BRICS RS-96-11 Dubhashi et al.: Near-Optimal, Distributed Edge Colouring via the Nibble Method

BRICS

Basic Research in Computer Science

Near-Optimal, Distributed Edge Colouring via the Nibble Method

Devdatt Dubhashi
David A. Grable
Alessandro Panconesi

BRICS Report Series RS-96-11

ISSN 0909-0878 May 1996

See back inner page for a list of recent publications in the BRICS Report Series. Copies may be obtained by contacting:

**BRICS
Department of Computer Science
University of Aarhus
Ny Munkegade, building 540
DK - 8000 Aarhus C
Denmark**

**Telephone: +45 8942 3360
Telefax: +45 8942 3255
Internet: BRICS@brics.dk**

BRICS publications are in general accessible through WWW and anonymous FTP:

`http://www.brics.dk/`
`ftp ftp.brics.dk (cd pub/BRICS)`

Near-optimal, distributed edge colouring via the nibble method *

Devdatt Dubhashi† BRICS, Aarhus University Denmark

David A. Grable‡ Informatik, HU Berlin Germany

Alessandro Panconesi§ Informatik, FU Berlin Germany

April 1996

Abstract

We give a distributed randomized algorithm to edge colour a network. Let G be a graph with n nodes and maximum degree Δ. Here we prove:

- If $\Delta = \Omega(\log^{1+\delta} n)$ for some $\delta > 0$ and $\lambda > 0$ is fixed, the algorithm almost always colours G with $(1+\lambda)\Delta$ colours in time $O(\log n)$.
- If $s > 0$ is fixed, there exists a positive constant k such that if $\Delta = \Omega(\log^k n)$, the algorithm almost always colours G with $\Delta + \Delta/\log^s n = (1+o(1))\Delta$ colours in time $O(\log n + \log^s n \log\log n)$.

By "almost always" we mean that the algorithm may fail, but the failure probability can be made arbitrarily close to 0.

The algorithm is based on the nibble method, a probabilistic strategy introduced by Vojtěch Rödl. The analysis makes use of a powerful large deviation inequality for functions of independent random variables.

1 Introduction

The edge colouring 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 colouring problems [12, 14, 17, 20]. In a distributed setting, the edge colouring problem can be used to model certain types of jobshop scheduling, packet routing, and resource allocation problems. For example, the problem of scheduling I/O operations in some parallel architectures can be modeled as follows [12, 7]. 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 constraint it is not hard to see that optimal edge colourings 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 colours are needed to edge colour the graph. A classical theorem of Vizing shows that $\Delta + 1$ colours are always sufficient, and the

*A preliminary version of this paper was presented at ESA '95. The paper has been invited to be published in a special issue of *Theoretical Computer Science* devoted to the proceedings of ESA '95.

†This work was partly done when at Max Planck Institute, Saarbrücken.

‡Supported by Deutsche Forschungsgemeinschaft project number Pr 296/4-1.

§Supported by an Alexander von Humboldt research fellowship. This work was partly done when at CWI Amsterdam, with financial support provided by an ERCIM postdoctoral fellowship.

proof is actually a polynomial time algorithm to compute such a colouring (see for example [5]). Interestingly, given a graph G, it is NP-complete to decide whether it is Δ or $\Delta+1$ edge colourable [11], even for regular graphs [9]. Efforts at parallelizing Vizing's theorem have failed; the best PRAM algorithm known is a randomized algorithm by Karloff & Shmoys that computes an edge colouring using very nearly $\Delta + \sqrt{\Delta} = (1 + o(1))\Delta$ colours. The Karloff & Shmoys algorithm can be derandomized by using standard derandomization techniques [4, 19]. Whether $(\Delta + 1)$–edge colouring is P-complete is an open problem. In the distributed setting the previously best known result was a randomized algorithm by Panconesi & Srinivasan that uses roughly $1.58\Delta + \log n$ colours with high probability and runs in $O(\log n)$ time with high probability, provided the input graph as "large enough" maximum degree. Precisely, it must satisfy the condition $\Delta = \Omega(\log^{1+\delta} n)$, where $\delta > 0$ is any positive real. For the interesting special case of bipartite graphs Lev, Pippinger & Valiant show that Δ-colourings can be computed in NC, whereas this is provably impossible in the distributed model of computation even if randomness is allowed (see [21]).

In this paper, we vastly improve on the previous state-of-the-art by giving a distributed randomized algorithm that computes a near-optimal edge colouring in time $O(\log n)$, provided the maximum degree is "large enough". More precisely, let G be a graph with n nodes and maximum degree Δ. In this paper, we prove the following.

- If $\Delta = \Omega(\log^{1+\delta} n)$ for some $\delta > 0$ and $\lambda > 0$ is fixed, the algorithm almost always colours G with $(1 + \lambda)\Delta$ colours in time $O(\log n)$.
- If $s > 0$ is fixed, there exists a positive constant k such that if $\Delta = \Omega(\log^k n)$, the algorithm almost always colours G with $\Delta + \Delta/\log^s n = (1 + o(1))\Delta$ colours in time $O(\log n + \log^s n \log\log n)$.

The statements on the number of colours used and the running time hold with high probability, meaning that the failure probability is $o(1)$, a quantity that goes to 0 as n grows. We note that while the first result requires no global knowledge to be stored at the vertices, the second one requires the vertices to know either the value of Δ or of n, neither of which might be readily available in a truly distributed system. The algorithm can be implemented in the PRAM model of computation at a cost of an extra $O(\log \Delta)$ factor in the running time, which is needed to simulate the message-passing mechanism of a distributed network.

Our algorithm is based on the *Rödl Nibble*, a beautiful probabilistic strategy introduced by Vojtech Rödl to solve a certain covering problem in hypergraphs [3, 23, 8]. The method has subsequently been used very successfully to solve other combinatorial problems such as asymptotically optimal coverings and colourings for hypergraphs [3, 13, 22, 24]. In this paper, we introduce the nibble as a tool for the design and analysis of randomized algorithms.[1] 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. The main problem of the analysis is that the random variables of interest turn out to be dependent, due to the interaction along the edges of the graph. To carry out the analysis we make use of a new martingale inequality which the second author recently developed, improving on results of Kim [16] and Alon, Kim and Spencer [2]. The inequality provides a methodology for proving sharp concentration results for not necessarily dependent random variables which yields clean and conceptually simple proofs. We expect this method to be widely applicabile in randomized

[1] This research was originally prompted by a conversation that the third author had with Noga Alon and Joel Spencer, in which they suggested that the nibble approach should work. Noga Alon has 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.

algorithms and we regard this paper as a non-trivial demonstration of its power. The high probability analysis is further simplified by the use of the nibbling feature which, intuitively, keeps the dependency between the random variables low.

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*, *i.e.* 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 done locally by each processor must be charged for.

An *edge colouring* of a graph G is an assignment of colours to edges such that incident edges always have different colours. The edge colouring problem is to find an edge colouring with the aim of minimizing the number of colours used. Given that determining an optimal (minimal) colouring is an NP-hard problem this requirement is usually relaxed to consider approximate, hopefully near-optimal, colourings. The edge colouring problem in a distributed setting is formulated as follows: a distributed network G wants to compute an edge colouring of its own topology. As remarked in the introduction, such a colouring might be useful in the context of scheduling and resource allocation.

We will make use of the following trivial algorithm. Each edge $e = uv$ is initially given a palette of $\deg(u) + \deg(v)$ colours. The computation takes place in rounds; in each round, each uncoloured edge independently picks a tentative colour uniformly at random from its current palette. If no neighbouring edge picks the same colour, it becomes final. Otherwise, the edge tries again in the next round. At the end of each round the palettes are updated in the obvious way: colours successfully used by neighboring edges are deleted from the current palette. Notice that each edge need only communicate with its neighbors. Henceforth, we will refer to this as the *trivial algorithm*. Elementary calculations show that the probability that an edge colours itself at each round is never less than a constant of value $e^{-2}(1 + o(1))$. It follows by well-known results on probabilistic recurrence relations that with high probability every edge is coloured within $O(\log n)$ rounds [6, 15].

Notation When we write $a \sim b$, we mean $a = b(1 + o(1))$. The set $\{1, 2, \ldots, n\}$ will be denoted by $[n]$.

3 A Large Deviation Inequality

A key ingredient of our proof is a large deviation inequality for functions of independent random variables, which was recently developed by the second author. Please see [10] for a proof, a more general result, and further discussion.

Assume we have a probability space generated by independent random variables X_i (choices), where choice X_i is from the finite set A_i, and a function $Y = f(X_1, \ldots, X_n)$ on that probability space. We are interested in proving a sharp concentration result on Y, *i.e.* to bound $\Pr[|Y - \mathrm{Ex}[Y]| > a]$, for any a, as well as we can. The well-known Chernoff-Hoeffding bounds give essentially best possible estimates when $Y = \sum_i X_i$. The Method of Bounded Differences (MOBD), a nicely packaged generalization of a martingale inequality known as Azuma's inequality, allows one to consider any function $f(X_1, \ldots, X_n)$ under the additional "bounded difference" requirement that changing the choice of one of the variables does not affect the final value of Y by too much [18]. More precisely, the result states that if, for all vectors A and B differing only in the i-th coordinate,

$$|f(A) - f(B)| \leq c_i$$

then

$$\Pr\left[|Y - \mathrm{Ex}[Y]| > \sqrt{\varphi \sum_i c_i^2/2}\right] \leq 2e^{-\varphi}. \tag{1}$$

This result was significantly strengthened by Kim for the case of 0–1 random variables and further generalized by Alon, Kim and Spencer [16, 2]. The result we discuss is a further generalization of this last paper. The idea is that much can be gained by determining the effect of changing each X_i in a dynamic way, instead of statically as in the MOBD.

Consider the following "query game," the aim of which is to determine the value of Y. We can ask queries of the form "what was the i-th choice?"—*i.e.* "what was the choice of X_i?"—in any order we want. The answer to the query is the random choice of X_i. The questioning can be adaptive, *i.e.* we can chose the next X_i to be queried as a function of the knowledge gained so far. The effect of changing X_i's value on the final value of Y is estimated at the time X_i is queried. The advantage of this framework is that, once some choices are exposed, many random variables, which at the outset could potentially affect Y significantly, do not anymore. As a result, we can substitute for $\sum_i c_i^2$ in equation 1 an estimate on the variance of Y that is in many cases much better than $\sum_i c_i^2$. The high probability analyis contained in this paper gives non-trivial examples where the MOBD would be awkward to use or simply too weak.

We now state the result precisely. A *querying strategy* for Y is a decision tree whose internal nodes designate queries to be made. Each node of the tree represents a query of the type "what was the random choice of X_i?". A node has as many children as there are random choices for X_i. It might be helpful to think of the edges as labeled with the particular $a \in A_i$ corresponding to that random choice. In this fashion, every path from the root to a node which goes through vertices corresponding to $X_{i_1}, \ldots, X_{i_k}$ defines an assignment $a_1, \ldots, a_k$ for these random variables. We can think of each node as storing the value $\mathrm{Ex}[Y|X_{i_1} = a_1 \ldots X_{i_k} = a_k]$. In particular, the leaves store the possible values of Y, since by then all relevant random choices have been determined.

Define the *variance* of a query (internal node) q concerning choice i to be

$$v_q = \sum_{a \in A_i} p_{i,a}\mu_{q,a}^2,$$

where

$$p_{i,a} = \Pr[\text{choice } i \text{ was } a]$$

and

$$\mu_{q,a} = \mathrm{Ex}[Y \mid \text{choice } i \text{ was } a \text{ and all previous queries}] - \mathrm{Ex}[Y \mid \text{all previous queries}].$$

In words, $\mu_{q,a}$ measures the amount which our expectation changes when the answer to query q is revealed to be a.

Also define the *maximum effect* of query q as

$$c_q = \max_{a,b \in A_i} |\mu_{q,a} - \mu_{q,b}|.$$

A way to think about c_q is the following. Consider the children of node q; c_q is the maximum difference between any values $\mathrm{Ex}[Y \mid \text{all previous queries}]$ stored at the children. In the sequel, we will often compute an upper bound on c_q for instance, by taking the maximum amount which Y can change if choice i is changed, but all other choices remain the same. In other words, to compute c_q we consider the subtree rooted at q and consider the maximum difference between any two values stored at the leaves of this subtree. As we shall see, in practice good upper bounds on c_q are very easy to obtain.

A *line of questioning* ℓ is a path in the decision tree from the root to a leaf and the *variance* of a line of questioning is the sum of the variances of the queries along it. Finally, the *variance* of a strategy $\mathcal{S}$ is the maximum variance over all lines of questioning

$$V(\mathcal{S}) = \max_{\ell} \sum_{q \in \ell} v_q.$$

The use of the term variance is meant to be suggestive: $V(\mathcal{S})$ is an upper bound on the variance of Y. The variance plays essentially the same role as the term $\sum_i c_i^2$ in the MOBD, but it is a much better upper bound to the variance of Y.

Proposition 1 *If there is a strategy for determining Y with variance at most V then*

$$\Pr\left[|Y - \mathrm{Ex}[Y]| > 2\sqrt{\varphi V}\right] \leq 2e^{-\varphi}$$

for every $0 \leq \varphi \leq V / \max c_q^2$.

Besides its "dynamic" aspect, another fact that makes the query game particularly powerful is that the the probability that a certain choice happens can be factored in when upper bounding the variance. When upper bounding

$$v_q = \sum_{a \in A_i} p_{i,a} \mu_{q,a}^2$$

it is often possible to partition the space A_i in two regions, the **Yes** region and the **No** region corresponding to two mutually exclusive events, the **Yes** and **No** events, which cover the whole space. In this paper, for instance, the A_i will be a set of colours and the **Yes** event will often be of the form "was choice i colour α?". If we denote by $p_{Y,q}$ the probability of the **Yes** event occurring, and if we know that the μ's differ by at most c_q, then

$$v_q = \sum_{a \in A_i} p_{i,a} \mu_{q,a}^2 \leq p_{Y,q}(1 - p_{Y,q}) c_q^2 \leq p_{Y,q} c_q^2. \tag{2}$$

This bound can be verified using elementary, but non-trivial, computations, which we omit.

4 The Algorithm

The algorithm runs in two phases. The first phase is an application of the Rödl nibble algorithm. It has the goal of colouring most of the edges using a palette of Δ colours. Starting with the input graph G_0 the algorithm generates a sequence $G_0, G_1, \ldots, G_{t_\epsilon}$ of graphs, where G_i is the graph induced by the edges still uncoloured at stage i. Each edge e has a palette of available colours—initially the whole set of Δ colours. At each stage, a small ϵ fraction of uncoloured edges is selected and each selected edge chooses a tentative colour at random from its current palette. If the tentative colour is not chosen by any neighboring edge, it becomes final. Palettes of the remaining uncoloured edges are updated in the obvious fashion—by deleting colours used by neighboring edges. The process is then repeated. A key idea of the method is that if colours are chosen independently, the probability of colour conflict is roughly ϵ^2, a negligible fraction of all edges attempting colouring at this stage. If the same "efficiency" is maintained throughout, the overall "wastage" will be very small. Another aspect of the method, which requires a non-trivial high probability analysis, is that the graphs G_i and the colour palettes evolve, respectively, "almost" as truly random subgraphs of the original graph and truly random subsets of the original palettes.

We run the first phase until the remaining graph has with high probability maximum degree at most $\epsilon\Delta(1+o(1))$. The algorithm then switches to the trivial algorithm described in Section 2, which gives each edge at most $2\epsilon\Delta(1+o(1))$ fresh colours and colours the remaining graph in $O(\log n)$ rounds with high probability. The total number of colours used by the algorithm is therefore at most $(1+2\epsilon)\Delta(1+o(1))$, which can be made as small as $(1+\lambda)\Delta$, for any (not necessarily fixed) $\lambda > 0$ by choosing a sufficiently small $\epsilon > 0$.

As we shall see in section 5, the number of iterations needed to bring the degree down from Δ to $\epsilon\Delta(1+o(1))$ is

$$t_\epsilon := \frac{1}{p_\epsilon} \log \frac{1}{\epsilon}$$

where $p_\epsilon = \epsilon(1-\epsilon/4)e^{-2\epsilon(1-\epsilon/4)}$. The total running time of the algorithm is then

$$O(t_\epsilon + \log n)$$

(and $O((t_\epsilon + \log n)\log\Delta)$ on a PRAM).

Note that in order to get a $(1+\lambda)\Delta$ colouring, where $\lambda > 0$ is a *fixed* constant, the first phase takes constant time and to get a $\Delta + \Delta/\log^s n = (1+o(1))\Delta$ colouring, the first phase requires $O(\log^s n \log\log n)$ time.

The statements concerning the performance of the algorithm hold with high probability. The exact probability of success will be determined in the analysis. We note here that an assumption on the maximum degree of the graph is necessary. In general, we will require $\Delta = \Omega(\log^{1+\delta} n)$, where $\delta > 0$ is any constant. But if we use more than a constant number of stages in the first phase, the requirement on Δ becomes more stringent.

Lastly note that if we want, as in our second claim, ε to be a function of n (or Δ), it is necessary that the processors know n (or Δ) in order to be able to calculate t_ϵ.

The algorithm is as follows.

Phase 1 NIBBLE ALGORITHM

The initial graph $G_0 := G$, the input graph. Each edge $e = uv$ is initially given the palette $A_0(e) = [\max\{\deg(u), \deg(v)\}]$. (This can be arranged in one round with each vertex communicating its own degree to each of its neighbours.)

For $i = 0, 1, \ldots, t_\epsilon - 1$ stages repeat the following:

- (*Select nibble*) Each vertex u randomly selects an $\epsilon/2$ fraction of the edges incident on itself. An edge is considered selected if either or both of its endpoints selects it.
- (*Choose tentative colour*) Each selected edge e chooses independently at random a tentative colour $t(e)$ from its palette $A_k(e)$ of currently available colours.
- (*Check colour conflicts*) Colour $t(e)$ becomes the final colour of e unless some edge incident on e has chosen the same tentative colour,
- (*Update graph and palettes*) The graph and the palettes are updated by setting

$$G_{i+1} = G_i - \{e \mid e \text{ got a final colour}\}$$

and, for each edge e, setting

$$A_{i+1}(e) = A_i(e) - \{t(f) \mid f \text{ incident on } e,\ t(f) \text{ is the final colour of } f\}.$$

Phase 2 Trivial Algorithm

Each uncoloured edge $e = uv$ introduces fresh new colours in order to have $\deg_{t_\epsilon}(u) + \deg_{t_\epsilon}(v)$ colours in its current palette and runs the trivial algorithm of Section 2.

5 Analysis: The Regular Case

We first carry out the analysis for the special case of Δ-regular graphs. We will show later how the general case can be reduced to it. The crux of the analysis is to show that the sequence of graphs $G_0, G_1, \ldots, G_{t_\epsilon}$ are "more or less" random subgraphs of the original input graph and that the palettes $A_i(e)$ are "essentially" random subsets of the original palettes. In more technical terms, we need to show that the graph definied by the uncoloured edges and the palettes evolve quasi-randomly. In the analysis we control three quantities:

- $|A_i(u)|$, the implicit vertex palette at stage i. This is the set of colours available at vertex u at stage i, *i.e.* the set of colours not used by any edges incident on u. Notice that, clearly, $|A_i(u)| = d_i(u)$, the degree of vertex u at stage i.
- $|A_i(e)|$, the edge palette at stage i; and
- $d_{i,\gamma}(u)$, the number of u-neighbors which, at stage i, have γ in their palettes.

The initial values are

$$|A_0(u)| = |A_0(e)| = d_{0,\gamma}(u) = \Delta$$

for all vertices u, edges e, and colours γ. We will show that

$$\begin{aligned} |A_i(u)| &\sim d_i := (1 - p_\epsilon) d_{i-1}, && (3)\\ |A_i(e)| &\sim a_i := (1 - p_\epsilon)^2 a_{i-1}, \text{ and} && (4)\\ d_{i,\gamma}(u) &\sim d_{i,\gamma} := d_i^2 / a_0 && (5) \end{aligned}$$

for all vertices u, edges e, colours γ, and stage i, where

$$p_\epsilon = \epsilon \left(1 - \frac{\epsilon}{4}\right) e^{-2\epsilon(1-\epsilon/4)};$$

Intuitively, p_ϵ is the probability that an edge colours itself at any given stage. The initial conditions are

$$d_0 = a_0 = \Delta.$$

Notice that, conforming to intuition,

$$d_{i,\gamma} = \frac{d_i^2}{a_0} = (1 - p_\epsilon)^{2i} a_0 = a_i \qquad (6)$$

which can be interpreted as saying: $d_{i,\gamma}(u) \sim |A_i(e)|$.

Notice also that by equation (3) the maximum degree of G_{t_ϵ} is

$$\Delta(G_{t_\epsilon}) \sim d_{t_\epsilon} = (1 - p_\epsilon)^{t_\epsilon} \Delta \le \epsilon \Delta$$

as claimed in Section 4. Hence, once equation (3) is established for $i = t_\epsilon$, we are done. To do this, we prove equations (3) through (5) by induction on i. The basis case, $i = 0$, holds true with equality. In the proof of the inductive step, we assume that (4) through (5) hold true as shown (the IH) and prove the same statements with i replaced by $i+1$. Each such proof has two steps. First, we show that the equations are true in *expectation*, namely that

$$\begin{aligned} \mathrm{Ex}[|A_{i+1}(u)|] &\sim d_{i+1}, \\ \mathrm{Ex}[|A_{i+1}(e)|] &\sim a_{i+1}, \text{ and} \\ \mathrm{Ex}[d_{i+1,\gamma}(u)] &\sim d_{i+1,\gamma}. \end{aligned}$$

Then we show that these random variables are sharply concentrated around their expectations.

Remark Equations (3) and (4) show that the $\lambda = o(1)$ term relative to the number of colours used can never be smaller than $1/\sqrt{\Delta}$ because by the time $d_i \sim \lambda\Delta = \sqrt{\Delta}$, the edge palette has vanished.

In the expectation computations we will need some basic facts about some atomic events. In what follows, let $E_{i,\gamma}(u)$ and $E_{i,\gamma}(e)$ be, respectively, the set of edges incident on vertex u and on edge e which, at the end of stage i, still retain γ in their palettes.

We say that and edge *γ–colours* if it (1) is selected to be coloured, (2) chooses tentative colour γ, and (3) finds no conflicting neighbour. Clearly, and edge palette must contain γ in order for the edge to γ–colour.

Fact 1 *If $\gamma \in A_i(e)$ then* $\Pr[e\ \gamma\text{–}colours] \sim p_\epsilon/a_i$.

Proof Let $e = uv$. By induction, $|E_{i,\gamma}(e)| = d_{i,\gamma}(u) + d_{i,\gamma}(v) \sim 2d_{i,\gamma}$. Using equation (6) and the IH's (4) and (5),

$$\begin{aligned} \Pr[e\ \gamma\text{–colours}] &= \frac{\epsilon(1-\epsilon/4)}{|A_i(e)|} \prod_{f \in E_{i,\gamma}(e)} \left(1 - \frac{\epsilon(1-\epsilon/4)}{|A_i(f)|}\right) \\ &\sim \frac{\epsilon(1-\epsilon/4)}{a_i} \left(1 - \frac{\epsilon(1-\epsilon/4)}{a_i}\right)^{2d_{i,\gamma}} \\ &\sim \frac{\epsilon(1-\epsilon/4)}{a_i} e^{-2\epsilon(1-\epsilon/4)} \\ &= \frac{p_\epsilon}{a_i}. \end{aligned}$$

□

Fact 2 *If f and g are two disjoint edges (i.e. $f \cap g = \emptyset$) and $\gamma \in A_i(f) \cap A_i(g)$ then*

$$\Pr[f \textit{ and } g\ \gamma\textit{-colour}] \sim (p_\epsilon/a_i)^2.$$

Proof Let I_γ be the set of edges incident on both f and g which have γ in their palettes. Observe that $0 \le |I_\gamma| \le 4$. By the IH's (4) and (5),

$$\begin{aligned} \Pr[f \text{ and } g\ \gamma\text{-colour}] &= \frac{\epsilon(1-\epsilon/4)}{|A_i(f)|} \frac{\epsilon(1-\epsilon/4)}{|A_i(g)|} \prod_{h \in E_{i,\gamma}(f) - I_\gamma} \left(1 - \frac{\epsilon(1-\epsilon/4)}{|A_i(h)|}\right) \\ &\quad \times \prod_{h \in E_{i,\gamma}(g) - I_\gamma} \left(1 - \frac{\epsilon(1-\epsilon/4)}{|A_i(h)|}\right) \prod_{h \in I_\gamma} \left(1 - \frac{\epsilon(1-\epsilon/4)}{|A_i(h)|}\right) \\ &\sim \left(\frac{\epsilon(1-\epsilon/4)}{a_i}\right)^2 \left[\left(1 - \frac{\epsilon(1-\epsilon/4)}{a_i}\right)^{2d_{i,\gamma}}\right]^2 \\ &\sim \left(\frac{p_\epsilon}{a_i}\right)^2. \end{aligned}$$

□

5.1 Proof of $|A_{i+1}(u)| \sim d_{i+1}$

First we use the two facts and the induction hypothesis to show that the expectation of $|A_{i+1}(u)|$ is indeed $d_{i+1}(1+o(1))$. Then we use the techniques of section 3 to show that this random variable is highly concentrated about its expectation so that with high probability, equation (3) is satisfied.

Lemma 1 $\mathrm{Ex}[|A_{i+1}(u)|] \sim d_{i+1}$.

Proof First notice that

$$\Pr[e \text{ colours}] = \sum_{\gamma \in A_i(e)} \Pr[e\ \gamma\text{-colours}] \sim p_\epsilon$$

by the IH, Fact 1 and the fact that the events "e γ-colours" are disjoint. Then, by the IH,

$$\mathrm{Ex}[|A_{i+1}(u)|] = \sum_{e \ni u} (1 - \Pr[e \text{ colours}]) \sim (1-p_\epsilon)d_i = d_{i+1}.$$

□

Lemma 2 *For each fixed vertex u, $|A_{i+1}(u)|$ is within $4\sqrt{3\epsilon d_i \log n}$ of its expectation with probability at least $1 - 2n^{-2}$.*

Remark Note that an error probability less than $2n^{-2}$ suffices since we will appeal to this lemma once for each vertex u during each stage $i \ll n$. Together these two lemmas implies that with high probability

$$|A_{i+1}(u)| = \mathrm{Ex}[|A_{i+1}(u)|] \pm 4\sqrt{3\epsilon d_i \log n} = \left(1 + o(1) \pm 4\sqrt{\frac{3\epsilon \log n}{(1-p_\epsilon)d_{i+1}}}\right) d_{i+1}.$$

Therefore, to ensure that $|A_{i+1}(u)| \sim d_{i+1}$, we require only that $(\epsilon \log n)/d_{i+1} = o(1)$.

Proof Instead of proving sharp concentration bounds for $|A_{i+1}(u)|$ we will prove, equivalently, sharp concentration bounds for $Y = |A_i(u)| - |A_{i+1}(u)|$, the number of edges incident on u which successfully colour themselves.

We will describe a strategy $\mathcal{S}_Y$ to determine Y whose total variance is

$$V(\mathcal{S}_Y) < 6\epsilon d_i,$$

which will give us the claim by proposition 1 with $\varphi = 2\log n$.

The strategy is defined as follows. First, we query all edges around u, for a total of $d_i(u)$ queries. By the IH, $d_i(u) \sim d_i$. Then we query edges incident on neighbors of u; we will argue that the total number of queries in this second group is at most $d_i a_i(1+o(1))$ (a saving from the naive estimate of $d_i^2(1+o(1))$, resulting in much better asymptotics).

Let e be a u-edge. Changing e's tentative colour can affect the final value of Y by at most $c_e = 2$. For each edge e we consider the underlying {Yes, No} probability space, where the Yes event is "e was selected for tentative colouring at this stage" so that $p_{Y,e} = \epsilon(1-\epsilon/4)$.

Using bound (2) we obtain an upper bound for the variance of this query of

$$v_e \le p_{Y,e} c_e^2 = 4\epsilon(1-\epsilon/4),$$

leading to an upper bound for the total variance of this initial segment of the query line of

$$d_i(u)v_e \leq 4\epsilon(1-\epsilon/4)d_i(1+o(1)).$$

We then proceed by querying edges incident on neighbors of u, to see how many of these "half-succesful" edges—edges which have no conflicts around u—have no conflicts at the other endpoint either. Let v be a neighbour of u and let $e = uv$. At this point, we already know $t(e)$, e's tentative colour choice. Therefore, we need only query the $(d_{i,t(e)}(u) - 1)$ edges incident on v which have $t(e)$ in their palette: the remaining edges can not affect Y in any way. The total number of queries affecting the final value of Y is, using the IH,

$$\sum_{v \in N_i(u)} (d_{i,t(e)} - 1) \sim d_i d_{i,\gamma} = d_i a_i.$$

To estimate the variance, it is convenient to split the edges incident on u-neighbours into two groups. Edges of type A have only one of their endpoints as a u-neighbour, whereas edges of type B have both endpoints as u-neighbours. Focus on one type A edge f incident on a u-neighbour v; $t(e)$ is, as usual, the tentative colour choice of $e = uv$. By the previous remark we assume that $t(e) \in A_i(f)$. Changing f's tentative colour can have maximum effect of at most $c_f = 1$—either f conflicts with e or it doesn't. We consider an underlying $\{\textsc{Yes}, \textsc{No}\}$ probability space for f where the **Yes** event is "f's tentative choice was $t(e)$" so that $p_{Y,f} = \epsilon(1-\epsilon/4)/|A_i(f)| \sim \epsilon(1-\epsilon/4)/a_i$, again by the IH. Using bound (2), the variance of f's query can be upper bounded thus:

$$v_f \leq p_{Y,f}c_f^2 \sim \epsilon(1-\epsilon/4)/a_i.$$

Consider now an edge $g = vw$ of type B, let $e_1 = uv$, $e_2 = uw$. We can assume that $t(e_1) \neq t(e_2)$ since otherwise g's tentative colour would certainly not affect the final value of Y (because of the conflict between e_1 and e_2). It follows that g's maximum effect is upper bounded by 1 because g can conflict with either e_1 or e_2, but not both. The **Yes** event we consider is: "g's tentative choice was $t(e_1)$ or $t(e_2)$". Then, $p_{Y,g} = 2\epsilon(1-\epsilon/4)/|A_i(f)| \sim 2\epsilon(1-\epsilon/4)/a_i$, by the IH. Using bound (2), g's variance can be bounded thus:

$$v_g \leq p_{Y,g}c_g^2 \sim 2\epsilon(1-\epsilon/4)/a_i.$$

Using the latter as a worst case estimate and multiplying for the total number of queries, we obtain an upper bound for the second segment of the query line of

$$2d_i\epsilon(1-\epsilon/4)(1+o(1)).$$

The total variance of the strategy is therefore

$$V(\mathcal{S}_Y) \leq 6\epsilon(1-\epsilon/4)d_i(1+o(1)) < 6\epsilon d_i$$

for large enough n. □

5.2 Proof of $|A_{i+1}(e)| \sim a_{i+1}$

Henceforth, an *e-pair* of an edge $e = uv$ is a pair (f,g) of edges such that f is a u-edge and g is a v-edge. When there is no room for confusion, we will simply say "pair" instead of "e-pair". First, the expectation computation:

Lemma 3 $\mathrm{Ex}[|A_{i+1}(e)|] \sim a_{i+1}$.

Proof Let $e = uv$. An e-pair γ*–colours* if both of its edges γ-colour. First, using Fact 2, we compute $\Pr[\gamma$ decays at $e]$, the probability that some e-edge γ-colours. By inclusion-exclusion, this probability is given by the probability that some u-edge γ–colours, plus the probability that some v-edge γ–colours, minus the probability that some e-pair γ–colours. To compute this last event we define the set $P_{i,\gamma}(e)$ of all e-pairs (f, g) such that $\gamma \in A_i(f)$ and $\gamma \in A_i(g)$. Equivalently, this is the set of pairs such that $f \in E_{i,\gamma}(u)$ and $g \in E_{i,\gamma}(v)$. By the IH, for each $\gamma \in A_i(e)$, $|P_{i,\gamma}(e)| = (d_{i,\gamma}(u) - 1)(d_{i,\gamma}(v) - 1) \sim d_{i,\gamma}^2 = a_i^2$. But not all pairs in $P_{i,\gamma}(e)$ can γ-colour; pairs which form a triangle with e, *i.e.* those pairs (f, g) such that $f \cap g \neq \emptyset$. The crucial observation is that the set $T_{i,\gamma}(e)$ of triangle pairs has cardinality at most $\min\{d_{i,\gamma}(u), d_{i,\gamma}(v)\}$ which, by the IH, is at most $d_{i,\gamma}(1 + o(1)) \sim a_i$. Hence, if we define $D_{i,\gamma}(e) = P_{i,\gamma}(e) - T_{i,\gamma}(e)$,

$$|D_{i,\gamma}(e)| \sim |P_{i,\gamma}(e)| \sim a_i^2$$

by the IH. Since the events "(f, g) γ-colours" are disjoint

$$\Pr[\text{some pair } \gamma\text{-colours}] = \sum_{(f,g)\in D_{i,\gamma}(e)} \Pr[(f, g)\ \gamma\text{-colours}].$$

By Facts 1 and 2 and equation (6),

$$\begin{aligned}
\Pr[\gamma \text{ decays at } e] &= \sum_{f\in E_{i,\gamma}(u)} \Pr[f\ \gamma\text{-colours}] + \sum_{g\in E_{i,\gamma}(u)} \Pr[g\ \gamma\text{-colours}] \\
&\quad - \sum_{(f,g)\in D_{i\gamma}(e)} \Pr[f, g\ \gamma\text{-colour}] \\
&\sim 2d_{i,\gamma}\frac{p_\epsilon}{a_i} - a_i^2 \frac{p_\epsilon^2}{a_i^2} \\
&= 2p_\epsilon - p_\epsilon^2.
\end{aligned}$$

It then follows that

$$E[|A_{i+1}(e)|] = \sum_{\gamma\in A_i(e)} (1 - \Pr[\gamma \text{ decays at } e]) \sim (1 - p_\epsilon)^2 a_i = a_{i+1}.$$

□

The next lemma establishes a strong concentration result for $|A_i(e)|$ by making use of a feature of the nibble process. This results in a much simplified analysis. Roughly speaking, in the proof we are concerned about a random variable whose expectation is $O(a_i)$. A naive use of the solitaire game gives a variance of $O(d_i)$. Since $d_i/a_i = 1/(1 - p_\epsilon)^i$, a $O(d_i)$ upper bound on the variance is still very good (best possible up to constants) if the number of iterations is constant, but becomes $\Omega(n^\epsilon a_i)$ for $\Omega(\log n)$ iterations, resulting in much worse asymptotics. What we want is an $O(a_i)$ bound for the variance. This could be obtained by analyzing the behaviour of the intersection between neighbouring edge palettes. By using the same methods as in this paper it is possible to show that

$$|A_i(f) \cap A_i(g)| \sim c_i := (1 - p_\epsilon)^3 c_{i-1}$$

for any two neighbouring edges f and g. Although conceptually identical to the other lemmas, the resulting proof would be rather long and tedious. Fortunately, the nibble offers a short cut. We want to satisfy the condition $\epsilon d_i \leq a_i$ or, equivalently,

$$\epsilon \leq \frac{a_i}{d_i} = (1 - p_\epsilon)^i \leq e^{-ip_\epsilon}.$$

This is satisfied as long as

$$i \le \frac{1}{p_\epsilon} \log \frac{1}{\epsilon}.$$

Conveniently, this is exactly the number t_ϵ of iterations of Phase I of the algorithm defined in section 4.

Lemma 4 *For each fixed edge e, $|A_{i+1}(e)|$ is within $26\sqrt{a_i \log n}$ of its expectation with probability at least $1 - 2n^{-3}$.*

Remark Note that an error probability less than $2n^{-3}$ suffices since we will appeal to this lemma once for each edge e during each stage $i \ll n$. This time, the two lemmas imply that with high probability

$$|A_{i+1}(e)| = \mathrm{Ex}[|A_{i+1}(e)|] \pm 26\sqrt{a_i \log n} = \left(1 + o(1) \pm \frac{26}{1-p_\epsilon}\sqrt{\frac{\log n}{a_{i+1}}}\right) a_{i+1}.$$

So, to ensure that $|A_{i+1}(e)| \sim a_{i+1}$, we require that $(\log n)/a_{i+1} = o(1)$.

Proof Let $e = uv$. An e-pair is *monochromatic* if both of its edges choose the same tentative colour. By inclusion-exclusion, the random variable of interest is

$$\begin{aligned} Y &= \#\text{ colours eaten by } u\text{-edges} + \#\text{ colours eaten by } v\text{-edges} \\ &\qquad -\#\text{ colours eaten by monochromatic } e\text{-pairs} \\ &:= A + B - C. \end{aligned}$$

We will show that these three variables are sharply concentrated around their means. (By symmetry, the arguments for A and B are the same.) We will find query strategies for A and B with total variance at most $5a_i$ each and a strategy for C with total variance $9a_i$. This leads, by proposition 1 with $\varphi = 3\log n$, to maximum deviations of $2\sqrt{15 a_i \log n}$ for A and B and $6\sqrt{3 a_i \log n}$ for C. Summing deviations leads to a maximum deviation of $26\sqrt{a_i \log n}$ for Y and hence for $|A_{i+1}(e)|$.

First, we give a strategy for A of maximum variance

$$V(\mathcal{S}_A) \le 5a_i.$$

We start by querying the edges around u; there are at most $d_i(1+o(1))$ of these, by the IH. Changing the tentative colour of one of these edges can affect the final value of A by at most 2. Define the **Yes** event associated with each u-edge f as "f was selected for tentative colouring at this stage". Then, $p_Y = \epsilon(1-\epsilon/4)$. Using bound (2) gives the upper bound

$$v_f \le p_{Y,f} c_f^2 = 4\epsilon(1-\epsilon/4),$$

which, when multiplied by the number of queries, gives a bound for this initial segment of the strategy of

$$4d_i\epsilon(1-\epsilon/4)(1+o(1)) \le 4a_i(1-\epsilon/4)(1+o(1)).$$

As in the proof of Lemma 2, at this point we know which edges are "half-succesful"—namely, those edges which have no colour conflict around u. To determine which of these will be completely succesful, we proceed as before: for each half-succesful edge $e = uv$ with tentative colour $t(e)$, we query the $(d_{i,t(e)} - 1)$–many edges incident on the u-neighbour v which have $t(e)$ in their palette (only these can affect the final value of A). The total number of such queries is at most

$$d_i d_{i,t(e)}(1+o(1)) \sim d_i a_i.$$

Each of these edges f can affect A only by conflicting with e; therefore, f's maximum effect is at most 1 and $p_{Y,f} \sim \epsilon(1-\epsilon/4)/a_i$, corresponding to the **Yes** event "f's tentative colour choice was $t(e)$". This leads to a bound for the total variance of this segment of

$$d_i\epsilon(1-\epsilon/4)(1+o(1)) \leq a_i(1-\epsilon/4)(1+o(1))$$

Altogether, a bound for A is

$$V(\mathcal{S}_A) \leq 5a_i(1-\epsilon/4)(1+o(1)) \leq 5a_i$$

for large enough n.

We now give a strategy for C, the one for B being the same as that for A. C counts the number of successful monochromatic pairs which use colours from $A_i(e)$ (a monochromatic pair is succesful if both edges succeed at this stage). First, we make $2d_i(1+o(1))$ queries for the u-edges and the v-edges. Each of these has maximum effect of 2. Using the **Yes** event "e was chosen for tentative colouring at this stage" we bound the variance of each query by $4\epsilon(1-\epsilon/4)$, which, when multiplied by the number of queries, gives an upper bound for the initial part of the query line of

$$8\epsilon(1-\epsilon/4)d_i(1+o(1)) \leq 8a_i(1-\epsilon/4)(1+o(1)).$$

At this point, there are at most $|A_i(e)| \sim a_i$ "half-succesful" monochromatic pairs of interest—*i.e.* those monochromatic pairs without colour conflicts so far. To determine if a monochromatic pair is succesful, we need to query the edges incident on the other endpoints. This total number of additional queries is then $a_i^2(1+o(1))$, because if edge f has chosen tentative colour $t(f)$, only the incident edges having $t(f)$ in their palettes can affect the value of C, and there are at most $(1+o(1))d_{i,t(f)} \sim a_i$ of these. To bound the variance c_h of each of these queried edges h we proceed as in Lemma 2. The edges are divided in three groups. Type A edges are those incident on two different half-succesful monochromatic pairs, *i.e.* h is incident on two edges f and g belonging to two different monochromatic pairs. Notice that these two pairs must have different tentative colours. Type B edges are those incident on two edges of the same pair and type C edges are those incident on just one edge of a pair.

For all types, the maximum effect of changing h's colour is 1. For type A edges we define the **Yes** event to be "h's tentative choice is α or β" where α and β are the tentative colours of the pairs incident on h. The maximum effect is bounded by $p_{Y,h}c_h^2 \sim 4\epsilon(1-\epsilon/4)/a_i$. For type B and C edges the **Yes** event is "h's choice is α", α being the tentative choice of the pair incident on h. The variance is bounded by $p_{Y,h}c_h^2 \sim \epsilon(1-\epsilon/4)/a_i$. Chosing the former as a worst case estimate on the variance and multiplying for the number of queries, we obtain the upper bound

$$4a_i\epsilon(1-\epsilon/4)(1+o(1)).$$

Altogether, our strategy to determine C has total variance of

$$V(\mathcal{S}_C) \leq a_i(1-\epsilon/4)(8+2\epsilon)(1+o(1)) \leq 9a_i$$

for large enough n and $\epsilon \leq 1/4$. □

5.3 Proof of $d_{i+1,\gamma}(u) \sim d_{i+1,\gamma}$

The expectation computation for $d_{i+1,\gamma}(u)$ could be performed similarly to Lemmas 1 and 3, but we would have to compute the probability of additional atomic events. We use instead a symmetry argument based on the (now valid) $(i+1)$ version of equation (3).

Lemma 5 $\mathrm{Ex}[d_{i+1,\gamma}(u)] \sim d_{i+1,\gamma}$.

Proof Since the graph is initially Δ–regular and the algorithm treats colours symmetrically

$$\Pr[\gamma \in A_{i+1}(w)] = \frac{\#\text{ colours left}}{\#\text{ initial colours}} = \frac{|A_{i+1}(w)|}{a_0}.$$

Therefore,

$$\mathrm{Ex}[d_{i+1,\gamma}(u)] = \sum_{w \in N_{i+1}(u)} \Pr[\gamma \in A_{i+1}(w)] \sim \sum_{w \in N_{i+1}(u)} \frac{d_{i+1}}{a_0} \sim \frac{d_{i+1}^2}{a_0}$$

by the $(i+1)$ version of equation (3). $\square$

To finish things off, we need a strong concentration result for $d_{i+1,\gamma}(u)$.

Lemma 6 *For each fixed vertex u and colour γ $|d_{i+1,\gamma}(u)|$ is within $6\sqrt{3\epsilon d_{i,\gamma}\log n}$ of its expectation with probability at least $1-2n^{-3}$.*

Remark Note that an error probability less than $2n^{-3}$ suffices since we will appeal to this lemma once for each vertex u and colour γ during each stage $i \ll n$. The two lemmas imply that with high probability

$$d_{i+1,\gamma}(u) = \mathrm{Ex}[d_{i+1,\gamma}(u)] \pm 6\sqrt{3\epsilon d_{i,\gamma}\log n} = \left(1 + o(1) \pm \frac{6}{1-p_\epsilon}\sqrt{\frac{3\epsilon \log n}{d_{i+1,\gamma}}}\right) d_{i+1,\gamma}.$$

So, to ensure that $d_{i+1,\gamma}(u) \sim d_{i+1,\gamma}$, we require that $(\epsilon \log n)/d_{i+1,\gamma} = o(1)$.

Proof At any given stage, a vertex u can lose a γ-neighbour v for only two reasons: (a) some v-edge colours itself γ, or (b) the edge $e = uv$ colours itself. (Notice that these events are neither disjoint nor independent.) We now describe a strategy to determine the value of $d_{i+1,\gamma}(u)$ of total variance at most $9\epsilon d_{i,\gamma}$. Propositon 1 with $\varphi = 3\log n$ then finishes the proof.

First we query all edges around u which have γ in their palettes. There are $d_{i,\gamma}(u) \sim d_{i,\gamma}$ of these, by the IH. The maximum effect of changing the tentative colour of one of these u-edges e is upper bounded by $c_e = 2$. Using the Yes event "e was selected for tentative colouring at this stage" gives a bound for this initial segment of the query line of

$$d_{i,\gamma} p_{Y,e} c_e^2 (1+o(1)) = 4\epsilon(1-\epsilon/4) d_{i,\gamma}(1+o(1)).$$

Call the u-edges without conflict around u *half-successful.*

In the second segment of the query line we want to determine *(i)* whether, for each half-succesful edge $e = uv$, there are conflicts at the other endpoint v and, *(ii)* whether any of the edges f incident on a u-neighbour v chooses tentative colour γ. Given that at this point the tentative choice $t(e)$ of each u-edge e is known, we need only query $d_{i,t(e)}(v) \sim d_{i,\gamma}$ edges to determine *(i)* and $d_{i,\gamma}(v) \sim d_{i,\gamma}$ edges to determine *(ii)*. The maximum effect of each of these queries is 1. The total number of queries of this second segment is therefore $2d_{i,\gamma}^2(1+o(1))$ because there are $d_{i,\gamma}(1+o(1))$ u-neigbours. By defining the Yes event as "f's tentative colour is γ or $t(e)$" we have a bound $v_f = 2\epsilon(1-p_\epsilon)/a_i(1+o(1))$ on the variance of each queried edge f. The resulting upper bound on the variance of this part of the query line is therefore

$$4d_{i,\gamma}\epsilon(1-\epsilon/4)(1+o(1)).$$

The third and final segment of the query line is to determine which edges incident on u-neighbours succesfully colour themselves with γ. The simple but crucial observation is that at most one v-edge f can colour itself γ. At this point of the query line we know which of these edges is half-succesful, *i.e.* which f's have no γ-conflict around the u-neighbour and we just need to query the other endpoint (not always, since f could connect two u-neighbours). This requires at most $d_{i,\gamma}$ additional queries per half-succesful edge. These queries have maximum effect bounded by 1 and variance at most $\epsilon(1-\epsilon/4)/a_i(1+o(1))$. The resulting variance for this third segment is therefore at most $d_{i,\gamma}\epsilon(1-\epsilon/4)(1+o(1))$. □

Remark Notice that in the proofs of Lemmas 4 and 6, the values of $|A_{i+1}(e)|$ and $d_{i+1,\gamma}(u)$ depend, respectively, on d_i^2 and d_i^3 edges. At the outset, the tentative choices of each of these edges could potentially affect $d_{i+1,\gamma}(u)$ by 1 or more. For this reason, the Method of Bounded Differences would give upper bounds on the variances no better than d_i^2 and d_i^3, which is orders of magnitude bigger than what we computed (and what we needed). Alternatively, there is an "expected" form of the MOBD [18, Cor. 6.10] which we might have used, but the computations and considerations would be more involved.

5.4 Error Analysis

We now briefly comment on the error propagation and its consequences for the analysis.

We proved that the error introduced at each stage is $o(1)$, hence if we iterate the nibble part of the algorithm only a constant number of times the final error will be still $o(1)$. This proves the first of our claims concerning the performance of the algorithm.

If the number of iterations is a function of n, however, the compounded error can become significant. Here we outline an error analysis that should enable the reader to verify our claims without too much trouble. There are two different kinds of error introduced in the analysis. One is the error due to random fluctuations. This error is computed in Lemmas 2, 4 and 6. Of these, the largest is the error introduced by Lemma 4. We use this as an upper bound for the error introduced by the i-th application of these lemmas and denote it by r_i.

The second type of error is due to the algebraic manipulations in the proofs of Facts 1 and 2 and the other lemmas which compute the expected values. The error is due to approximations like $1/(1-o(1)) = (1+o(1)), e^{a(1+o(1))} = e^a(1+o(1))$, etc. If we denote the total error accumulated at stage i by $(1 \pm e_i)$, we can check that the above manipulations increase the error from $(1 \pm e_i)$ to $(1 \pm Ce_i) = (1 \pm \Theta(e_i))$. In fact, careful inspection shows that the constant C is of the form $1 + K\epsilon$, where K is a (rather small) constant and ϵ is the nibble size. This enables us to use the nibble to keep the error under control.

To summarize, in the i-th phase, the error goes from $(1 \pm e_i)$ to $(1 \pm Ce_i \pm r_i)$, where $C = 1 + K\epsilon$, $r_i = c\sqrt{\log / a_i}$, and c is some other constant. In terms of the e_i's, $e_0 = 0$ and we have the recurrence

$$e_{i+1} = C\left(e_i + \sqrt{(\log n)/a_i}\right) = C\left(e_i + (1-p_\epsilon)^{-i}\sqrt{(\log n)/\Delta}\right).$$

The constant c disappeared from the recursion because it can be absorbed by C.

Setting $A = \sqrt{(\log n)/\Delta}$ and $B = 1/(1-p_\epsilon)$ and solving this recurrence gives

$$e_t = A[C^t + C^{t-1}B + \ldots + CB^{t-1}].$$

Notice that since $B = 1/(1-p_\epsilon)$, B is also of the form $1 + K'\epsilon$ for some constant K'. Setting $L = \max\{K, K'\}$, we see that

$$e_t \leq t(1+L\epsilon)^t\sqrt{(\log n)/\Delta}.$$

Since we need to maintain $e_{t_\epsilon} \ll 1$, this leads to the condition

$$\epsilon \gg \left(\frac{\log n}{\Delta}\right)^{1/4L}.$$

In particular, to support our second claim, ϵ can be set to $1/\log^s n$ provided that $\Delta \gg \log^{4Ls+1} n$.

6 Analysis: The Irregular Case

If the value of the maximum degree Δ could be inexpensively distributed to all of the edges, the regular case analysis would also be valid in the general case. However in a distributed architecture this might be too costly, so it is important that the algorithm rely on local information alone. This motivates the initial palette size of $\max\{\deg(u), \deg(v)\}$ for edge $e = uv$.

In fact what happens in the case when neighbouring edges receive different sized initial palettes is that the probability of conflict is decreased and so the edges succeed in colouring themselves even more rapidly than in the regular case. We will argue that the graph will be completely coloured at least as fast as a regular graph where all edges have the same initial palette size as the edge in our irregular graph with the smallest initial palette size. We'll do this by fixing an edge e and a round i and showing that the probability that e succeeds in colouring itself in this round is at least as high as if the graph were locally as in the regular case.

We modify e's neighbourhood in several ways, all of which decrease the probability of e's success. First of all, for every edge f incident with e, we ignore every colour from f's palette which was not in e's initial palette. This increases the probability of conflict between e and f by forcing f to choose a tentative colour which e at least has a chance of choosing and therefore decreases the probability of e's success. Next we add phantom edges to the vertex of e with lower degree. Say $e = uv$ and $\deg_i(u) \leq \deg_i(v)$, so we add phantom edges to u. To create the phantom edges' palettes and fill out the real edges' short palettes, we randomly add colours from e's initial palette until $\deg_{i,\gamma}(u) = \deg_{i,\gamma}(v) = d_{i,\gamma}$ for all colours γ. This only decreases e's probability of success, since it creates more opportunities for conflict.

But now the situation is locally just as if the graph were $\max\{\deg(u), \deg(v)\}$-regular and so the probability of e's success is as in the regular case analysis. And thus, in the original irregular graph, the probability of e's success is at least as high.

Acknowledgments

We would like to thank Noga Alon, Jiří Matoušek and Joel Spencer for their suggestion that the Rödl nibble should work, and to Noga Alon for several other comments and suggestions. We also thank Sem Borst, Marco Combe, and Kurt Melhorn for useful discussions. The first author is grateful to the CWI and the Altec project for making possible an extended visit to CWI. The third author acknowledges the generous hospitality of MPI and BRICS.

References

[1] N. Alon, Private Communications.

[2] N. Alon, J.H. Kim, and J. Spencer, Nearly perfect matchings in regular simple hypergraphs, preprint, 1995.

[3] N. Alon, J. Spencer, and P. Erdős, The Probabilistic Method, Wiley–Interscience Series, John Wiley & Sons, Inc., New York, 1992.

[4] B. Berger and J. Rompel, Simulating ($\log^c n$)-wise Independence in NC, Journal of the ACM, vol. 38 (4), 1991, 1026–1046

[5] B. Bollobás, Graph Theory, Springer Verlag, New York, 1979.

[6] S. Chaudhuri and D. Dubhashi, Probabilistic Recurrence Relations (Revisited), in R. Baeza–Yates, E. Goles and P. Poblete (eds): LATIN'95: Theoretical Informatics, Second Latin American Symposium, Valparaiso, Chile, LNCS 911, 207–219, 1995.

[7] D. Durand, R. Jain and D. Tseytlin, Distributed Scheduling Algorithms to improve the Performance of Parallel Data Transfers, DIMACS Tech Report 94–38, 1994.

[8] P. Frankl and V. Rödl, Near perfect coverings in graphs and hypergraphs, European Journal of Combinatorics, vol. 6, 1985, 317–326.

[9] Z. Galil and D. Leven, NP-completeness of finding the chromatic index of regular graphs, Journal of Algorithms, vol. 4, 1983, 35–44.

[10] D.A. Grable, A large deviation inequality for functions of independent, multi-way choices, preprint, 1996.

[11] I. Holyer, The NP-completeness of edge coloring, SIAM Journal of Computing, 1981, vol. 10, 718–720.

[12] 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, vol. 16 (4), 1992, 352–362.

[13] J. Kahn, Colouring nearly-disjoint hypergraphs with $n + o(n)$ colors, J. Comb. Theory, Series A, 1992, vol. 59, 31-39.

[14] H. J. Karloff and D. B. Shmoys, Efficient Parallel Algorithms for Edge Coloring Problems, Journal of Algorithms, 1987, vol. 8, 39–52.

[15] R.M. Karp, Probabilistic Recurrence Relations, 23rd STOC, 1991, 190–197.

[16] J.H. Kim, The Ramsey number R(3,t) has order of magnitude $t^2/\log t$, Ranodm Structures and Algorithms 7 (1995), 173–207.

[17] N. A. Lynch, Upper bounds for static resource allocation in a distributed system, Journal of Computer and System Sciences, 1981, vol. 23, 254–278.

[18] C. McDiarmid, On the method of bounded differences, In *Surveys in Combinatorics*, J.Siemons ed., London Math. Society Lecture Note Series 141 (1989), 148–188.

[19] R. Motwani, J. Naor, and M. Naor, The Probabilistic Method Yields Deterministic Parallel Algorithms, Proceedings of 30th FOCS, 1989, 8–13.

[20] A. Panconesi and A. Srinivasan, Randomized Distributed Edge Coloring via an Extension of the Chernoff-Hoeffding bounds, to appear in SIAM Journal of Computing. Preliminary version in Proceedings of PODC, 1992, 251–262.

[21] A. Panconesi and A. Srinivasan, Improved Distributed Algorithms for Coloring and Network Decomposition Problems, Proceedings of 24th STOC, 1992, 581–592.

[22] N. Pippinger and J. Spencer, Asymptotic behaviour of the chromatic index for hypergraphs, J. Combinatorial Theory, Series A, 1989, vol. 51, 24–42.

[23] V. Rödl, On a packing and covering problem, European Journal of Combinatorics, vol. 5, 1985, 69–78.

[24] J. Spencer, Asymptotically Good Coverings, Pacific Journal of Mathematics, 1985, vol. 118 (2), 575–586.

Recent Publications in the BRICS Report Series

RS-96-11 **Devdatt Dubhashi, David A. Grable, and Alessandro Panconesi.** ***Near-Optimal, Distributed Edge Colouring via the Nibble Method*****. May 1996. 17 pp. Invited to be published in in a special issue of *Theoretical Computer Science* devoted to the proceedings of ESA '95.**

RS-96-10 **Torben Braüner and Valeria de Paiva.** ***Cut-Elimination for Full Intuitionistic Linear Logic*****. April 1996. 27 pp. Also available as Technical Report 395, Computer Laboratory, University of Cambridge.**

RS-96-9 **Thore Husfeldt, Theis Rauhe, and Søren Skyum.** ***Lower Bounds for Dynamic Transitive Closure, Planar Point Location, and Parentheses Matching*****. April 1996. 11 pp. To appear in *Algorithm Theory: 5th Scandinavian Workshop*, SWAT '96 Proceedings, LNCS, 1996.**

RS-96-8 **Martin Hansen, Hans Hüttel, and Josva Kleist.** ***Bisimulations for Asynchronous Mobile Processes*****. April 1996. 18 pp. Appears in *Tbilisi Symposium on Language, Logic, and Computation*, 1995.**

RS-96-7 **Ivan Damgård and Ronald Cramer.** ***Linear Zero-Knowledegde - A Note on Efficient Zero-Knowledge Proofs and Arguments*****. April 1996. 17 pp.**

RS-96-6 **Mayer Goldberg.** ***An Adequate Left-Associated Binary Numeral System in the λ-Calculus (Revised Version)*****. March 1996. 19 pp. Accepted for *Information Processing Letters*. This report is a revision of the BRICS Report RS-95-38.**

RS-96-5 **Mayer Goldberg.** ***Gödelisation in the λ-Calculus (Extended Version)*****. March 1996. 10 pp.**

RS-96-4 **Jørgen H. Andersen, Ed Harcourt, and K. V. S. Prasad.** ***A Machine Verified Distributed Sorting Algorithm*****. February 1996. 21 pp. Abstract appeared in *7th Nordic Workshop on Programming Theory*, NWPT '7 Proceedings, 1995.**

RS-96-3 **Jaap van Oosten.** ***The Modified Realizability Topos*****. February 1996. 17 pp.**