2007 Conference on Analysis of Algorithms, AofA 07 DMTCS proc. **AH**, 2007, 235–246

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

Sonny Ben-Shimon[1†] and Dan Vilenchik[1 ‡]

[1] *School of Computer Science,*
Raymond and Beverly Sackler Faculty of Exact Sciences,
Tel Aviv University, Tel Aviv, Israel.

received 23rd Feb 2007, revised 17th May 2007, accepted 19th January 2008.

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. In 1962 Gallager 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[(i)] 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 between the two problems – coloring and decoding. Theorem 1.1 makes this notion formal. Hopefully, following this insight, other

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

(i) By writing whp we mean with high probability, i.e., with probability tending to 1 as n goes to infinity.

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 $n/120$ 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). From here on we will refer to Gallager's decoding algorithm by Gallager and to Alon and Kahale's coloring algorithm by Alon − Kahale.

Theorem 1.1 *Fix $\varepsilon \leq \frac{1}{120}$ 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 constant. 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 − Kahale is also described in Section 2.4 for the sake of completeness. For simplicity, we identify by 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, or more precisely, at least $(1-e^{-\Theta(np)})n$ vertices are properly colored. This is optimal in some sense as whp there will be $e^{-\Theta(np)}n$ vertices in G with degree at most $k-2$, which can actually take more than one color without violating the k-coloring constraints. Nevertheless, in both the decoding setting and the coloring 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 Structure of Paper

The remaining of the paper is structured as follows. In Section 2 we present Gallager's decoding algorithm – Gallager – and it's adaptation to the coloring 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 become 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 $b \in \{0,1\}$.

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

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

Fig. 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.

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 $C_j \in E(G)$

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, $x_i = B_i$.

Fig. 2: Gallager's algorithm

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.

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

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

Fig. 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 $\mathcal{G}_{n,p,3}^{\text{plant}}$, 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 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

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 $G[U]$.

Fig. 4: Alon and Kahale's coloring algorithm

vertices. This coloring typically colors all but $e^{-\Theta(np)}n$ vertices. Finally, in step 4, the small graph induced by the uncolored vertices is exhaustively searched for a k-coloring that extends the k-coloring of the rest of the graph.

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. It is then left simply to observe that the unified step of the new Alon $-$ Kahale is exactly 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 in $\mathcal{G}^{\text{plant}}_{n,p,3}$. These properties will come in handy when proving Theorem 1.1 in the next section. Properties of similar flavor can be found in (1) along with complete proofs which are omitted here due to space considerations. Given a subset U of vertices, $e(U)$ denotes the number of edges spanned by the vertices of U.

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 $np/15$.*

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 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 vertices 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

A graph G is said to be *typical* in $\mathcal{G}^{\text{plant}}_{n,p,3}$ if Propositions 3.1, 3.3, 3.4 and 3.5 hold for it. The discussion in Section 3 guarantees that whp G is typical. Therefore, to prove Theorem 1.1 it suffices to consider a typical G. In the propositions below we assume that G is such and do not state anew each time.

We will split the proof into two parts corresponding to the three items of Theorem 1.1. First we show that the messages sent between variables and constraints spanned by $\mathsf{core}\,(G)$ converge to φ^* when starting with a not "too noisy" coloring. Second we show that the messages sent in the non-core part of the graph converge to φ^* or to the UNDECIDED message. This combined with Proposition 3.5 implies that the coloring of the UNDECIDED vertices can be completed in linear time (list coloring of a tree).

Proposition 4.1 *Let φ be a 3-coloring of G that disagree with φ^* on at most $n/120$ vertices. Then whp all messages of* $\mathsf{Gallager}(\varphi,2)$ *spanned by* $\mathsf{core}\,(G)$ *converge after at most $O(\log n)$ iterations. Furthermore, the message B_i of $v_i \in \mathsf{core}\,(G)$ equals $\varphi^*(v_i)$.*

Proof. For every vertex u we define $\varphi^{(i)}_{u\to e}$ to be the value of the message $u \to e$ that the variable (vertex) u sends the constraint (edge) $e=(u,v)$ in iteration i of Gallager (according to the message defined in Figure 3). Let U_i be the set of core vertices for which $\varphi^{(i)}_{(u,e)} \neq \varphi^*(u)$. It suffices to prove that $|U_i| \leq |U_{i-1}|/2$ (if this is true, then after $\log n$ iterations $U_{\log n} = \emptyset$). Observe that by our assumption on φ – $|U_0| \leq n/60$. By contradiction, assume that not in very iteration $|U_i| \leq |U_{i-1}|/2$, and let j be the first iteration violating

the inequality – $|U_j| \geq |U_{j-1}|/2$. Consider a vertex $u \in U_j$. If $\varphi^{(i)}_{u\to e} \neq \varphi^*(u)$, then there must be at least $np/20$ vertices w amongst u's neighbors s.t. $\varphi^{(i-1)}_{w\to e} \neq \varphi^*(w)$. To see this, observe that any vertex in $\mathsf{core}\,(G)$ has at most $np/20$ neighbors outside $\mathsf{core}\,(G)$ (by definition of core), and hence if it has at most $np/20 + np/20 = np/10$ wrongly colored neighbors, then it has at least $np/4 - np/10 > np/10$ vertices from the each of the correct colors, and therefore $\varphi^{(i)}_{u\to e}$ agrees with $\varphi^*(u)$ (contradicting our assumption). For conclusion, let $U = U_j \cup U_{j-1}$. Then every $u \in U_j$ has at least $np/10$ neighbors in U_{j-1}. Assuming $|U_j| \geq |U_{j-1}|/2$, the average degree in $G[U]$ is at least $\frac{np/10\cdot|U_j|}{|U|} \geq \frac{np/10\cdot|U_j|}{1.5|U_j|} \geq np/15$. Recalling that $|U| \leq 2 \cdot n/120 \leq n/60$ contradicts Proposition 3.1. ■

Proposition 4.2 *Let i_0 be the iteration in which $\varphi^{(i_0)}_{u\to e} = \varphi^*(u)$ for every $u \in \mathsf{core}\,(G)$ ($\varphi^{(i)}_{u\to e}$ is the message $u \to e$ in the i^{th} iteration). Then after at most $t = i_0 + O(\log n)$ iterations, all messages of* $\mathsf{Gallager}(\varphi, 2)$ *converge. Furthermore, for every $u \in V(G)$, $\varphi^{(t)}_{u\to e}$ is either $\varphi^*(u)$ or* UNDECIDED.

Proof. We first 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. Let $G_0 = G[V \setminus \mathsf{core}\,(G)]$ and consider the factor graph that G_0 induces (which is also a tree); for a tree edge (u, e) in the factor graph we define $\texttt{level}(u, e)$ to be r if r is the maximal length of a path between u and a leaf in the factor graph from which the edge (u, e) is removed. By induction on the level r we prove that in iteration $r + i_0$ all vertices at level r transmit either UNDECIDED or their planted coloring. Observe that by the assumption in the proposition, at all iterations $i > i_0$, all messages leaving $\mathsf{core}\,(G)$ into G_0 state the correct color.

The base case is an edge (u, e) at level 0. If $\texttt{level}(u, e) = 0$ then Then $\varphi^{(i_0)}_{v\to e}$ is either $\varphi^*(v)$ or UNDECIDED since the only messages that take part in the calculation are the ones coming from $\mathsf{core}\,(G)$ (if any), and they agree with $\varphi^*(u)$. Now consider an edge (u, e) at level r, and consider iteration $i_0 + r$. The key observation is that the level of all edges (v, e'), $e' = (u, v)$ is strictly smaller than that of (u, e) (since when removing (u, e), the way from u to all the leaves passes through every (v, e')). Now apply the induction hypothesis and repeat the same argument of the induction step. Finally, since by Proposition 3.4 the level of an edge is at most $O(\log n)$, we have that after $O(\log n)$ iterations (from i_0) Gallager converges to either the proper coloring or to UNDECIDED. ■

This completes the proof of Theorem 1.1.

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, coding theory and more. Some experimental results show the effectiveness of such algorithms for instances that seem "hard" for many other heuristics (5). 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 result also works in the other direction, that is by introducing new analytical tools to analyze Gallager in the colorability setting we suggest the same tools for establishing rigorous results in the ever developing study of LDPC codes.

Our results also extend naturally to other partition problems where spectral techniques provide a first approximation and then an iterative algorithm is used to find a proper solution (2; 10; 4). It would be interesting to see whether one can adjust Gallager to these settings and prove similar results to ours.

Another interesting question is to study the counterpart of code-rate in the colorability problem. Specifically, what is the "farthest" coloring from the planted one, from which one can prove convergence of Gallager. A

random 3-coloring will be at distance roughly $2n/3$ from the planted one. Now if every vertex with d neighbors has exactly $d/3$ neighbors in every color class (which is what happens in expectation when starting from a random coloring), then this is a fixed point of the system and Gallager stays "stuck". Therefore it is safe to guess $2n/3$ as an upper bound for that distance. In this work our techniques enable us to prove $n/120$ as a lower bound, and although no effort was made to optimize the constants, we do not believe to be able to reach the aforementioned upper bound. It will be interesting to close this gap.

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.