Near-Optimal Distributed Edge Coloring *

Devdatt Dubhashi[1]** and Alessandro Panconesi[2]***

[1] **BRICS***, Department of Computer Science, University of Aarhus, Ny Munkegade, DK-8000 Aarhus C, Denmark
[2] Centrum voor Wiskunde en Informatica 413 Kruislaan, 1098 SJ, Amsterdam, Holland

Abstract. We give a distributed randomized algorithm to edge color a network. Given a graph G with n nodes and maximum degree Δ, the algorithm,

- For any fixed $\lambda > 0$, colours G with $(1+\lambda)\Delta$ colours in time $O(\log n)$.
- For any fixed positive integer s, colours G with $\Delta + \frac{\Delta}{(\log \Delta)^s} = (1 + o(1))\Delta$ colours in time $O(\log n + \log^s \Delta \log\log \Delta)$.

Both results hold with probability arbitrarily close to 1 as long as $\Delta(G) = \Omega(\log^{1+d} n)$, for some $d > 0$. The algorithm is based on the Rödl Nibble, a probabilistic strategy introduced by Vojtech Rödl. The analysis involves a certain quasi–random phenomenon involving sets at the vertices of the graph.

1 Introduction

The edge coloring problem is a basic problem in graph theory and combinatorial optimization. Its importance in distributed computing, and computer science generally, stems from the fact that several scheduling and resource allocation problems can be modeled as edge coloring problems [9, 11, 14, 17]. In this paper, we give a distributed randomized algorithm that computes a near-optimal edge coloring in time $O(\log n)$. By "near-optimal" we mean that the number of colors used is $(1 + o(1))\Delta$ where Δ denotes the maximum degree of the network and the $o(1)$ term can be as small as $1/\log^s \Delta$, for any $s > 0$. Both performance guarentees – the running time and the number of colours used – hold with high probability as long as the maximum degree grows at least logarithmically with

* A preliminary version of this work was presented at the 15th International Symposium on Mathematical Programming, August 1994, Ann Arbour, Michigan, USA.

** dubhashi@daimi.aau.dk. Work done partly while at the Max–Planck–Institute für Informatik supported by the ESPRIT Basic Research Actions Program of the EC under contract No. 7141 (project ALCOM II).

*** puck@math.tu-berlin.de. Supported by an Ercim postdoctoral fellowship. Current address: Fachbereich Mathematik, MA 6-1, TU Berlin, Str. des 17 Juni 136, 10623 Berlin.

*** Basic Research in Computer Science,
Centre of the Danish National Research Foundation.

n. Our algorithm can be implemented directly in the PRAM model of computation.

Motivation and Related Work. The edge coloring problem can be used to model certain types of jobshop scheduling, packet routing, and resource allocation problems in a distributed setting. For example, the problem of scheduling I/O operations in some parallel architectures can be modeled as follows [9, 6]. We are given a bipartite graph $G = (\mathcal{P}, \mathcal{R}, E)$ where, intuitively, $\mathcal{P}$ is a set of processes and $\mathcal{R}$ is a set of resources (say, disks). Each processor needs data from a subset of resources $R(p) \subseteq \mathcal{R}$. The edge set is defined to be $E = \{(p, r) : r \in R(p), p \in \mathcal{P}\}$. Due to hardware limitations only one edge at the time can be serviced. Under this constraints it is not hard to see that optimal edge colorings of the bipartite graph correspond to optimal schedules that is, schedules minimizing the overall completion time. Clearly, if a graph G has maximum degree Δ then at least Δ colors are needed to edge color the graph. A classical theorem of Vizing shows that $\Delta+1$ colors are always sufficient, and the proof is actually a polynomial time algorithm to compute such a coloring (see for example [4]). Interestingly, given a graph G, it is NP-complete to decide whether it is Δ or $\Delta+1$ edge colorable [8], even for regular graphs [7]. Efforts at parallelizing Vizing's theorem have failed; the best PRAM algorithm known is a randomized algorithm by Karloff & Shmoys that computes an edge coloring using very nearly $\Delta + \sqrt{\Delta} = (1 + o(1))\Delta$ colors. The Karloff & Shmoys algorithm can be derandomized by using standard derandomization techniques [3, 16]. In the distributed setting the previously best known result was a randomized algorithm by Panconesi & Srinivasan that uses roughly $1.58\Delta + \log n$ colors with high probability and runs in $O(\log n)$ time with high probability. For the interesting special case of bipartite graphs Lev, Pippinger & Valiant show that Δ-colorings can be computed in NC, whereas this is provably impossible in the distributed model of computation even if randomness is allowed (see [18]).

Our solution. To state our results precisely, we reproduce below our main theorem:

Theorem 1. *For any fixed $\lambda > 0$, given a graph with n vertices and maximum degee Δ, we can edge colour the graph with $(1 + \lambda)\Delta$ colours in time $O(\log n)$ where n is the number of vertices in the graph. For any fixed positive integer s, we can edge colour it with $\Delta + \Delta/\log^s \Delta = (1 + o(1))\Delta$ colours in time $O((\log \Delta)^s \log\log \Delta + \log n)$. The results hold with failure probability decreasing to 0 faster than any polynomial (in n) provided that $\Delta = \Omega(\log^{1+d} n)$ for some $d > 0$.*

Our algorithm is based on the *Rödl Nibble*, a beautiful probabilistic strategy introduced by Vojtech Rödl to solve a certain covering problem in hypergraphs [2, 20]. The method has subsequently been used very successfully to solve other combinatorial problems such as asymptotically optimal coverings and colorings for hypergraphs [2, 10, 19, 21]. In this paper, we introduce it as a tool for the

design and analysis of randomized algorithms.[4] Although the main component of our algorithm is the Rödl nibble and the intuition behind it rather compelling, the algorithm requires a non-trivial probabilistic analysis of a so called quasi-random process. To explain what this is, it is perhaps best to give a brief outline of our algorithm. Starting with the input graph G_0 the algorithm generates a sequence $G_0, G_1, \ldots, G_t$ of graphs. One can view each edge e as possessing a palette of available colors, starting with the whole set of $[\Delta]$ colours initially. At an arbitrary stage, a small ϵ fraction of uncolored edges is selected, and each selected edge chooses a tentative color at random from its current palette. If the tentative color is not chosen by any neighboring edge it becomes final. Palettes of the remaining uncolored edges are updated in the obvious fashion– by deleting colors used by neighboring edges. The process is then repeated. Like other proofs based on the same method our proof hinges on two key features of the Rödl nibble. The first key idea of the method is that if colors are chosen independently, the probability of color conflict is roughly ϵ^2, a negligible fraction of all edges attempting coloring at this stage. If the same "efficiency" is maintained throughout, the overall "wastage" will be very small. The second aspect of the Rödl nibble is a deeper mathematical phenomenon called *quasi-randomness* (see [2]). In our context, quasi-randomness means that the palettes of available colors at the edges at any stage are "essentially" truly independent random subsets of the original full palette. The crux of the analysis is to show that despite the potential of a complicated interaction regulated by the topology of the underlying graph, the "nibbling" feature of the coloring process ensures that the palettes are evolving almost independently of each other. In all applications of the nibble method, it is the quasi-random aspect which is mathematically challenging and which usually requires a quite laborious probabilistic analysis.

2 Preliminaries

A *message–passing distributed network* is an undirected graph $G = (V, E)$ where vertices (or nodes) correspond to processors and edges to bi–directional communication links. Each processor has its unique *id.* The network is *synchronous*, , computation takes place in a sequence of *rounds*; in each round, each processor reads messages sent to it by its neighbors in the graph, does any amount of local computation, and sends messages back to all of its neighbors. The time complexity of a distributed algorithm, or *protocol*, is given by the number of rounds needed to compute a given function. If one wants to translate an algorithm for this model into one for the PRAM then computation locally done by each processor must be charged for. An *edge coloring* of a graph G is an assignment of colors to edges such that incident edges always have different colors. The edge

[4] This research was originally prompted by a conversation that the second author had with Noga Alon and Joel Spencer, in which they suggested that the nibble approach should work. Noga Alon has recently informed us that he is already in possession of a solution with similar performance [1]. However, at the time of writing, a written manuscript was not available for comparison.

coloring problem is to find an edge coloring with the aim of minimizing the number of colors used. Given that determining an optimal (minimal) coloring is an NP-hard problem this requirement is usually relaxed to consider approximate, hopefully even near-optimal, colorings. The edge coloring problem in a distributed setting is formulated as follows: a distributed network G wants to compute an edge coloring of its own topology. As remarked in the introduction such a coloring might be useful in the context of scheduling and resource allocation. The set $\{1, 2, \ldots, n\}$ will be denoted by $[n]$. Given a graph G and a set of edges F, $G[F]$ denotes the subgraph of G whose edge set is F. In the paper we will use the following approximations repeatedly: $(1 - 1/n)^n \approx e^{-1}$, and $e^\epsilon \approx 1 + \epsilon$ or $e^\epsilon \approx 1 + \epsilon + \epsilon^2/2$, for small values of ϵ. Whenever such an approximation is in effect, we will use the sign $\approx$ in place of the equality sign. We will make use of a slight modification of a well-known vertex coloring algorithm by Luby [13]. Luby's algorithm computes a $(\Delta + 1)$–vertex coloring of a graph in expected time $O(\log n)$, where n is the number of vertices of a graph of maximum degree Δ. The running time of the algorithm is $O(\log n)$ with high probability [12, 13]. When applied to the line graph of G the algorithm computes a $(2\Delta - 1)$–edge coloring. In the original algorithm each vertex is initially given a palette of $\Delta + 1$ colors; it can be easily verified that the algorithm still works in the same fashion if each vertex u is given a palette of $\deg(u) + 1$ colors instead, where $\deg(u)$ is the degree of u. This modification is introduced for explanatory purposes.

3 The Algorithm

The algorithm is in two phases. The first phase is an application of the Rödl nibble and has the goal of coloring most of the edges using a palette of Δ colors. By the end of this phase we will be left with a graph whose maximum degree is at most $\kappa\Delta$ with high probability. In the second phase the modified Luby's algorithm is used to color the remaining graph with at most $2\kappa\Delta$ fresh colors. As we shall see in section 4.1, the number of iterations needed to bring the degree down from Δ to $\kappa\Delta$ is $O(\log(1/\kappa)/\alpha\kappa^2)$, where $\alpha = \epsilon(1 - \epsilon)e^{-4\epsilon}$. Hence, in order to get a $(1 + \lambda)\Delta$, where $\lambda > 0$ is *any* fixed constant, the first phase takes constant time. To get a $(1 + o(1))\Delta$ coloring takes $O((\log \Delta)^{2s} \log\log \Delta)$ time, where the $o(1)$ term is $1/(\log \Delta)^s$, for any $s > 0$. This holds with high probability. The exact probability of success will be determined in the section devoted to the analysis. We note here that an assumption on the maximum degree of the graph is needed, namely $\Delta(G) = \Omega(\log^{1+d} n)$, for some $d > 0$ (n denotes the number of vertices of G). Phase 2 takes $O(\log n)$ time, with high probability. The basic idea underlying the first phase of the algorithm is for each vertex to select a small "nibble" of edges incident upon it and assign tentative colors to them independently at random. Most of these edges are expected to avoid conflicts with other edges vying for coloring, and get successfully colored at this stage. This is because the nibble keeps the "efficiency" of the coloring close to 1 at each stage. To describe the algorithm more precisely, we introduce some definitions that will also be used later in the analysis. At any stage $k \geq 1$, we have a graph

$G_k(V, E_k)$. Initially, $G_0(V, E_0) := G(V, E)$, the input graph. By Δ_k we denote the maximum degree of the graph G_k (note $\Delta_0 = \Delta(G)$ initially). Each vertex has a palette of available colors, A_k^u with $A_0^u = [\max_{w \in \delta(u \cup \{u\}} \deg(w)]$. (This can be arranged in one round with each vertex communicating its own degree to each of its neighbours.) The set of edges successfully colored at stage k is denoted by C_k. Then, $G_{k+1} := G_k[E - C_k]$ is the graph passed on to the next stage. In the algorithm, $t(\epsilon, \kappa)$ denotes the number of stages needed to bring the maximum degree of the graph from Δ to $\kappa\Delta$ with high probability, and has value

$$t(\epsilon, \kappa) = \left\lceil \frac{\ln(1/\kappa)}{\epsilon(1-\epsilon)e^{-4\epsilon}\kappa} \right\rceil .$$

The algorithm is more precisely described as follows

Phase 1. RODL NIBBLE

For $k = 1, 2, \ldots, t(\epsilon, \kappa)$ stages repeat the following:

- Each vertex u randomly selects an ϵ fraction of the edges incident on itself, and independently at random assigns them a tentative color from its palette A_k^u of currently available colors. If an edge $e = \{u, v\}$ is selected by both its endpoints, it is simply dropped and not considered for coloring at this stage.
- Let $e = \{u, v\}$ be a selected edge, and $c(e)$ its tentative color. Color $c(e)$ becomes the final color of e unless one of the following two conflict types arises: *i)* some edge incident on e is given the same tentative color, or *ii)* $c(e) \notin A_k^u \cap A_k^v$, the tentative color given to e is not available at the other endpoint of e.
- The graph is updated by setting

$$A_{k+1}^u = A_k^u - \{c : \ e \text{ incident on } u,\ c(e) = c \text{ is the final color of } e\}$$

and $G_{k+1} = G_k[E_k - C_k]$, where C_k is the set of edges which got a final color at stage k.

Phase 2.

Color $G_{t(\epsilon,\kappa)}$ with fresh new colors by using the modified Luby's algorithm.

4 Analysis

4.1 Intuitive Outline

Suppose for a start that the graph is Δ–regular. Intuitively, the palettes A_k^u are more–or–less random subsets of the base set $[\Delta]$. Let us assume they are indeed truly random subsets of $[\Delta]$, so precisely, let us assume that the palette of each vertex at stage $k \geq 0$, is a uniformly and independently chosen random subset of $[\Delta]$ of the same size Δ_k. Then, at stage k (with high probability), the size of the common palette between any two vertices is Δ_k^2/Δ. So the probability that a colour chosen by a vertex as a tentative colour for an incident edge is also valid at the other end–point is Δ_k/Δ. Hence, the probability that an edge

is successfully coloured at stage k is roughly, $\epsilon\frac{\Delta_k}{\Delta}$ and we have the following recurrence for the vertex degree,

$$\Delta_{k+1} \leq (1-\epsilon\frac{\Delta_k}{\Delta})\Delta_k \leq \exp(-\epsilon\frac{\Delta_k}{\Delta})\Delta_k$$

This recurrence implies that given a fixed $0 < \lambda < 1$, the vertex degree drops to $\lambda\Delta$ within a constant number of stages, or that for any positive integer $s > 0$, the degree drops to $\Delta/(\log\Delta)^s$ in a poly–logarithmic (in Δ) number of stages. This yields the required time complexity analysis for the algorithm. Unfortunately, neither of the two assumptions above are in fact valid. First, because the graph G can have a very complex, irregular topology, it is not true that vertex degrees and palettes are uniform, at the outset, and they are even less likely to remain so at subsequent stages. In addition, the palettes are not truly independent random subsets either, as they can interact over the stages in a potentially complicated fashion governed by the topology of the graph. However, we show in § 4.2 below, that despite the possibility of a complex interaction in the graph, the "nibbling" feature of the colouring process leads to an essentially local interaction of the palettes. So, while the palettes are not truly random subsets, they behave essentially as such, specifically, with regard to the relative size and composition of the common palettes and the palettes themselves. Given this one simple, but crucial feature of the interaction of the palettes,it follows that the decay law is essentially as given above. To highlight the essential ideas, we start with some simplifying assumptions and progressively, we remove the assumptions and refine the argument. First we give an analysis under the assumption that the initial network is Δ–regular. This will bring out to both the nature of the interaction of the palettes due to the "nibbling" feature of the colouring, and how that determines the decay law. With a high probability analysis using a martingale, we show that the concerned random variables are sharply concentrated around their means. Thus the graph continues to remain almost Δ_k–regular at each stage $k \geq 0$. Finally, we indicate how to remove the assumptions of uniformity made at the outset.

4.2 The Regular Case

Let us start by assuming that the graph is initially Δ–regular, and that it retains symmetry between vertices at each stage. Thus, at each stage $k \geq 0$, each vertex has some degree Δ_k which is also the size of its palette, and the common palette between any two neighbouring vertices also has the same value uniformly, which we denote by Θ_k. The probability that an edge is successfully coloured at stage k is

$$p_k = 2\epsilon(1-\epsilon)\frac{\Theta_k}{\Delta_k}e^{-4\epsilon} = \alpha\frac{\Theta_k}{\Delta_k}.$$

where we define $\alpha = \alpha(\epsilon) := 2\epsilon(1-\epsilon)e^{-4\epsilon}$. (The factor $2\epsilon(1-\epsilon)$ is the probability that the edge is chosen by exactly one endpoint. The fraction Θ_k/Δ_k is the probability that the tentative color chosen is present at the other endpoint and,

$e^{-4\epsilon}$ is the probability that there is no color conflict.) Hence, we have for $k \geq 0$ and any vertex u,

$$\begin{aligned} E[\Delta_{k+1} \mid \Delta_k, \Theta_k] &= \Delta_k - \sum_{w \in N_k(u)} \alpha \frac{\Theta_k}{\Delta_k} \\ &= \Delta_k - \alpha\Theta_k \\ &= \Delta_k(1 - \alpha\frac{\Theta_k}{\Delta_k}) \\ &= \Delta_k(1 - \alpha\eta_k) \\ &\approx \Delta_k e^{-\alpha\eta_k} \end{aligned} \tag{1}$$

where we put $\eta_k := \frac{\Theta_k}{\Delta_k}$.

To compute $E[\Theta_{k+1}|\Delta_k, \Theta_k]$ we make use of the "nibbling" feature of the colouring process. For each edge (u, v) and color c,

$$\begin{aligned} \Pr[c \in A^u_{k+1} \cap A^v_{k+1}|\Delta_k, \Theta_k] &= 1 - \Pr[c \notin A^u_{k+1} \cap A^v_{k+1}|\Delta_k, \Theta_k] \\ &= 1 - (\Pr[c \notin A^u_{k+1}|\Delta_k, \Theta_k] + \Pr[c \notin A^v_{k+1}|\Delta_k, \Theta_k]) \\ &\quad + \Pr[c \notin A^u_{k+1}, c \notin A^v_{k+1}|\Delta_k, \Theta_k] \\ &\approx 1 - (\Pr[c \notin A^u_{k+1}|\Delta_k, \Theta_k] + \Pr[c \notin A^v_{k+1}|\Delta_k, \Theta_k]) \\ &\approx (1 - \Pr[c \notin A^u_{k+1}|\Delta_k, \Theta_k])(1 - \Pr[c \notin A^v_{k+1}|\Delta_k, \Theta_k]) \\ &= \Pr[c \in A^u_{k+1}|\Delta_k, \Theta_k]\Pr[c \in A^v_{k+1}|\Delta_k, \Theta_k] \end{aligned}$$

(since $\Pr[c \notin A^u_{k+1}, c \notin A^v_{k+1}|\Delta_k, \Theta_k] = O(\epsilon^2) = \Pr[c \notin A^u_{k+1}|\Delta_k, \Theta_k]\Pr[c \notin A^v_{k+1}|\Delta_k, \Theta_k]$). Thus, the "nibbling" feature of the colouring process is such that the common palette $A^u_{k+1} \cap A^v_{k+1}$ evolves as if it were the intersection of two palettes evolving independently of each other. Thus for an edge (u, v) at stage $k \geq 0$,

$$\begin{aligned} E[\Theta_{k+1} \mid \Delta_k, \Theta_k] &\approx \Theta_k - \sum_{w \in N_k(u)} \alpha(\frac{\Theta_k}{\Delta_k})^2 - \sum_{w \in N_k(v)} \alpha(\frac{\Theta_k}{\Delta_k})^2 \\ &= \Theta_k(1 - 2\alpha\frac{\Theta_k}{\Delta_k}) \\ &= \Theta_k(1 - 2\alpha\eta_k) \end{aligned} \tag{2}$$

It is important to note here the factor $(\frac{\Theta_k}{\Delta_k})^2$ – this arises because when a colour is selected for an edge neighbouring (u, v), it must be in the common palette of *both* edges. Then, ¿from (2) it follows that

$$\begin{aligned} E[\Theta_{k+1}] &= E[E[\Theta_{k+1} \mid \Delta_k, \Theta_k]] \\ &= (1 - 2\alpha\eta_k)E[\Theta_k] \\ &\approx e^{-2\alpha\eta_k}E[\Theta_k]. \end{aligned} \tag{3}$$

Let us write $\eta_k := \frac{\Theta_k}{\Delta_k} \approx \frac{E[\Theta_k]}{E[\Delta_k]}$; we will justify this shortly by showing that the r.v.s Δ_k and Θ_k are sharply concentrated at their means. Thus from (1) and (3),

if follows that [5]

$$\eta_{k+1} = \eta_k e^{-\alpha\eta_k}. \tag{4}$$

This recurrence is well–studied, see for instance [5, § 8.5]. We have that

$$E[\Delta_k] = \Delta \exp(-\alpha \sum_{i \le k} \eta_i)$$

It can be verified that $E[\Delta_k] \le \lambda\Delta$ whenever $k \ge k_0 := (\frac{\log(1/\lambda)}{\alpha\lambda})$. In computing the expectations above, we assumed that the graph was Δ_k–regular at stage k. Even if we assume the initial graph is Δ–regular, it will not remain regular at later stages due to statistical fluctuations. However, we shall now refine the argument by a high–probability analysis and show that the random variables Δ_k^u (denoting the size of the palette of vertex u and also its degree) and $\Theta_k^{u,v}$ (denoting the common palette size $|A_k^u \cap A_k^v|$) are each sharply concentrated around their means computed in the last section. Thus, the graph does remain "almost" regular at each stage. We shall let $E\Theta_k$ and $E\Delta_k$ be the recurrences determined by

$$E\Theta_0 := \Delta, E\Theta_{k+1} = e^{-2\alpha\eta_k} E\Theta_k,$$

and

$$E\Delta_0 := \Delta, E\Delta_{k+1} = e^{-\alpha\eta_k} E\Delta_k.$$

where, as before, η_k is the sequence determined by the recurrence (4) (with $\eta_0 := 1$). We will show that for each vertex u and each edge (u, v), with high probability,

$$(1 - \delta_k)E\Delta_k \le \Delta_k \le (1 + \delta_k)E\Delta_k,$$

and

$$(1 - \delta_k)E\Theta_k \le \Theta_k \le (1 + \delta_k)E\Theta_k,$$

(for a sequence δ_k to be specified). In this sense, if we start with a graph which is Δ–regular, it remains "almost" regular as we progress through the stages. We shall prove these statements by induction on k; they are trivially true at the start for $k = 0$. The number of edges coloured at any stage around a given vertex or edge is the sum of indicator random variables which are 1 with the probability computed earlier. We would like to use large deviation bounds to show that this sum is sharply concentrated around its mean. However, these random variables are manifestly *not* independent, and we cannot employ the usual Chernoff bound. However due to the nature of the association of the r.v.s in our case, we are able to salvage the Chernoff bound nevertheless. The most efficient way of doing this is to to use the following martingale argument sometimes called "the method of bounded differences" [15]:

[5] We use here our approximation that $\frac{1-x}{1-y} \approx 1 + y - x$ and so strictly should write $\approx$. However, since $e^{-(1+\epsilon)x} \le 1 - x \le e^{-x}$ for any $\epsilon > 0$ if x is sufficiently small, we can use $=$ with the tacit understanding that one can substitute these exact inequalities if required.

Proposition 2 (Method of Bounded Differences). *Let $X := X_1, \ldots, X_m$ be independent random variables with X_k taking values in a set A_k. Suppose the measurable function $f : \prod_k A_k \to \mathsf{R}$ satisfies*

$$|f(X) - f(X')| \leq c_k,$$

whenever X and X' differ only in the kth co-ordinate for each $k \in [m]$ and for some constants $c_1, \ldots, c_m$. Then, for any $t > 0$,

$$\Pr[|f(X) - E[f(X)]| > t] \leq 2\exp(-2t^2/\sum_k c_k^2).$$

To apply this proposition to compute the palette size of a given vertex or edge after an arbitrary stage $k \geq 0$, we proceed as follows. Let us consider the edge palettes. Fix a certain order of considering the vertices, and think of the random colouring process at stage k as determining the tentative colour assignments to edges in order corresponding to the vertices they are incident on. For an edge e incident on a fixed vertex u, let X_e be the tentative colour it is assigned at this stage, provided it is also available at the other endpoint and suffers no conflict at that endpoint (we can think of each X_e being a special colour $\perp$ at the start, thus the sets A_k in the proposition are each $[\Delta] \cup \{\perp\}$). By the properties of the algorithm, these variables (which are $|A_k^u|$ in number) are indeed independent. The function f, we choose is the size of the resulting edge palette, under these choices at stage k. It can be verified that this function has the "bounded difference" property with each $c_k := 2$. Given Δ_k, Θ_k, we have computed the expectations before. Now, applying the Chernoff bound shows that given Δ_k, Θ_k, we have with high probability (namely that given above), for any $0 < \delta < 1$,

$$e^{-2\alpha}\Theta_k(1-\delta) \leq \Theta_{k+1} \leq e^{-2\alpha}\Theta_k(1+\delta),$$

and inductively assuming high probability bounds on Θ_k, this implies that

$$E\Theta_{k+1}(1-\delta)(1-\delta_k) \leq \Theta_{k+1} \leq E\Theta_{k+1}(1+\delta)(1+\delta_k).$$

or,

$$E\Theta_{k+1}e^{-\delta_k-\delta} \leq \Theta_{k+1} \leq E\Theta_{k+1}e^{\delta_k+\delta}.$$

So, taking $\delta_k := k\delta$ verifies the inductive claim. Similarly for Δ_k, with high probability,

$$E\Delta_k(1-\delta_k) \leq \Delta_k \leq E\Delta_k(1+\delta_k).$$

The analysis is exactly the same for the vertex palettes. The failure probability is pessimistically estimated as $k_0 n$ times the failure probability at a vertex at the last stage k_0. This in turn is given by Proposition 2. We are interested only in vertices which have degree at least $\lambda\Delta$ at this stage. Noting that $\Delta_{k+1} \geq e^{-2\epsilon}\Delta_k$ for any $k \geq 0$, we get that the failure probability is at most

$$2nk_0\exp(-\lambda^2\Delta).$$

The statement on the failure probability in Theorem 1 follows ¿from this (recall that we assume $\Delta = \Omega(\log^{1+d} n)$ for some $d > 0$).

4.3 Removing the Regularity Assumption

In this section, we outline how to remove the assumption that the graph is initially Δ-regular. Note that because we want our algorithm to work in a truly distributed fashion, we cannot assume that the maximum degree is known to all vertices. As we shall demonstrate below, the essential feature of the interaction of the palettes (namely the locality) and the decay law obeyed by the palettes continues to hold without the regularity assumption. Let $\eta_k^{\max}$ be determined by the recurrence relations

$$\eta_0^{\max} := \max_{u,v} \frac{\Theta_0^{u,v}}{\Delta_0^u},$$

and

$$\eta_{k+1}^{\max} = \eta_k^{\max} \exp(-\alpha \eta_k^{\max}).$$

Similarly, let $\eta^{\min}$ be determined by the corresponding recurrence with max replaced by min. It is easy to verify by induction that for each $k \geq 0$,

$$1 \leq \eta_k^{\max}/\eta_k^{\min} \leq \eta_0^{\max}/\eta_0^{\min}.$$

Let us now write down the equation corresponding to (2) in the non-regular setting (with Θ_k, Δ_k denoting the vector of the random variables at stage k). Recall that the algorithm sets $\Delta_0^u = \max_{w \in N(u) \cup \{u\}} \deg(w)$ initially and that $\Theta_0^{u,v} = \min(\Delta_0^u, \Delta_0^v)$.

$$\begin{aligned} E[\Theta_{k+1}^{u,v} \mid \Theta_k, \Delta_k] &= \Theta_k^{u,v} - \alpha \sum_{w \in N_k(u)} (\frac{\Theta_k^{u,w}}{\Delta_k^u})^2 + (\frac{\Theta_k^{u,w}}{\Delta_k^w})^2 - \alpha \sum_{w \in N_k(v)} (\frac{\Theta_k^{v,w}}{\Delta_k^v})^2 + (\frac{\Theta_k^{v,w}}{\Delta_k^w})^2 \\ &= \Theta_k^{u,v} - \alpha \sum_{w \in N_k(u)} (\eta_k^{u,w})^2 + (\eta_k^{w,u})^2 - \alpha \sum_{w \in N_k(v)} (\eta^{v,w})^2 + (\eta^{w,v})^2 \end{aligned}$$

Now, once again, inductively, we have for any two edges (u,v) and (u',v'),

$$\frac{\eta_0^{\min}}{\eta_0^{\max}} \leq \frac{\eta_k^{u,v}}{\eta_k^{u',v'}} \leq \frac{\eta_0^{\max}}{\eta_0^{\min}}.$$

Using this in the previous equation, we get:

$$\Theta_k^{u,v} - \alpha \frac{\eta_0^{\max}}{\eta_0^{\min}} \sum_{w \in N_k(u)} (\eta_k^{u,v})^2 + (\eta_k^{v,u})^2 - \alpha \frac{\eta_0^{\max}}{\eta_0^{\min}} \sum_{w \in N_k(v)} (\eta^{v,u})^2 + (\eta^{u,v})^2 \leq E[\Theta_{k+1}^{u,v} \mid \Delta_k, \Theta_k].$$

and

$$E[\Theta_{k+1}^{u,v} \mid \Delta_k, \Theta_k] \leq \Theta_k^{u,v} - \alpha \frac{\eta_0^{\min}}{\eta_0^{\max}} \sum_{w \in N_k(u)} (\eta_k^{u,v})^2 + (\eta_k^{v,u})^2 - \alpha \frac{\eta_0^{\min}}{\eta_0^{\max}} \sum_{w \in N_k(v)} (\eta^{u,v})^2 + (\eta^{v,u})^2.$$

Thus,

$$\Theta_k^{u,v}(1-\alpha') \leq E[\Theta_{k+1}^{u,v} \mid \Delta_k, \Theta_k] \leq \Theta_k^{u,v}(1-\alpha'').$$

We are thus back in essentially the same situation as before and we can refine the calculation of expected values into a high probability argument as before. Finally we have proved:

Theorem 3. *For any fixed $\lambda > 0$, given a graph with n vertices and maximum degee Δ, we can edge colour the graph with $(1+\lambda)\Delta$ colours in time $O(\log n)$ where n is the number of vertices in the graph. For any fixed positive integer s, we can edge colour it with $\Delta + \Delta/\log^s \Delta = (1+o(1))\Delta$ colours in time $O((\log \Delta)^s \log\log \Delta + \log n)$. The results hold with failure probability decreasing to 0 faster than any polynomial (in n) provided that $\Delta = \Omega(\log^{1+d} n)$ for some $d > 0$.*

REMARK: It is unlikely that one can improve the above analysis to get a colouring better than the $\Delta + \Delta/(\log \Delta)^s$ bound above, while still retaining a poly–logarithmic running time (in n and Δ). To see this, recall from the intuitive outline in § 4.1, that even if we assume that the initial graph is regular and that the palettes evolve as truly random independent subsets, the decay law has the form

$$\Delta_{k+1} \leq \exp(-\epsilon \frac{\Delta_k}{\Delta})\Delta_k.$$

If η_k is the determined by the recurrence

$$\eta_{k+1} := e^{-\alpha\kappa\eta_k}\eta_k,$$

then one can show (see for instance, [5, § 8.5]) that $\eta_k \geq 1/k$. So, if the shrinking of a vertex degree is governed by an equation of the form $\eta_{k+1} := e^{-\alpha\kappa\eta_k}\eta_k$ the number of iterations needed to bring the degree down to $\Delta/g(\Delta)$ is $k(\Delta) = \Omega(g(\Delta))$.

Acknowledgments

We are indebted to Noga Alon and Joel Spencer for their suggestion that the Rödl nibble should work, and to Noga Alon for pointing out a mistake in a previous draft of this work and for several suggestions. We are grateful to Jirka Matousek who independently also suggested the use of the Rödl Nibble and who gave us a nice explanation of the swift working of the nibble. Kurt Melhorn provided, as usual, insightful remarks. We also thank Sem Borst and Marco Combe for useful discussions. The first author is grateful to the CWI and the Altec project for making possible an extended visit to CWI. The second author acknowledges the generous hospitality of MPI and BRICS.

References

1. N. Alon. Private Communication.
2. N. Alon, J. Spencer, and P. Erdős. *The Probabilistic Method.* Wiley–Interscience Series, John Wiley & Sons, Inc., New York, 1992.
3. B. Berger and J. Rompel. Simulating $(\log^c n)$-wise independence in NC. *J. Assoc. Comput. Mach.*, 38(4):1026–1046, 1991.
4. B. Bollobás. *Graph Theory.* Springer Verlag, New York, 1979.
5. N.G. de Bruijn. *Asymptotic methods in Analysis.* Number 4 in Bibliotheca Mathematics. North Holland Publishing Co., 1958.

6. R. Jain D. Durand and D. Tseytlin. Distributed scheduling algorithms to improve the performance of parallel data transfers. Technical Report 94–38, DIMACS, 1994.
7. Z. Galil and D. Leven. NP-completeness of finding the chromatic index of regular graphs. *J. of Algorithms*, 4:35–44, 1983.
8. I. Holyer. The NP-completeness of edge coloring. *SIAM J. Comp.*, 10:718–720, 1981.
9. R. Jain, K. Somalwar, J. Werth, and J. C. Browne. Scheduling parallel i/o operations in multiple bus systems. *Journal of Parallel and Distributed Computing*, 16(4):352–362, 1992.
10. J. Kahn. Coloring nearly-disjoint hypergraphs with $n + o(n)$ colors. *J. Comb. Theory, Series A*, 59:31–39, 1992.
11. H. J. Karloff and D. B. Shmoys. Efficient parallel algorithms for edge coloring problems. *Journal of Algorithms*, 8:39–52, 1987.
12. R. M. Karp. Probabilistic recurrence relations. In *Proceedings of the ACM Symposium on Theory of Computing*, pages 190–197, 1991.
13. M. Luby. Removing randomness in parallel computation without a processor penalty. In *Proceedings of the IEEE Symposium on Foundations of Computer Science*, pages 162–173, 1988. To appear in a special issue of *Journal of Computer and System Sciences*, devoted to FOCS 1988.
14. N. A. Lynch. Upper bounds for static resource allocation in a distributed system. *Journal of Computer and System Sciences*, 23:254–278, 1981.
15. C. McDiarmid. On the method of bounded differences. In J. Siemons, editor, *Surveys in Combinatorics*, volume 141 of *London Math. Soc. Lecture Notes Series*, pages 148–188. Cambrideg University Press, 1989.
16. R. Motwani, J. Naor, and M. Naor. The probabilistic method yields deterministic parallel algorithms. In *Proceedings of the IEEE Symposium on Foundations of Computer Science*, pages 8–13, 1989.
17. A. Panconesi and A. Srinivasan. Fast randomized algorithms for distributed edge coloring. In *Proceedings of the ACM Symposium on Principles of Distributed Computing*, pages 251–262, 1992.
18. A. Panconesi and A. Srinivasan. Improved distributed algorithms for coloring and network decomposition problems. In *Proceedings of the ACM Symposium on Theory of Computing*, pages 581–592, 1992.
19. N. Pippinger and J. Spencer. Asymptotic behaviour of the chromatic index for hypergraphs. *J. Combinatorial Theory, Series A*, 51:24–42, 1989.
20. V. Rödl. On a packing and covering problem. *European Journal of Combinatorics*, 5:69–78, 1985.
21. J. Spencer. Asymptotically Good Coverings. *Pacific Journal of Mathematics*, 118(2):575–586, 1985.