambda^{k/2} \leq \sum_{k=1}^{\lambda n} \left(\frac{7en}{k}\right)^k \cdot k^k \cdot n^k \cdot \left(\frac{d}{n^2}\right)^k \cdot \lambda^{k/2} = \sum_{k=1}^{\lambda n} (7e \cdot d \cdot \sqrt{\lambda})^k$$

Assuming that Proposition 25 holds (which is the case **whp**), then $\lambda = e^{-\Theta(d)}$ and $7e \cdot d \cdot \sqrt{e^{-\Theta(d)}}$ is much smaller than 1. In particular, the last summation is simply the sum of a decreasing geometric series with quotient smaller than say 1/2, which sums up to at most $14e \cdot d \cdot \sqrt{e^{-\Theta(d)}} = e^{-\Theta(d)}$. ∎

Corollary 29 proves Proposition 16. It remains to prove Proposition 27. The key to proving Proposition 27 is in the following inclusion.

Lemma 30. *For every cycle C and every minimal inducing set F_C, $\mathcal{H}'(\mathcal{F},\varphi,\alpha,\pi) \subseteq \mathcal{H}(\mathcal{F} \cup F_C,\varphi,\alpha,\pi)$.*

This lemma clarifies the motivation for defining $\mathcal{H}'$. It is not necessarily true that $\mathcal{H}(F) \subseteq \mathcal{H}(F \cup F_C)$. For example, a variable which appears in $\mathcal{H}(F)$ could disappear from $\mathcal{H}(F \cup F_C)$ since the clauses in F_C make it unstable. Loosely speaking, $\mathcal{H}'$ helps overcome this problem.

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

Proof.(Proposition 27)

$$Pr[F_C \subseteq \mathcal{F} \bigwedge V(C) \cap \mathcal{H} = \emptyset] \leq Pr[F_C \subseteq \mathcal{F} \bigwedge J \cap \mathcal{H} = \emptyset] = Pr[J \cap \mathcal{H} = \emptyset | F_C \subseteq \mathcal{F}] Pr[F_C \subseteq \mathcal{F}].$$

Therefore, it suffices to prove

$$Pr[J \cap \mathcal{H} = \emptyset | F_C \subseteq \mathcal{F}] \leq Pr[J \cap \mathcal{H}' = \emptyset].$$

$$Pr[J \cap \mathcal{H}' = \emptyset] = \sum_{F : J \cap \mathcal{H}'(F) = \emptyset} Pr[\mathcal{F} = F] \underbrace{\geq}_{\mathcal{H}'(F) \subseteq \mathcal{H}(F \cup F_C)} \sum_{F : J \cap \mathcal{H}(F \cup F_C) = \emptyset} Pr[\mathcal{F} = F]$$

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

$$\sum_{F' : F' \cap F_C = \emptyset, J \cap \mathcal{H}(F' \cup F_C) = \emptyset} \sum_{F'' : F'' \subseteq F_C} Pr[\mathcal{F} \setminus F_C = F' \wedge \mathcal{F} \cap F_C = F'']$$

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

$$\sum_{F' : F' \cap F_C = \emptyset, J \cap \mathcal{H}(F' \cup F_C) = \emptyset} \sum_{F'' : F'' \subseteq F_C} Pr[\mathcal{F} \setminus F_C = F'] Pr[\mathcal{F} \cap F_C = F''] =$$

$$\sum_{F' : F' \cap F_C = \emptyset, J \cap \mathcal{H}(F' \cup F_C) = \emptyset} Pr[\mathcal{F} \setminus F_C = F'] \underbrace{\sum_{F'' : F'' \subseteq F_C} Pr[\mathcal{F} \cap F_C = F'']}_{1} =$$

$$\sum_{F' : F' \cap F_C = \emptyset, J \cap \mathcal{H}(F' \cup F_C) = \emptyset} Pr[\mathcal{F} \setminus F_C = F']$$

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

$$\sum_{F' : F' \cap F_C = \emptyset, J \cap \mathcal{H}(F' \cup F_C) = \emptyset} Pr[\mathcal{F} \setminus F_C = F' | F_C \subseteq \mathcal{F}] = Pr[J \cap \mathcal{H} = \emptyset | F_C \subseteq \mathcal{F}].$$

■

H Proof of Proposition 17

Proof. By contradiction, let $C_0 \to x$ be the first wrongly evaluated message in the iteration. W.l.o.g. assume $C_0 = (\ell_x \vee \ell_y \vee \ell_z)$. 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 $C'' \to y$, $C'' \in F[\mathcal{H}] \cap N^{++}(y) \cap N^-(y)$ 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, $y \in \mathcal{H}$ and therefore it supports at least $d/4$ clauses in $\mathcal{F}[\mathcal{H}]$. 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. ■

I Proof of Proposition 19

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

$$y \to C \geq \underbrace{d/7}_{\text{property (c) in defn. 11}} - \underbrace{d/30}_{\text{property } (b) \text{ in defn. 11}} - \underbrace{d/30}_{\text{non-core messages}} > d/14.$$

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 $C' \in N^{++}(y) \cap N^-(y) \cap \mathcal{F}[\mathcal{H}]$ then its value is '0' by the induction hypothesis. Since x is not violated by α, property (b) in definition 11 ensures that to begin with $|\#\mathbf{1}_\alpha(N^{++}(y)) - \#\mathbf{1}_\alpha(N^-(y))| \leq d/30$. $y \in \mathcal{H}$, 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 (a) in definition 11 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 (c) in definition 11 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. ■