Message passing for the coloring problem: Gallager meets Alon and Kahale

Sonny Ben-Shimon* Dan Vilenchik†

May 10, 2007

Abstract

Message passing algorithms are popular in many combinatorial optimization problems. For example, experimental results show that *survey propagation* (a certain message passing algorithm) is effective in finding proper k-colorings of random graphs in the near-threshold regime. Gallager in 1962 introduced the concept of Low Density Parity Check (LDPC) codes, and suggested a simple decoding algorithm based on message passing. In 1994 Alon and Kahale exhibited a coloring algorithm and proved its usefulness for finding a k-coloring of graphs drawn from a certain planted-solution distribution over k-colorable graphs. In this work we show an interpretation of Alon and Kahale's coloring algorithm in light of Gallager's decoding algorithm, thus showing a connection between the two problems - coloring and decoding. This also provides a rigorous evidence for the usefulness of the message passing paradigm for the graph coloring problem.

1 Introduction and Results

A *k-coloring* f of a graph $G = (V, E)$ is a mapping from the set of vertices V to $\{1, 2, ..., k\}$. f is a *proper coloring* of G if for every edge $(u, v) \in E$, $f(u) \neq f(v)$. In the graph coloring problem we are given a graph $G = (V, E)$ and are asked to produce a proper k-coloring f with a minimal possible k. The minimal value of k is called the *chromatic number* of the graph G, commonly denoted by $\chi(G)$. The problem of properly k-coloring a k-colorable graph is notoriously hard. It is one of the most famous NP-Complete problems, and even approximating the chromatic number within an acceptable ratio is NP-hard [7].

Message passing algorithms are popular in many combinatorial optimization problems. For example, experimental results show that *survey propagation* (a certain message passing algorithm) is effective in finding proper k-colorings of random graphs in the near-threshold regime [5]. In his seminal work [14], Gallager in 1962 introduced the concept of Low Density Parity Check (LDPC) codes, and suggested a simple decoding algorithm based on message passing.

In 1994 Alon and Kahale [1] introduced, in another innovative work, a spectral algorithm for coloring sparse ("low density") k-colorable graphs, and showed that their algorithm works *whp*[1] over a certain planted-solution distribution over k-colorable graphs on n vertices.

1.1 Our Contribution

In this paper we connect the two latter results. Specifically, we show that Alon and Kahale's coloring algorithm implicitly contains Gallager's decoding algorithm, thus asserting an interesting relation

*Email: sonny@tau.ac.il.
†Email: vilenchi@tau.ac.il.

[1]By writing *whp* we mean with high probability, i.e., with probability tending to 1 as n goes to infinity.

between the two problems – coloring and decoding. Theorem **??** makes this notion formal. Hopefully, following this insight, other heuristics that are useful in decoding algorithms can be applied to the coloring problem; we are not aware of previous works pinpointing a connection between the two problems. Our result also gives a rigorous analysis of a message passing algorithm in the context of the colorability problem, while up until now only experimental results showed the usefulness of the message passing paradigm.

Our result works in the other direction as well. LDPC codes drew the attention of many researchers ever since introduced by Gallager in 1962. The problem of graph coloring can be viewed as a specific instance of the decoding problem – the codeword being a proper k-coloring of the graph (assuming there is a single one). Therefore, the analysis of Gallager's algorithm that we provide here can be useful for the analysis of Gallager's algorithm on random LDPC codes. Specifically, the study of LDPC codes is concerned with two main problems. The first is designing "good" LDPC codes based on expander graphs and study their decoding efficiency close to the code's rate (which usually depends on the expansion properties of the graph), for example [20, 16]. These works show that whenever the expander is "good", one can decode a noisy codeword of length n close to the code's rate, leaving at most $o(n)$ errors. The second problem concerns the efficiency of decoding a noisy codeword with error-ratio some fraction of the code's rate, for example [22, 17]. In these works it is shown that whenever the expander is "good" and the codeword doesn't have too many errors then typically one can reconstruct the entire codeword (for example, the rate-1/2 regular codes of Gallager can provably correct up to 5.17% errors, or the irregular codes in [17] with 6.27%).

Our result concerns the second problem. In our setting, the input graph G (sampled from the planted distribution over k-colorable graphs, to be defined shortly) on which we analyze Gallager's algorithm is typically *not* an expander, though it contains a large subgraph which is an expander. Loosely speaking, we are able to show that typically Gallager's algorithm converges "quickly" on G when starting from a noisy coloring which differs from a proper one on the color of say $n/20$ (5%) vertices to a proper k-coloring of G (up to a small number of vertices which remain "undecided" and whose coloring can be completed efficiently). Thus we are able to analyze Gallager's algorithm while relaxing the demand of expansion for the entire graph, and replacing it with a more modest requirement. Our result should be applicable to the analogous LDPC setting, and in particular to random LDPC codes similar to the ones introduced by Gallager himself. Thus, in turn, Alon and Kahale's coloring algorithm (interpreted in Gallager's spirit) inspires a simple decoding algorithm for noisy codewords with error rate by some constat smaller than the code's rate.

1.2 The Planted Graph Coloring Distribution

The model we analyze was suggested by Kučera [15] as a model for generating random k-colorable graphs with an arbitrary average vertex-degree, denoted throughout by $\mathcal{G}^{\text{plant}}_{n,p,k}$. First, randomly partition the vertex set $V = \{1, ..., n\}$ into k classes $V_1, ..., V_k$, of size n/k each. Then, for every $i \neq j$, include every possible edge connecting a vertex in V_i with a vertex in V_j with probability $p = p(n)$. Throughout, $\varphi^* : V \to \{1 \dots, k\}$ denotes the planted coloring, where the corresponding graph will be clear from context. $\mathcal{G}^{\text{plant}}_{n,p,k}$ is also referred to as the planted distribution, and is the analog of the planted clique, planted bisection, and planted SAT distributions, studied e.g. in [2, 8, 11, 12].

$\mathcal{G}^{\text{plant}}_{n,p,k}$ is similar to LDPC in many ways. Both constructions are based on random 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 $\mathcal{G}^{\text{plant}}_{n,p,k}$ the adjacency matrix of the graph contains noisy information about the planted coloring – embedded in eigenvectors corresponding to the $(k-1)$ smallest eigenvalues.

1.3 Our Result

To avoid a cumbersome presentation we state the results for the case $k = 3$, and point out that the result is easily extended to any fixed k. We call a k-coloring φ *partial* if some vertices in φ are UNASSIGNED (that is, are left uncolored).

Theorem 1.1 *Fix $\varepsilon \leq \frac{1}{100}$ and let G be a random graph in $\mathcal{G}^{\text{plant}}_{n,p,3}$, $np \geq C_0$, $C_0 = C_0(\varepsilon)$ a sufficiently large constan. Then running* Gallager *on G, starting from φ, some 3-coloring at distance at most εn from φ^*, satisfies the following with probability $1 - e^{-\Theta(np)}$:*

1. Gallager(G, φ) *converges to a partial coloring φ' after at most $O(\log n)$ iterations.*
2. *Let $V_A = \{v \in V : \varphi'(v) = \varphi^*(v)\}$, then $|V_A| \geq (1 - e^{-\Theta(np)})n$. Furthermore, all vertices in $V \setminus V_A$ are UNASSIGNED.*
3. *$G[V \setminus V_A]$ can be properly colored (extending the proper coloring of $G[V_A]$) in time $O(n)$.*

Remark 1.2 *In item 2, if $np \geq C \log n$ for a sufficiently large constant C, then whp $V_A = V$, namely all the vertices are properly colored, and item 3 becomes void.*

The algorithm Gallager and its adaptation to the coloring problem are given in Section 2 ahead. Alon and Kahale's algorithm is also given in Section 2.4 for the sake of completeness. For simplicity, we identify with "Gallager" both the decoding algorithm and its adaptation to the colorability problem, while it will be clear from context which setting is regarded.

Comparing our results with the coding setting, known results show that Gallager, applied to carefully designed LDPC codes, typically recovers the entire codeword (if the noisy codeword has error rate below the code's rate) or all but $o(1)$-fraction of it (when close to the rate). In our case, Gallager colors correctly ("decodes") all but a constant fraction of the vertices – $e^{-\Theta(np)}n$. This is optimal in some sense as *whp* there will be $e^{-\Theta(np)}n$ isolated vertices in G, or vertices with very low degrees, which can actually take more than one color and still the graph remains k-colorable. Nevertheless, in both the coloring setting and the decoding one it is shown that the quality of decoding/coloring improves exponentially with the number of iterations.

The main difference between the distribution we consider, $\mathcal{G}^{\text{plant}}_{n,p,3}$ and constructions of LDPC codes is the fact that we do not require the graph to be an expander and we do not "engineer" the graph so that Gallager works well for it. Furthermore, in the coloring setting we prove that Gallager either colors a vertex correctly or leaves it UNASSIGNED. In the coding setting the situation is different. The $o(1)$-fraction of the bits that Gallager fails to decode are set wrongly and their location is not known. Therefore our result is stronger in the sense that the graph is not required to be an expander, and there are no wrongly colored vertices when Gallager terminates.

Our result is also comparable with the work of [9] where the Warning Propagation message passing algorithm was analyzed for the satisfiability problem. Though Warning Propagation cannot be viewed as an adaptation of Gallager's algorithm to the satisfiability problem, the two algorithms are similar in the sense that they both involve discrete and rather simple (natural) messages. The result in [9] is very similar in nature to the one we obtain, though the initial satisfying assignment can be a random one (due to the inherent break of symmetry that SAT possesses – variables can appear positively or negated).

1.4 Paper's Structure

The remaining of the paper is structured as follows. In Section 2 we present Gallager's decoding algorithm – Gallager – and it's adaptation for the colorability problem. We also present Alon and

Kahale's coloring algorithm and show how it contains Gallager, thus justifying the title of this paper. In Section 3 we discuss some properties that a typical instance in $\mathcal{G}^{\text{plant}}_{n,p,3}$ possesses. These properties come handy when proving Theorem 1.1 in Section 2.5. Concluding remarks are given in Section 5.

2 Gallager's Decoding Algorithm

Before presenting Gallager we give a short introduction to linear codes and their graphical representation.

2.1 Graphical Representation of Codes

A linear code $\mathcal{C}$ of length n and dimension r is defined by a matrix $H \in \{0,1\}^{r\times n}$; a vector $c = (c_1,\ldots,c_n) \in \{0,1\}^n$ is a *codeword* ($c \in \mathcal{C}$) if $Hc \equiv 0 \pmod 2$. The matrix H can be viewed as the incidence matrix of some bipartite graph $\mathcal{B}[H]$ with n nodes on the left, $\{x_1,\ldots,x_n\}$, and r nodes on the right, $\{C_1,\ldots,C_r\}$. Every left-side node corresponds to a different bit of the message (we refer to x_i as a the i^{th} *variable*), and every right-side node represents a *constraint* (we refer to C_j as the j^{th} *constraint*). We add the edge (x_i, C_j) if and only if $H_{i,j} = 1$. Every constraint C_j involves the variables in $N(C_j) = \{x_i : (x_i, C_j) \in E(\mathcal{B}[H])\}$, i.e. the neighbor set of C_j. The code $\mathcal{C}$ can now be defined equivalently in terms of the bipartite graph $\mathcal{B}[H]$ in the following way: $c \in \{0,1\}^n$ belongs to $\mathcal{C}$ if when assigning $x_i = c_i$, then for every constraint C_j the exclusive-or of the bits in $N_{\mathcal{B}[H]}(C_j)$ is 0. If every variable appears in exactly s constraints, and every constraint involves exactly t variables, then the bipartite graph is called (s,t)-regular (and so is the corresponding code). If t is constant, that is – every constraint involves at most a constant number of variables, then the code is called low-density (LDPC).

2.2 Gallager's Algorithm

Gallager's decoding algorithm is a message passing algorithm defined on the bipartite graph $\mathcal{B}[H]$, which is also referred to as the *factor graph* in the context of message passing algorithms. Gallager's algorithm is an example of hard decision decoding, which signifies the fact that at each step the messages are derived from local decisions of whether each bit is 0 or 1, and this is all the information the message contains (as opposed to more detailed probabilistic information). We note that Gallager also proposes a belief propagation type decoding algorithm, which uses a more complicated message set.

There are two types of messages associated with every edge (x_i, C_j) of $\mathcal{B}[H]$. A message from a constraint C_j to a variable x_i and vice-versa. Intuitively, we think of the message $x_i \to C_j$ as some sort of a majority vote over the messages $C_{j'} \to x_i$ (for $j \neq j'$). Similarly, C_j sends x_i the "preferred" value for x_i – if the exclusive-or, $\oplus$, without x_i is $b \in \{0,1\}$, then to keep the constraint satisfied, x_i should take the value b.

Formally, let $\tau \geq 0$ be some fixed integer and $\alpha = (\alpha_1, \alpha_2, ..., \alpha_n)$ the received codeword. Set $\mathcal{C}_j^{i,b} = \{C_{j'} : C_{j'} \to x_i = b \wedge j' \neq j\}$ for $b \in \{0,1\}$.

$$x_i \to C_j = \begin{cases} b, & \left|\mathcal{C}_j^{i,b}\right| \geq \tau \\ \alpha_i, & \text{otherwise} \end{cases}$$

$$C_j \to x_i = \bigoplus_{x \in N_{\mathcal{B}[H]}(C_j)\setminus\{x_i\}} x \to C_j$$

Figure 1: Gallager's messages

It is convenient to define a third message which is not sent during the algorithm, but is used to calculate the final decoding. For every x_i define

$$B_i = \begin{cases} b, & |\{C : C \to x_i = b\}| \geq \tau \\ \alpha_i, & \text{otherwise} \end{cases}$$

We are now ready to present Gallager.

Gallager(α, τ):

1. for every edge (x_i, C_j) : set $x_i \to C_j = \alpha_i$
2. repeat until no message changed:
   (a) update in parallel all messages $C_j \to x_i$.
   (b) update in parallel all messages $x_i \to C_j$.
3. return for every i, $x_i = B_i$.

Figure 2: Gallager's algorithm

The algorithm is allowed not to terminate (the messages may keep changing every iteration). If the algorithm does terminate, then we say that it *converges*. In practice, it is common to make an a-priori limit on the number of iterations, and return failure if the algorithm does not converge within the given limit. It should be noted that Gallager is often described with τ being the size of the constraint minus 1.

2.3 Gallager for Colorability

Given a k-colorable graph G one can define its factor graph $\mathcal{B}[G]$. The constraints (right hand side nodes) are the edges of G, and the left-hand nodes, the variables, are the vertices. For $x_i \in V(G)$, and $C_j \in E(G)$ we add the edge (x_i, C_j) if $x_i \in C_j$. The vector $c \in \{1, \ldots, k\}^{|V(G)|}$ is a proper k-coloring of the graph if the exclusive-or of each constraint (edge) is *non*-zero (where the exclusive-or of two integers is the bitwise exclusive-or of their binary representation). The factor graph of the coloring problem is a special case of the LDPC factor graph in which every constraint involves exactly two variables.

We suggest the following interpretation of the messages in Figure 1 for the coloring setting. Fix some constant τ, and set as before $\mathcal{C}_j^{i,b} = \{C_{j'} : C_{j'} \to x_i = b \wedge j' \neq j\}$, $b \in \{1, \ldots, k\}$. We let argmin denote the index of the minimal elements of the given set.

$$x_i \to C_j = \begin{cases} \text{UNDECIDED}, & \exists l, m \text{ s.t. } l \neq m \text{ and } \left|\mathcal{C}_j^{i,l}\right|, \left|\mathcal{C}_j^{i,m}\right| < \tau; \\ l, & l = \text{argmin}_t \left|\mathcal{C}_j^{i,t}\right|, \end{cases}$$

$$C_j \to x_i = x_k \to C_j.$$

Figure 3: Gallager's messages for colorability

Though at first glance the messages may appear different than the ones in Figure 1, this is only due to the fact that the coloring setting is not a binary one. x_i sends C_j the minority vote over the colors it receives from the other constrains (edges) it appears in (or UNDECIDED if there weren't enough votes to begin with). The constraint C_j sends the variable x_i which color it must'nt take.

To see the similarity to the coding setting, observe that a congruent set of messages for Gallager's algorithm would be for a variable to send the least popular bit amongst the messages it received, and for a constraint to send the value which the variable should not take. The constraint to variable message, $C_j \to x_i$, is the same in both settings since in the coloring problem the exclusive-or is taken over one item (every constraint – edge in this case – contains exactly two variables), and therefore it is the item itself.

Gallager can now be used for the coloring problem, with the input $\alpha = (\varphi(x_1), \ldots, \varphi(x_n))$ being the (not necessarily proper) k-coloring vector of the vertices, and the B_i's defined by

$$B_i = \begin{cases} \text{UNDECIDED}, & \exists l, m \text{ s.t. } l \neq m \text{ and } |\{C_j : C_j \to x_i = l\}|, |\{C_j : C_j \to x_i = m\}| < \tau \\ l, & l = \text{argmin}_t \, |\{C_j : C_j \to x_i = t\}| \end{cases} \tag{2.1}$$

2.4 Alon and Kahale's coloring algorithm

The seminal work of Alon and Kahale [1] paved the road towards dealing with large constant-degree planted distributions. They present an algorithm that *whp* 3-colors a random graph in $\mathcal{G}^{\text{plant}}_{n,p,3}$, where $np \geq C_0$ and C_0 is a sufficiently large constant. Since our result refers to their algorithm we give a rough outline of it here, and refer the interested reader to [1] for the complete details. The algorithm is denoted throughout by Alon − Kahale.

Alon-Kahale(G, k):
step 1: spectral approximation.
1. obtain an initial k-coloring of the graph using spectral methods.
step 2: recoloring procedure.
2. for $i = 1$ to $\log n$ do:
  2.a for all $v \in V$ simultaneously color v with the least popular color amongst its neighbors.
step 3: uncoloring procedure.
3. while $\exists v \in V$ with less than $np/10$ neighbors colored in some other color do:
  3.a uncolor v.
step 4: Exhaustive Search.
4. let $U \subseteq V$be the set of uncolored vertices.
5. consider the graph $G[U]$.
  5.a if there exists a connected component of size at least $\log n$ - fail.
  5.b otherwise, exhaustively extend the coloring of $V \setminus U$ to $G[U]$.

Figure 4: Alon and Kahale's coloring algorithm

The algorithm is composed of three main steps. First using spectral techniques one typically obtains a k-coloring that agrees with the planted one on many vertices (say $0.99n$). Then follows a refining procedure (step 2,3) which terminates typically with a partial k-coloring that coincides with the planted coloring on the colored vertices. This coloring typically colors all but $e^{-\Theta(np)}n$ vertices. Finally, in step 4, the graph induced by the uncolored vertices is exhaustively searched.

2.5 Gallager meets Alon-Kahale

We now show how Alon − Kahale actually runs Gallager. This observation is of course not made in [1], nor the connection to message passing whatsoever.

We first claim that one can unify the recoloring and unassignment steps of Alon − Kahale (steps 2 and 3) as follows. In the recoloring step, assign each vertex with the least popular color amongst its neighbors, or set it UNDECIDED if it has less than τ neighbors colored in some color other than

its own (τ is some fixed integer, say 2). Using the same arguments as in [1] one can prove that the algorithm performs the same with the unified recoloring-uncoloring scheme.

Now observe that the unified step of the new $\mathsf{Alon}-\mathsf{Kahale}$ is exactly $\mathsf{Gallager}$ (with the messages as in Figure 3) since the message B_i (2.1) which implies the color of v_i reads exactly this.

3 Properties of a Random $\mathcal{G}^{\text{plant}}_{n,p,3}$ Instance

In this section we introduce some properties of a typical graph sampled from $\mathcal{G}^{\text{plant}}_{n,p,3}$. These properties will come handy when proving Theorem 1.1 in the next section. Properties of similar flavor can be found in [1]. For a subset U of vertices, $e(U)$ denotes the number of edges spanned by the vertices of U. For lack of space the propositions are given without complete proofs, whose main details can be found in [1].

The first property we discuss is discrepancy. Specifically, a random graph *whp* will not contain a small yet unexpectedly dense subgraph. Formally,

Proposition 3.1 *Let G be a graph distributed according to $\mathcal{G}^{\text{plant}}_{n,p,3}$ with $np \geq C_0$, C_0 a sufficiently large constant. Then whp there exists* no *subgraph of G containing at most $n/60$ vertices whose average degree is at least $np/10$.*

The next property we discussed was introduced in [1] and plays a crucial role in the analysis of the algorithm. Loosely speaking, a vertex v is considered "safe" w.r.t. φ^* if v has many neighbors from every color class of φ^* other than its own, and these neighbors are also "safe" w.r.t. φ^*. v is "safe" in the sense that if it converged to a wrong color (w.r.t. φ^*) when the algorithm terminates, then it has many wrongly-colored neighbors w.r.t. φ^*, and so do they. This avalanche effect of many wrongly-colored vertices is, under certain conditions, very unlikely to happen. Therefore, typically the "safe" vertices converge correctly. The following definition makes this notion formal.

Definition 3.2 *A set of vertices is called a* ***core*** *of $G = (V, E)$, denoted $\mathsf{core}\,(G)$ if the following properties hold: for every $v \in \mathsf{core}\,(G)$:*

- *v has at least $np/5$ neighbors in $\mathsf{core}\,(G) \cap V_i$ for every $i \neq \varphi^*(v)$.*
- *v has at most $np/20$ neighbors from $V \setminus \mathsf{core}\,(G)$,*

Proposition 3.3 *Let G be a graph distributed according to $\mathcal{G}^{\text{plant}}_{n,p,3}$ with $np \geq C_0$, C_0 a sufficiently large constant. Then whp $|\mathsf{core}\,(G)\,| \geq (1 - e^{-\Theta(np)})n$.*

The main idea of the proof is to prove that the following procedure typically outputs a big core. *Set X to be the set of vertices having at least $np/4$ neighbors in G in each color class of φ^* other than its own. Then, repeatedly, delete from X any vertex that has less than $np/5$ neighbors in X in some color class other than its own or more than $np/20$ neighbors not in X.* This procedure clearly defines a core. To see why this core is typically large – observe that to begin with very few vertices are eliminated from the core (since all but $e^{-\Theta(np)}n$ vertices have degree, say, $0.99np/3$ in every color class other then their own). If too many variables were removed in the iterative step then a small but dense subgraph exists (as every removed vertex contributes at least $np/20$ edges to that subgraph). Proposition 3.1 bounds the probability of the latter occurring.

Next we characterize the structure of the graph induced by the non-core vertices (the non-core graph).

Proposition 3.4 *Let G be a graph distributed according to $\mathcal{G}^{\text{plant}}_{n,p,3}$ with $np \geq C_0$, C_0 a sufficiently large constant. Then whp every connected component in the non-core graph contains $O(\log n)$ vertices.*

Proposition 3.4, whose complete proof is given in [1], will not suffice to prove Theorem 1.1, and we need a further characterization of the non-core graph. Using similar techniques to those used in the proof of Proposition 3.4 we prove:

Proposition 3.5 *Let G be a graph distributed according to $\mathcal{G}^{\text{plant}}_{n,p,3}$ with $np \geq C_0$, C_0 a sufficiently large constant. Then with probability $1 - e^{-\Theta(np)}$ there exists no cycle in the non-core graph.*

4 Proof of Theorem 1.1

Using our analysis of the $\mathcal{G}^{\text{plant}}_{n,p,3}$ distribution we can proceed to the proof of Theorem **??**. We will split the proof into three parts covering the three items of Theorem **??**. First we show that the messages sent between variables and constraints spanned by $\mathsf{core}\,(G)$ must converge to the proper coloring when starting with a not "too noisy" coloring. Second we show that the messages sent in the non-core part of the graph, must also converge to the proper coloring or to the UNDECIDED message. Third and last, we show how in linear time one can recover a proper coloring of the graph which agrees with $\varphi*$ on all vertices which were colored properly by the message passing phase.

Proposition 4.1 *Let G be a graph sampled from $\mathcal{G}^{\text{plant}}_{n,p,3}$, and φ be a not-necessarily proper 3-coloring of G, that disagree with the planted coloring on the coloring of at most εn of the vertices for some $\varepsilon < \frac{1}{60}$. Then, whp all messages of the coloring version of $\mathsf{Gallager}(\varphi, 2)$, sent between variables and constraints spanned by $\mathsf{core}\,(G)$, converge after at most $O(\log n)$ iterations to the proper coloring.*

Roughly, we show that if a core variable, u, sends a wrong color (w.r.t to $\varphi*$) to some constraint during the i^{th} iteration, this must mean that in the previous iteration many of its core neighbors, w, sent to the constraint (edge) that connects them, $C_{u,w}$ a wrong color (i.e. not $\varphi^*(w)$). If $i = \Omega(\log n)$, tracing back iteratively to the first iteration, all the vertices which sent a wrong color produce a set which must be "too dense" with respect to Proposition **??** (as every vertex belonging to the core has many neighbors in every color class, so in order for it to take a wrong color, many of its neighbor should do the same), thus *whp* for $i = \omega(\log n)$, no such u can exist. The complete proof of Proposition 4.1 can be found in the appendix.

Proposition 4.2 *Let G be a graph sampled from $\mathcal{G}^{\text{plant}}_{n,p,3}$, and φ be a not-necessarily proper 3-coloring of G, that disagree with the planted coloring on the coloring of at most εn of the vertices for some $\varepsilon < \frac{1}{60}$. Then, with probability at least $1 - e^{\Theta(np)}$ all messages of $\mathsf{Gallager}(\varphi, 2)$ (in the coloring setting) converge after at most $O(\log n)$ iterations. Furthermore, all messages sent from a variable to a constraint that did not converge to* UNDECIDED*, converge to the planted coloring.*

By Proposition 4.1 we may assume that the messages spanned by $\mathsf{core}\,(G)$ converge after $O(\log n)$ iterations to the planted coloring. Therefore, after $O(\log n)$ iterations, $\mathsf{Gallager}$ is only running on the factor graph induced by the non-core vertices (with constant, correct, messages coming from the core). By Propositions **??** and 3.5, the non-core graph is *whp* a forest in which the largest tree contains at most a logarithmic number of vertices. Next, using induction, one proves that with each following iteration, the leaves in every tree in the remaining non-core factor graph must be colored correctly or colored by UNDECIDED, and this coloring remains constant for the rest of the execution (thus "chopping off" these leaves from the trees, and remaining with smaller ones). Therefore after additional $O(\log n)$ iterations, the algorithm converges and stops, and every vertex is either colored according to the planted coloring or is UNDECIDED. The complete proof of Proposition 4.2 can be found in the appendix.

In the third phase of Alon − Kahale it is proposed to extended the coloring obtained on core (G) to the non-core vertices colored by UNDECIDED using exhaustive search, which will run in polynomial time, but with a possible very large exponent. Having proved in Proposition C.3 that with probability at least $1 - e^{-\Theta(np)}$ the non-core induces a forest, one can perform a more efficient algorithm to color the non-core vertices. First, it is easy to recover in linear time all non-core vertices with neighbors in two different color classes. We are left with all UNDECIDED vertices which have neighbors in at most one color class. Define a list of possible colors for each of these vertices, each of length at least 2. Since the graph induced on these vertices is a forest, it is straightforward to find in linear time a proper coloring of these vertices from their respective lists.

5 Discussion

Message passing algorithms are a trendy and promising area of research in many interesting and fundamental optimization problems, attracting researchers from different disciplines, e.g. statistical physics. Some experimental results show the effectiveness of such algorithms for instances that seem "hard" for many other heuristics [**?**]. Alas, not many of the message passing algorithms were rigorously analyzed, due to the inherent complexity of this task. Nevertheless, in this work we give a rigorous analysis of a well-known message passing algorithm in the context of the colorability problem, pinpointing an interesting connection between this problem and decoding algorithms, and showing the possible value of message passing heuristics for colorability. Our results also extend naturally to the setting of 2-colorable 3-uniform bipartite hypergraphs [6]. One should also mention the work of Feige et al. [9] that show a similar result to Theorem **??**, for Warning Propagation (a message passing algorithm), in the context of planted 3SAT with a sufficiently large, yet constant, clause-variable ratio.

The coloring problem is a special case of the more general graph partition problems such as min-bisection, max-clique, the max independent set, to mention just a few, where spectral techniques have been shown to produce *whp* close to optimal solutions [2, 10, 4]. It would be interesting to understand whether one can define a proper setting for a message passing algorithm that can recover the optimal solution *whp*.

Acknowledgement

The authors would like to thank Michael Krivelevich and Simon Litsyn for their helpful comments.

References

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

[2] 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.

[3] B. Bollobás. The chromatic number of random graphs. *Combinatorica*, 8(1):49–55, 1988.

[4] Ravi B. Boppana. Eigenvalues and graph bisection: An average-case analysis (extended abstract). In *Proc. 28th IEEE Symp. on Found. of Comp. Science*, pages 280–285, 1987.

[5] A. Braunstein, M. Mezard, M. Weigt, and R. Zecchina. Constraint satisfaction by survey propagation. *Computational Complexity and Statistical Physics*, 2005.

[6] H. Chen and A. M. Frieze. Coloring bipartite hypergraphs. In W. H. Cunningham, T. S. McCormick, and M. Queyranne, editors, *Integer Programming and Combinatorial Optimization, 5th*

International IPCO Conference, Proceedings, volume 1084 of *Lecture Notes in Computer Science*, pages 345–358. Springer, 1996.

[7] U. Feige and J. Kilian. Zero knowledge and the chromatic number. *J. Comput. and Syst. Sci.*, 57(2):187–199, 1998. Complexity 96—The Eleventh Annual IEEE Conference on Computational Complexity (Philadelphia, PA).

[8] 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.

[9] U. Feige, E. Mossel, and D. Vilenchik. Complete convergence of message passing algorithms for some satisfiability problems. In *RANDOM*, 2006.

[10] U. Feige and E. Ofek. Finding a maximum independent set in a sparse random graph. 2006.

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

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

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

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

[15] L. Kučera. Expected behavior of graph coloring algorithms. In *Proc. Fundamentals of Computation Theory*, volume 56 of *Lecture Notes in Comput. Sci.*, pages 447–451. Springer, Berlin, 1977.

[16] M. Luby, M. Mitzenmacher, and M. A. Shokrollahi, Analysis of Random Processes via And-Or Trees. In *Proc. 9th Symp. on Discrete Algorithms*, 1998.

[17] 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.

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

[19] T. Łuczak. The chromatic number of random graphs. *Combinatorica*, 11(1):45–54, 1991.

[20] 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.

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

[22] M. Sipser, and D. A. Spielman. *Expander codes*. IEEE Trans. Inform. Theory, vol. 42, pp. 1710-1722, Nov. 1996.

A Proof of Proposition ??

Let $d = np$. Consider a fixed graph H on t vertices in which there are at least $\frac{edt}{2}$ edges. The probability of H being a subgraph of the sampled graph G is at most $\Pr[\text{the edges of } H \text{ were included}] = p^{edt/2}$. Let $\mathcal{A}$ be the bad property that a there exists such a graph H. Using the union bound one gets,

$$\begin{aligned}\Pr[\mathcal{A}] \leq & \sum_{t=1}^{n} \binom{n}{t} \binom{\binom{t}{2}}{\frac{edt}{2}} \cdot p^{edt/2} \leq \sum_{t=1}^{n} \left(\frac{en}{t}\right)^t \cdot \left(\frac{t}{d}\right)^{edt/2} \cdot p^{edt/2} \\ = & \sum_{t=1}^{n} \left(\frac{en}{t} \cdot \left(\frac{t}{n}\right)^{ed/2}\right)^t = \sum_{t=1}^{n} \left(\frac{e}{n} \cdot \left(\frac{t}{n}\right)^{ed/2-1}\right)^t \\ \leq & \sum_{t=1}^{\infty} \left(\frac{e}{n}\right)^t = o(1).\end{aligned}$$

Since n is large enough, and d is a sufficiently large constant, it is easy to see that $\Pr[\mathcal{A}]$ is bounded by a decreasing geometric series.

B Proof of Proposition 3.3

Let $\eta = e^{-\Theta(np)}$. Partition the variables not in $\mathsf{core}(G)$ into variables that were removed in the first step (variable that had an untypically large degree in some color class), call them A_0, and variables that were removed in the iterative step, A_i is the set of variables that were removed in the ith iteration, and let $\mathcal{A} = \bigcup_{i \geq 1} A_i$. Let C be the vertex set in the resulting $\mathsf{core}(G)$. If $C \leq (1-\eta)n$, then at least one of $\{A_0, \mathcal{A}\}$ has cardinality at least $\eta n/2$. Consequently,

$$\Pr[C \leq (1-\eta)n] \leq \underbrace{\Pr[|A_0| \geq \eta n/2]}_{(a)} + \underbrace{\Pr\left[|\mathcal{A}| \geq \eta n/2 \;\middle|\; |A_0| \leq \eta n/2\right]}_{(b)}.$$

Standard probabilistic calculations (e.g., Chernoff bound), can show that (a) occurs with small probability. To bound (b), observe that every variable that is removed in iteration i of the iterative step, has at least $(1-\delta)np/3 - (1-2\delta)np/3 = \delta np/3$ neighbors amongst $\{a_1, a_2, ..., a_{i-1}\} \cup A_0$, where a_i is the vertex that was removed in iteration i. Consider iteration $i = \eta n/2$. Assuming $|A_0| \leq \eta n/2$, by the end of this iteration there exists a set containing at most ηn vertices, inducing at least $\delta np/3 \cdot \eta n/2$ edges. By Proposition **??** this implies that (b) is bounded.

C Proof of Proposition 3.5

Let C be a fixed k-cycle, and let $V(C)$ be the variables in C. In order for C to belong to the non-core graph it must be that the edges of C were included in the input graph G and that $V(C) \cap \mathsf{core}(G) = \emptyset$. Let us start by bounding $\Pr[C \subseteq G \bigwedge V(C) \cap \mathsf{core}(G) = \emptyset]$ and then use the union bound over all possible cycles. 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 $\{C \subseteq G\}$ might have on the structure of $\mathsf{core}(G)$. To this end we introduce a set $\mathsf{core}^*(G)$, defined similarly to $\mathsf{core}(G)$ only "cushioned" in some sense to overcome the dependency issues. This is done using similar techniques to [1].

We start by defining the new set of core variables $\mathsf{core}^*(G)$ (again w.r.t. the planted coloring). First, set $\mathsf{core}(G)$ to be the set of vertices having at most $1.01np/3 - 2$ neighbors in G in each color class other than its own, and at least $0.99np/3$. Then, repeatedly, delete any vertex in $\mathsf{core}(G)$ having less than $0.98np/3$ neighbors in $\mathsf{core}^*(G)$ in some color class, other than its own.

Proposition C.1 *whp* $|\mathsf{core}^*(G)| \geq (1 - e^{-\Theta(d)})n$.

Observe that the only difference in the definition of $\mathsf{core}^*(G)$ is the addition of the -2 is the first step of the construction. Therefore, all the arguments in the proof of Proposition 3.3 carry through.

Lemma C.2 *For every cycle* C, $\mathsf{core}^*(G) \subseteq \mathsf{core}(G \cup C)$.

This lemma clarifies the motivation for defining $\mathsf{core}^*(G)$. It is not necessarily true that $\mathsf{core}(G) \subseteq \mathsf{core}(G \cup C)$. For example, a vertex which appears in $\mathsf{core}(G)$ could disappear from $\mathsf{core}(G \cup C)$ since the C makes its degree too high in some color class. Loosely speaking, the cushioning in $\mathsf{core}^*(G)$ defuses the effect of C on the core.

Proof.(Lemma C.2) The lemma is proved using induction on i (i being the iteration counter in the construction of the core). For the base case, if a vertex v survives the first step of constructing $\mathsf{core}^*(G)$, then v must have between $0.99np/3$ and $1.01np/3-2$ neighbors in every color class other than its own. When considering v's degree w.r.t. $G \cup C$, it has between $0.99np/3$ and $(1.01np/3-2)+2 = 1.01np/3$ neighbors in every color class other than its own (as the degree of v in C is at most 2), therefore v also survives the first step in the construction of $\mathsf{core}(G \cup C)$. As for the induction step, let $\mathsf{core}^*(G)^{(i)}$ be the core at the end of iteration i, and the same for $\mathsf{core}(G \cup C)$. If $v \in \mathsf{core}^*(G)^{(i)}$, this implies that v has at most $0.01np/3 - 2$ neighbors outside $\mathsf{core}^*(G)^{(i-1)}$. Since by the induction hypothesis, $\mathsf{core}^*(G)^{(i-1)} \subseteq \mathsf{core}(G \cup C)^{(i-1)}$, then v has at most $(0.01np/3 - 2) + 2 = 0.01np/3$ neighbors outside $\mathsf{core}(G \cup C)^{(i-1)}$, and therefore $v \in \mathsf{core}(G \cup C)^{(i)}$. ■

Proposition C.3

$$\Pr[C \text{ is a subgraph of } G \text{ and } V(C) \cap \mathsf{core}(G) = \emptyset] \leq \Pr[C \text{ is a subgraph of } G] \cdot \Pr[V(C) \cap \mathsf{core}^*(G) = \emptyset].$$

Proof. It suffices to show that

$$\Pr[V(C) \cap \mathsf{core}(G) \,|\, C \text{ is a subgraph of } G] \leq \Pr[V(C) \cap \mathsf{core}^*(G) = \emptyset].$$

But, by Lemma C.2,

$$\begin{aligned}
&\Pr[V(C) \cap \mathsf{core}^*(G) = \emptyset] = \\
&\sum_{F: V(C) \cap \mathsf{core}^*(F) = \emptyset} \Pr[E(G) = F] \geq \\
&\sum_{F: V(C) \cap \mathsf{core}(F \cup C) = \emptyset} \Pr[E(G) = F] = \\
&\sum_{F': F' \cap C = \emptyset, V(C) \cap \mathsf{core}(F' \cup C) = \emptyset} \Pr[E(G) \setminus C = F'] = \\
&\sum_{F': F' \cap C = \emptyset, V(C) \cap \mathsf{core}(F' \cup C) = \emptyset} \Pr[E(G) \setminus C = F' \,|\, C \text{ is a subgraph of } G] = \\
&\Pr[V(C) \cap \mathsf{core}(G) \,|\, C \text{ is a subgraph of } G].
\end{aligned}$$

where F ranges over the sets of edges with endpoints in different color classes, and F' ranges over those sets that do not intersect C. The third equation follows by regrouping the edge-sets F according to $F' = F \setminus C$ and noting (the obvious fact) that, for a given set F' that does not intersect C, the probability that $E(G) = F$ for some F such that $F \setminus C = F'$ is equal to $\Pr[E(G) \setminus C = F']$. The fourth equation follows from the independence of the events $E(G) \setminus C = F'$ and C is a subgraph of G. ■

Corollary C.4 *Let λ be s.t. the number of vertices in $\mathsf{core}(G)$ is $(1-\lambda)n$, then*

$$\Pr[C \text{ is a subgraph of } G \text{ and } V(C) \cap \mathsf{core}(G) = \emptyset] \leq p^k \cdot \lambda^k.$$

Proof. As the edges are chosen independently,

$$\Pr[C \text{ is a subgraph of } G] = p^k.$$

Since C is a fixed cycle and the choice of $\mathsf{core}^*(G)$ is independent of $V(C)$, it follows that

$$\Pr[V(C) \cap \mathsf{core}(G) = \emptyset] \leq \frac{\binom{\lambda n}{k}}{\binom{n}{k}} \leq \lambda^k.$$

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

Corollary C.5 *Let G be a graph sampled according to $\mathcal{G}^{\text{plant}}_{n,p,k}$, np a sufficiently large constant, then the probability of a cycle in the graph induced by the vertices not in $\mathsf{core}(G)$ is at most $e^{-\Theta(np)}$.*

Proof. Let λ be s.t. the number of vertices in $\mathsf{core}(G)$ is $(1-\lambda)n$. Let us bound the number of cycles of length k. There are $\binom{n}{k}$ ways to choose the vertices inducing the cycles, and $k!/2$ ways to order them on the cycle. Once the order of the vertices is fixed, the edges are uniquely determined. Using the union bound one obtains that the probability of a cycle in the non-core graph is at most

$$\sum_{k=1}^{\lambda n} \binom{n}{k} \cdot k! \cdot p^k \cdot \lambda^k \leq \sum_{k=1}^{\lambda n} \left(\frac{en}{k}\right)^k \cdot k^k \cdot p^k \cdot \lambda^k = \sum_{k=1}^{\lambda n} (e \cdot np \cdot \lambda)^k$$

Assuming that Proposition C.1 holds (which is the case *whp*), then $\lambda = e^{-\Theta(np)}$ and $e \cdot np \cdot e^{-\Theta(np)}$ 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 $2e \cdot np \cdot e^{-\Theta(np)} = e^{-\Theta(np)}$. ■

Corollary C.5 completes the proof of Proposition 3.5.

D Proof of Proposition 4.1

Let $u \in \mathsf{core}(G)$, and assume that $x_u \xrightarrow{i} C = r \neq \varphi^*(u)$ for some constraint C and some iteration i (where $\xrightarrow{i}$ denote messages sent during the ith iteration). For every vertex $w \in N_G(u)$, let $\varphi^0(w) = \varphi(w)$ be the not-necessarily proper coloring of the vertices of G. In every iteration of the message passing algorithm we think of the messages sent from w to the constraint that represents the edge connecting it to u, $C_{(w,u)}$ as a (not-necessarily proper) coloring of $N_G(u)$, setting $\varphi^i(w) = x_w \xrightarrow{i} C_{(w,u)}$. For $j \in \{1,2,3\}$, denote by $W^j = \{w \in N_G(u) \ : \ \varphi^{i-1}(w) = j\}$, the set of u's neighbors which are colored by j at the end of iteration $i-1$, by $W^* = \{w \in N_G(u) \cap \mathsf{core}(G) \ : \ \varphi^{i-i}(w) \neq \varphi^*(w)\}$, the set of u's wrongly colored $\mathsf{core}(G)$ neighbors at the end of iteration $i-1$, and by $R = \{w \in N_G(u) \cap \mathsf{core}(G) \ : \ \varphi^*(w) = r\}$, the set of all of u's $\mathsf{core}(G)$ neighbors, whose planted coloring is r. Set $d = np/3$. Note that if $r \in \{1,2,3\}$ then $|R| \geq (1-2\delta)d$, since u has at least $(1-2\delta)d$ core-vertex neighbors in each color class. Moreover, u has at most $2((1+\delta)d - (1-2\delta)d) = 6\delta d$ neighbors outside of $\mathsf{core}(G)$.

First, assume $r \in \{1,2,3\}$. Any $\mathsf{core}(G)$ neighbor of u that was properly colored at the end of iteration $i-1$ and whose planted coloring is r is in $R \setminus W^*$, thus $|R| - |W^*| \leq |W^r|$. Since r was the least popular color in the ith iteration, we have that $|W^r| \leq |W^{\varphi^*(u)}|$. And finally, any neighbor of u

colored by $\varphi^*(u)$ at the end of iteration $i-1$ is obviously wrongly colored, thus $|W^{\varphi^*(u)}| \leq |W^*| + 6\delta d$. Putting it all together yields that $|W^*| \geq (\frac{1}{2} - 4\delta)d$. Second, assume $r = \text{UNDECIDED}$, then by the end of iteration $i-1$, u must have had at least $2(1-2\delta)d - 6\delta d - 4 \geq (\frac{1}{2} - 4\delta)d\,\mathsf{core}\,(G)$ neighbors colored by UNDECIDED, since in order to be colored by UNDECIDED u must have received at most 2 messages from at least two colors.

Let U_0 be the set of $\mathsf{core}\,(G)$ vertices which are wrongly colored by φ, and set for every positive integer k, U_k to be the $\mathsf{core}\,(G)$ vertices that by the end of iteration k send to some constraint a wrong color or UNDECIDED. We will prove inductively that $|U_k| \leq \frac{2}{3}|U_{k-1}|$. We have seen that every vertex in U_1 has at least $(\frac{1}{2} - 4\delta)d$ neighbors in U_0. On the other hand, every vertex in U_0 has at most $2(1+\delta)d$ neighbors. Set $U = U_0 \cup U_1$, and $U' = U_0 \cap U_1$. We have that

$$\frac{1}{2}(1-2\delta)d|U_1| - \frac{1}{4}(1-8\delta)d|U'| \leq e(U_0, U_1) \leq 2(1+\delta)d|U_0| - (1+\delta)d|U'|,$$

where $e(U_0, U_1)$ denotes the number of edges that have one endpoint in U_0 and the other in U_1. Thus, $|U_1| \leq \frac{4(1+\delta)}{1-8\delta}|U_0| = \delta'|U_0|$, and recalling that $|U_0| \leq \varepsilon n$ implies $|U| \leq |U_0| + |U_1| \leq \varepsilon(1+\delta')n$. By Proposition **??**, we have that $e(U) \leq \varepsilon(1+\delta')d|U|$. On the other hand,

$$\frac{1}{4}(1-8\delta)d|U_1| \leq e(U_0, U_1) \leq e(U).$$

It follows that

$$|U_0| + |U_1| \geq |U| \geq \frac{1-8\delta}{4\varepsilon(1+\delta')}|U_1| = \frac{(1-8\delta)^2}{\varepsilon(20-16\delta)}|U_1| > \frac{5}{2}|U_1|,$$

completing the base of the induction. By the induction hypothesis, we have that $|U_{k-1}| \leq \varepsilon n$, and the proof of the general case follows exactly the steps of the base case.

The above obviously implies that all messages from $\mathsf{core}\,(G)$ variables to the constraints converge after $O(\log n)$ iterations, but it also implies that the messages from the constraints (edges) spanned by the $\mathsf{core}\,(G)$ variables also converge, since every message sent from a constraint to a variable, is a repeated message received at this constraint in the previous iteration, completing the proof.

E Proof of Proposition 4.2

By Propositions 3.5 and 4.1, we may assume that the graph $G_0 = G[V \setminus \mathsf{core}\,(G)]$ is a forest, and that after $O(\log n)$ iterations all messages spanned by $\mathsf{core}\,(G)$ converge to the proper coloring. We note that it suffices to show that all messages leaving variables will converge as stated, since all messages leaving the constraints are just repeaters of messages received at the constraint in the previous iteration, thus must converge properly as well. Assume that the algorithm has been run for $j = \Theta(\log n)$ iteration, long enough for it to converge on $\mathsf{core}\,(G)$. For all iterations $k > j$, all messages leaving $\mathsf{core}\,(G)$ variables carry the proper color. Since G_0 is a forest, we proceed to prove inductively that with every iteration the message passing algorithm will converge on the leaves of the remainder graph. For the base case, let K_0 be the leaves of G_0, and let v be such a leaf. All neighbors of v but at most one are in $\mathsf{core}\,(G)$, thus all $C \xrightarrow{k} x_v = \varphi^*(u)$ for all constraints C but at most one, where C is the constraint corresponding to the edge (x_u, x_v). It follows that $x_v \xrightarrow{k} C \in \{\varphi^*(v), \text{UNDECIDED}\}$ for all $k \geq j+1$. Now, for the general case, define iteratively for every positive integer r, G_r to be the graph $G[V(G_{r-1}) \setminus K_{r-1}]$, and K_r its leaves. Let v be some vertex in K_r. Following our induction hypothesis, by the end of iteration $j+r$ all of the neighbors but at most one of v will either be properly colored or colored by UNDECIDED. Therefore for all $k \geq j+r$, $C \xrightarrow{k} x_v \in \{\varphi^*(u), \text{UNDECIDED}\}$ for all constraints C but at most one, where C is constraint corresponding to the edge (x_u, x_v). Thus, $x_v \xrightarrow{k} C \in \{\varphi^*(v), \text{UNDECIDED}\}$ as in the base case, since x_v can get at most one "wrong" message in iteration $j+r$ (UNDECIDED is not considered as a "wrong" message). Finally, since by Proposition

?? the depth of the forest G_0 is at most $O(\log n)$, we have that after $O(\log n)$ iterations the message passing algorithm converges to either the proper coloring or to UNDECIDED.