On the Time and the Bit Complexity of Distributed Randomised Anonymous Ring Colouring

Y. Métivier, J. M. Robson, N. Saheb-Djahromi and A. Zemmari

Université de Bordeaux, LaBRI, UMR CNRS 5800
351 cours de la Libération, 33405 Talence, France
{*metivier, robson, saheb, zemmari*}*@labri.fr*

Abstract

We present and analyse a very simple randomised distributed vertex colouring algorithm for ring graphs. Its time complexity is $\log_2 n + o(\log n)$ on average and $2\log_2 n + o(\log n)$ with probability $1 - o(n^{-1})$. Each message containing 1 bit we deduce the same values for its bit complexity. Then we compose this algorithm with another and we obtain a 3-colouring algorithm for ring graphs. Thanks to an overlapping, we obtain once more the same values for the time complexities on average and with probability $1 - o(n^{-1})$. The same results hold for the bit complexity. These results are obtained using the Mellin transform.

We establish lower bounds (on average and with probability $1 - o(n^{-1})$) for the distributed randomised anonymous ring colouring problem. We prove that our algorithms match these lower bounds modulo a negligible additive function (negligible with respect to $\log_2 n$).

We assume that the ring is anonymous: unique identities are not available to distinguish the processes; we only assume that each vertex distinguishes between its neighbours. Furthermore we do not assume that the size (or an upper bound on the size) of the ring is known.

Keywords: Bit Complexity, Colouring, Mellin Transform, Probabilistic Analysis, Randomised Distributed Graph Algorithm, Ring, Time Complexity.

1. Introduction

1.1. The Problem

Let $G = (V, E)$ be a simple undirected graph. A vertex *colouring* of G assigns colours to each vertex in such a way that neighbours have different colours. If we need at most 3 colours then we say a 3-colouring.

In this paper we discuss how efficient (time, bits and number of colours) vertex colouring may be accomplished by exchange of bits between neighbouring vertices in a ring graph. Even if ring graphs are simple, they are used as a case study in many problems, as explained by Attiya and Welch in [AW04], page 31:

"rings are a convenient structure for message-passing systems and correspond to physical communication systems, for example, token rings."

The distributed complexity of vertex colouring is of fundamental interest for the study and analysis of distributed computing. Usually, the topology of a distributed system is modelled by a graph and paradigms of distributed systems are encoded by classical problems in graph theory; among these classical problems one may cite the problems of vertex colouring, computing a maximal independent set, finding a vertex cover or finding a maximal matching. Each solution to one of these problems is a building block for many distributed algorithms: symmetry breaking, topology control, routing, resource allocation.

1.2. The Model

The Network. We consider the standard message passing model for distributed computing. The communication model consists of a point-to-point communication network described by a simple undirected ring graph $G = (V, E)$ where the vertices V represent network processors and the edges represent bidirectional communication channels. Processes communicate by message passing: a process sends a message to another by depositing the message in the corresponding channel. We assume the system synchronous and synchronous wake up of processors: processors have access to a global clock and all processors start the algorithm at the same time.

In this model, a distributed algorithm is given by a local algorithm that all processes should execute; thus processes having the same degree have the same algorithm. A local algorithm consists of a sequence of computation steps interspersed with instructions to send and to receive messages.

A probabilistic algorithm is an algorithm which makes some random choices based on some given probability distributions.

A distributed probabilistic algorithm is a collection of local probabilistic algorithms; furthermore the network is anonymous; thus if two processes have the same degree their local probabilistic algorithms are identical and have the same probability distribution.

A Las Vegas algorithm is a probabilistic algorithm such that when it terminates its result is correct, nevertheless its runtime is unknown (not deterministically bounded): it terminates with probability 1.

Time Complexity. A round (cycle) of each processor is composed of the following three steps: 1. Send messages to (some of) the neighbours, 2. Receive messages from (some of) the neighbours, 3. Perform some local computation. As usual (see for example Peleg [Pel00]) the time complexity is the maximum possible number of rounds needed until every node has completed its computation.

Bit Complexity. A bit round is a round such that each processor can send/receive at most 1 bit to/from each neighbour. As in [KOSS06], the bit complexity of an algorithm $\mathcal{A}$ is the number of bit rounds to complete algorithm $\mathcal{A}$. One round of the algorithm contains 1 or more bit round.

It is considered as a finer measure of communication complexity and it has been studied for breaking and achieving symmetry or for colouring in [BMW94, KOSS06, DMR08]. Dinitz et al. explain in [DMR08] that it may be viewed as a natural extension of communication complexity (introduced by Yao [Yao79]) to the analysis of tasks in a distributed setting. An introduction to this area can be found in Kushilevitz and Nisan [KN99].

Remark 1. In this paper we assume that at each round each vertex which communicates with a neighbour through a port sends either the bit 1 or the bit 0. In this way processors do not need to access to a global clock: our algorithm works under the hypothesis of a synchronisation by rounds. In our work, if a vertex v sends neither the bit 0 nor the bit 1 to a neighbour w in a round, it means that v has no information to transmit to w, v ignores w or w does not participate to this round.

In some models, if a vertex does not send a bit to a neighbour then it is considered as information. If we allow the case where no bit is sent, so that there must be synchronisation by a global clock, we obtain communications with an alphabet of three letters, which can be simulated easily by communications with an alphabet of two letters. This simulation does not change the bit complexity modulo a multiplicative constant.

Remark 2. Dinitz et al. [DMR08] define the bit complexity as the number of transmitted bits required to solve a task. The definition we adopt in this paper gives a bit complexity which is an upper bound of the bit complexity induced by the definition given in [DMR08].

Network and Process Knowledge. The network is an anonymous ring: unique identities are not available to distinguish the processes. We do not assume any knowledge of the size of the graph or of an upper bound on the size of the graph, any position or distance information. Each processor knows from which channel it receives a message. An important fact due to the initial symmetry is: *there is no deterministic distributed algorithm for arbitrary anonymous graphs for vertex colouring assuming all vertices wake up simultaneously,* see [Pel00].

1.3. Our Contribution

Métivier et al. [MRSDZ10] present and analyse a randomised distributed vertex colouring algorithm, denoted Algorithm $\mathcal{B}$. This algorithm runs in $O(\log n)$ rounds on average and with high probability[1] (w.h.p. for short). Algorithm $\mathcal{B}$, called the 3-colouring algorithm in this paper, is composed of a colouring algorithm and a decreasing colours algorithm.

In this paper, we do a more precise study for the time (bit) complexity of the colouring algorithm and of the 3-colouring algorithm in the case of ring graphs. In particular for the multiplicative constant hidden in $O(\log n)$.

[1] With high probability means with probability $1 - o(n^{-1})$.

In the colouring algorithm, each vertex computes a proper colour (a colour different from its neighbours). As soon as a vertex has a proper colour it starts the decreasing colours algorithm: an algorithm which reduces the number of colours used in all the graph to achieve 3 colours.

Using the Mellin transform, we prove that the colouring algorithm computes a proper colouring for any ring graph of size n in $\log_2 n + o(\log n)$ rounds on average and in at most $2\log_2 n + o(\log n)$ rounds w.h.p. (which matches the lower bound by Kothapalli et al). We prove also that the decreasing colours algorithm acts in at most $2\log_2 n$ rounds w.h.p. to achieve a $3-$colouring for such a graph.

Then we show how the two algorithms can be overlapped to obtain the optimal solution of the problem: *given a ring graph G of size n, the 3-colouring algorithm computes a $3-$colouring in $\log_2 n + o(\log n)$ rounds on average and $2\log_2 n + o(\log n)$ w.h.p.*

The algorithms use messages of size of one bit. That is, the same values hold for their bit complexity.

In [KOSS06], Kothapalli et al. prove that $\Omega(\log n)$ rounds are necessary to colour a ring of size n. In Section 6, we obtain exact values for the constant hidden in the $\Omega()$. Note that their bound is on a proper colouring of the ring with no restriction on the number of colours. Finally, we deduce that our algorithms match our new lower bounds modulo a negligible additive function (negligible with respect to $\log_2 n$).

1.4. Related Work: Comparisons and Comments

Vertex Colouring. Vertex colouring is a fundamental problem in distributed systems. It is mainly studied under two assumptions:

1. vertices have unique identifiers, and more generally, they have an initial colouring,
2. every vertex has the same initial state and initially only knows its own edges.

Vertex colouring is a classical technique to break symmetry. If vertices have unique identifiers (or if there is an initial colouring) then the initial local symmetry is naturally broken; otherwise, as usual, it is broken by using randomisation.

Vertices Have an Initial Colouring. In this case, as we said before, the local symmetry is broken and, generally, vertex colouring algorithms try to decrease the number of colours (for example, to $\Delta + 1$ or to $O(\Delta)$ where Δ is the maximum vertex degree in the graph). Classical examples are given in [Pel00] (Chapter 7). The model assumes that each node has a unique $O(\log n)$ bits identifier. More recently, Kuhn and Wattenhofer [KW06] have presented efficient time complexity algorithms to obtain $O(\Delta)$ colours in the case where every vertex can only send its own current colour to all its neighbours. Cole and Vishkin [CV86] show that there is a distributed algorithm which colours a cycle on n vertices with 3 colours and runs in $O(\log^* n)$. Lower bounds for colouring particular families of graphs are given in [Lin92]. That paper presents also an algorithm which

colours a graph of size n with $O(\Delta^2)$ colours and runs in $O(\log^* n)$.

Vertices Have the Same Initial State and no Knowledge. In this case we have no choice: we use randomised algorithms. In [Joh99], Johansson analyses a simple randomised distributed vertex colouring algorithm. Each vertex u keeps a palette of colours initialised to $\{0, ..., d\}$ if the degree of u is d. The algorithm then proceeds in rounds. In a round each uncoloured vertex u randomly chooses a colour c from its palette. It sends c to its neighbours and receives a colour from each neighbour. If the colour c chosen by u is different from colours chosen by its neighbours then c becomes the final colour of u, u informs its neighbours and becomes passive. At the beginning of the next round each active vertex removes from its palette final colours of its neighbours.

Johansson proves that this algorithm runs in $O(\log n)$ rounds with high probability on graphs of size n.

1.5. Summary

First, we present in Section 2 the Mellin transform and technical results which we use for the analysis of the colouring algorithm and the 3-colouring algorithm. Section 3 describes and analyses the colouring algorithm. Section 4 gives and analyses the 3-colouring algorithm. Section 5 studies more precisely the complexity of the 3-colouring algorithm thanks to the overlapping of the colouring algorithm and of the decreasing colours algorithm. Section 6 gives lower bounds for the randomised ring colouring problem matched by our algorithm. Section 7 is the conclusion.

2. Mellin Transform

This section presents the definition of the Mellin transform and some of its properties. The reader is refered to [FRS85] for more details. Then we give an application used for the analysis of the colouring algorithm and of the 3-colouring algorithm.

2.1. Definition and Properties

The *Mellin transform* of a real valued function $F(x)$ defined over $[0, +\infty]$ is the complex function $F^*(s)$ of the complex variable s given by:

$$F^*(s) = \int_0^\infty F(x)x^{s-1}dx. \tag{1}$$

In general, the integral exists only for complex values of $s = Re(s) + i\, Im(s)$ such that $\alpha < Re(s) < \beta$, where α and β depend on the function $F(x)$. This introduces what is called the *fundamental strip* of the Mellin transform and is denoted $< \alpha, \beta >$. In some cases, this strip may extend to a half-plane ($\alpha = -\infty$ or $\beta = +\infty$) or to the whole complex plane ($\alpha = -\infty$ and $\beta = +\infty$).

One can obtain F from F^* using the following theorem:

Theorem 1. *(Inversion Theorem) For any real c inside the fundamental strip, one has:*

$$F(x) = \frac{1}{2i\pi} \int_{c-i\infty}^{c+i\infty} F^*(s) x^{-s} ds. \tag{2}$$

2.2. Application

In this section, we present an application of the Mellin transform to obtain the singular expansion of some functions. The main lemma proved here will be used in the following sections.

Lemma 2. *Let F and G be the functions defined for any real number x by*

$$F(x) = \sum_{k\geq 0} \left(1 - e^{-x/2^k}\right), \text{ and } G(x) = \sum_{k\geq 0} (2k+1) \left(1 - e^{-x/2^k}\right).$$

Then, as x tends to ∞, we have the following singular expansion of F and G:

$$F(x) = \log_2 x + \frac{1}{2} + \frac{\gamma}{\log 2} + Q(\log_2 x) + O\left(x^{-2}\right), \tag{3}$$

where $Q(u) = -\frac{1}{\log 2} \sum_{k\in\mathbb{Z}\setminus\{0\}} \Gamma\left(\frac{2ik\pi}{\log 2}\right) e^{-2ik\pi u}$ and

$$\begin{aligned} G(x) &= (\log_2 x)^2 + \left(1 + \frac{2\gamma}{\log 2}\right) \log_2 x \\ &\quad + \frac{1}{3} + \frac{\gamma}{\log 2} + \frac{\pi^2}{6(\log 2)^2} + \left(\frac{\gamma}{\log 2}\right)^2 + \Delta(\log_2 x) + O\left(\frac{1}{x^2}\right), \end{aligned} \tag{4}$$

where $\Delta(u) = -\frac{2}{(\log 2)^2} \sum_{k\in\mathbb{Z}\setminus\{0\}} \Gamma(\chi_k) e^{-2ik\pi u}$ and γ denotes the Euler-Mascheroni constant.

Proof. We first give the complete proof of (3), then, we give a sketch of the proof of (4).

The Mellin transform of $F(x)$ is:

$$\begin{aligned} F^*(s) &= \int_0^\infty x^{s-1} F(x) dx \\ &= \sum_{k\geq 0} \int_0^\infty x^{s-1} \left(1 - e^{-x/2^k}\right) dx \\ &= -\Gamma(s) \sum_{k\geq 0} 2^{ks} \\ &= -\frac{\Gamma(s)}{1-2^s}, \end{aligned}$$

with fundamental strip $< -1, 0 >$, where:

$$\Gamma(s) = \frac{e^{-\gamma s}}{s} \prod_{l=1}^{\infty} \frac{1}{1 + \frac{s}{l}} e^{s/l}.$$

Function $F^*(s)$ has a double pole at $s = 0$, and imaginary poles at $s = \chi_k = \frac{2ik\pi}{\log 2}$ for any $k \in \mathbb{Z} \setminus \{0\}$. It admits a meromorphic continuation to $< -1, +2 >$ and is analytic on $Re(s) = 2$. Thus, the singular expansion of $F^*(s)$ in $< -\frac{1}{2}, +2 >$ is:

$$F^*(s) = \left[\frac{1}{\log 2}\frac{1}{s^2} - \frac{\gamma + \frac{1}{2}\log 2}{s \log 2}\right] + \left[\frac{1}{\log 2}\sum_{k\in\mathbb{Z}\setminus\{0\}} \frac{\Gamma(\chi_k)}{s - \chi_k}\right]. \tag{5}$$

We can also verify that, for any s such that $-1/2 \leq Re(s) \leq 2$:

$$F^*(s) = O\left(\mid s \mid^{-1}\right) \text{ as } \mid s \mid \to \infty. \tag{6}$$

Now, consider the integral:

$$I(T) = \frac{1}{2i\pi}\int_{\mathcal{C}(T)} F^*(s) x^{-s} ds,$$

where $\mathcal{C}(T)$ denotes the rectangular contour defined by the segments

$$\begin{aligned}[-1/2 - iT, -1/2 + iT],\ [-1/2 + iT, +2 + iT],\\ [+2 + iT, +2 - iT],\ [+2 - iT, -1/2 - iT].\end{aligned} \tag{7}$$

By Cauchy's theorem, $I(T)$ is equal to the sum of residues.
On the other hand, for any $k > 0$,:

$$Res\left(\frac{x^{-s}}{(s-\zeta)^k}, \zeta\right) = \frac{(-1)^{k-1}}{(k-1)!} x^{-\zeta} (\log x)^{k-1}.$$

Thus:

$$Res\left(\frac{x^{-s}}{s^2}, 0\right) = -\log x \text{ and } Res\left(\frac{x^{-s}}{s}, 0\right) = 1, \tag{8}$$

and, for any $k \in \mathbb{Z} \setminus \{0\}$:

$$Res\left(\frac{x^{-s}}{s - \chi_k}, \chi_k\right) = e^{-\chi_k \log x}. \tag{9}$$

Thus:

$$I(T) = -\frac{\log x}{\log 2} - \frac{\gamma + 1/2 \log 2}{\log 2} + \frac{1}{\log 2}\sum_{k\in\mathbb{Z}\setminus\{0\}} \Gamma(\chi_k) e^{-\chi_k \log x}. \tag{10}$$

Now, we can write $I(T)$:

$$I(T) = I_1(T) + I_2(T) + I_3(T) + I_4(T), \tag{11}$$

with:

$$I_1(T) = \frac{1}{2i\pi}\int_{-1/2-iT}^{-1/2+iT} F^*(s)x^{-s}ds, \quad I_2(T) = \frac{1}{2i\pi}\int_{-1/2+iT}^{2+iT} F^*(s)x^{-s}ds,$$
$$I_3(T) = \frac{1}{2i\pi}\int_{2+iT}^{2-iT} F^*(s)x^{-s}ds, \qquad I_4(T) = \frac{1}{2i\pi}\int_{2-iT}^{-1/2iT} F^*(s)x^{-s}ds.$$

Now let T tend to $+\infty$. By (6), the integrals $I_2(T)$ and $I_4(T)$ are $O\left(T^{-1}\right)$ and thus tend to 0 as T tends to ∞. The integral $I_1(T)$ along the vertical line $Re(s) = c = -1/2$ that lies inside the fundamental strip tends to the inverse Mellin integral which converges given the growth of $F^*(s)$ and equals $F(x)$ by the (Inversion Theorem) Theorem 1. Finally, by (6), and as $x \to \infty$, the integral along the vertical line $Re(s) = 2$ verifies:

$$\begin{aligned}\frac{1}{2\pi}\int_{2-i\infty}^{2+i\infty} \mid F^*(s) \mid\mid x^{-s} \mid\mid ds \mid &= O(1)\int_0^\infty \frac{x^{-t}}{t+1}dt \\ &= O\left(x^{-2}\right)\int_0^\infty \frac{dt}{(t+1)^2} \\ &= O\left(x^{-2}\right).\end{aligned}$$

Thus, in the limit, $I(\infty)$ equals $F(x)$ plus a term that is $O\left(x^{-2}\right)$ plus the sum of the residues. This yields:

$$F(x) = \log_2 x + \frac{1}{2} + \frac{\gamma}{\log 2} + Q\left(\log_2 x\right) + O\left(x^{-2}\right), \tag{12}$$

where:

$$Q(u) = -\frac{1}{\log 2}\sum_{k\in\mathbb{Z}\setminus\{0\}} \Gamma\left(\frac{2ik\pi}{\log 2}\right) e^{-2ik\pi u}.$$

This establishes the expansion given in (3).

To prove (4), we first give the Mellin transform of G:

$$\begin{aligned}G^*(s) &= \int_0^\infty x^{s-1}G(x)dx \\ &= \sum_{k\geq 0}(2k+1)\int_0^\infty x^{s-1}\left(1-e^{-x/2^k}\right)dx \\ &= -\Gamma(s)\sum_{k\geq 0}(2k+1)2^{ks} \\ &= -\Gamma(s)\frac{2^s+1}{(1-2^s)^2},\end{aligned}$$

with fundamental strip $\langle -1, 0\rangle$.

The function $G^*(s)$ admits a pole of order 3 on $s = 0$ and imaginary poles of order 2 on $s = \chi_k = \frac{2ik\pi}{\log 2}$ for any $k \in \mathbb{Z} \setminus \{0\}$. It admits a meromorphic continuation to $< -1, +2 >$ and is analytic on $Re(s) = 2$. That is, we have:

$$\begin{aligned}G^*(s) &= -\frac{2}{(\log 2)^2}\frac{1}{s^3} + \left(\frac{1}{\log 2} + \frac{2\gamma}{(\log 2)^2}\right)\frac{1}{s^2} \\ &\quad -\left(\frac{1}{3} + \frac{\gamma}{\log 2} + \frac{\pi^2}{6(\log 2)^2} + \frac{\gamma^2}{(\log 2)^2}\right)\frac{1}{s} \\ &\quad +\frac{2}{(\log 2)^2}\sum_{k\in\mathbb{Z}\setminus\{0\}}\frac{\Gamma(\chi_k)}{(s-\chi_k)^2}.\end{aligned} \tag{13}$$

To obtain the singular expansion of $G(x)$ from $G^*(s)$, we use the same reasoning as before. Indeed, for any s such that $-1/2 \leq Re(s) \leq +2$, we have $G^*(s) =\mid s \mid^{-1}$ as $s \to \infty$. Then, we consider the integral:

$$J(T) = \frac{1}{2i\pi} \int_{C(T)} x^{-s} G^*(s) ds,$$

where $C(T)$ is the contour defined in (7). We can also see that:

$$Res\left(\frac{x^{-s}}{s^3}, 0\right) = \frac{1}{2}\left(\log x\right)^2, \tag{14}$$

and, for any $k \in \mathbb{Z} \setminus \{0\}$:

$$Res\left(\frac{x^{-s}}{\left(s-\chi_k\right)^2}, \chi_k\right) = -e^{-\chi_k \log x} \log x. \tag{15}$$

Accordingly, by a similar decomposition of $J(T)$ as in (11), and using (8), (14), (15), we derive:

$$\begin{aligned} G(x) &= (\log_2 x)^2 + \left(1 + \frac{2\gamma}{\log 2}\right) \log_2 x \\ &\quad + \frac{1}{3} + \frac{\gamma}{\log 2} + \frac{\pi^2}{6(\log 2)^2} + \left(\frac{\gamma}{\log 2}\right)^2 + \Delta\left(\frac{\log x}{\log 2}\right) + O\left(\frac{1}{x^2}\right), \end{aligned} \tag{16}$$

where $\Delta(u) = -\frac{2}{(\log 2)^2} \sum_{k \in \mathbb{Z}\setminus\{0\}} \Gamma(\chi_k) e^{-2ik\pi u}$.

This ends the proof. □

Remark 3. A part of the proof of Lemma 2 can be obtained by a direct application of the theorem *Reverse mapping* from [FS96]. Indeed, using this theorem, one obtains (12) directly from (5), and (16) directly from (13).

3. The Colouring Algorithm

The colouring algorithm is given as Algorithm 1. At each round of the colouring algorithm, some vertices are permanently coloured, and stop executing the algorithm, some edges are removed from the graph (this means that they become passive, i.e., they do not participate in the sequel of the algorithm) and the other vertices continue the execution of the algorithm in the residual graph. Let $G = (V, E)$ be the initial graph, we denote by $(G_i)_{i \geq 0}$ the sequence of graphs induced by active vertices, where $G_0 = G$ and G_i is the residual graph obtained after the i^{th} round.

Formally, each vertex v maintains a list $active_v$ of active vertices, i.e., neighbours which are not yet coloured and with which the symmetry is not yet broken; initially $active_v$ is equal to $N(v)$ (the neighbours of v). We denote by $colour_v$ the colour of the vertex v which is the word formed by bits generated by the vertex v (we denote by $\oplus$ the concatenation operation on words); initially $colour_v$

is the empty word. The vertex v generates a random bit b_v; it concatenates b_v to $colour_v$, i.e., the new colour of v is $colour_v \oplus b_v$; it sends b_v to all its active neighbours, receives the bits sent by these vertices and then updates its list. This action is repeated until the symmetry is broken with its two neighbours and hence the vertex has obtained its final colour.

Algorithm 1 The Colouring Algorithm

```
1: var:
2:         colour_v : word Init empty-word;
3:         active_v: ⊆ N(v) Init N(v);
4:         b_v: ∈ {0,1};
5: while active_v ≠ ∅ do
6:    b_v ← flip(0,1) ;
7:    colour_v ← colour_v ⊕ b_v;
8:    for all u ∈ active_v do
9:       send b_v to u;
10:      receive b_u from u;
11:      if b_v ≠ b_u then
12:         active_v ← active_v \ {u}
13:      end if
14:   end for
15: end while
```

Remark 4. The colour of a node v is the concatenation of all the bits generated from the start of execution of the algorithm to the time where v has no active neighbours.

We have the following theorem:

Theorem 3. *Let $G = (V, E)$ be a ring of size $n \geq 3$. Let T_1 denote the number of rounds necessary to colour all vertices by the colouring algorithm.*

- *The expected value of T_1 is asymptotically equal to $\log_2 n - \frac{1}{2} + \frac{\gamma}{\log 2} + Q(\log_2 n) + O\left(n^{-2}\right)$, where $Q(u) = -\frac{1}{\log 2} \sum_{k \in \mathbb{Z}\setminus\{0\}} \Gamma\left(\frac{2ik\pi}{\log 2}\right) e^{-2ik\pi u}$ is a Fourier series with period 1 and with an amplitude which does not exceed 10^{-6},*
- *it is less than $2\log_2 n$ w.h.p.*

Proof. If n is odd, then we need at least tow rounds to achieve the graph colouring. Hence, $\mathbb{P}r(T_1 > 1) = 1$. Otherwise, if n is even, it is easy to see that $\mathbb{P}r(T_1 > 1) = 1 - \frac{1}{2^{n-1}}$.

More generally, using the Sieve principle [Ros00], for any $k \geq 2$:

$$\begin{aligned}\mathbb{P}r\left(T_1 > k\right) &= \sum_{i=1}^{n-1}(-1)^{i+1}\binom{n}{i}\left(\frac{1}{2^i}\right)^k + (-1)^{n+1}\left(\frac{1}{2^{n-1}}\right)^k \\ &= 1 - \left(1-\frac{1}{2^k}\right)^n + (-1)^{n+1}\left(\left(\frac{1}{2^{n-1}}\right)^k - \left(\frac{1}{2^n}\right)^k\right). \quad (17)\end{aligned}$$

Hence, with $k = 2\log_2 n$, we get:

$$\mathbb{P}r\left(T_1 > k\right) \sim 1 - e^{-1/n} \to 0 \text{ as } n \to \infty.$$

This proves the second claim of the theorem.

On the other hand, since $\mathbb{E}\left(T_1\right) = \sum_{k\geq 1}\mathbb{P}r\left(T_1 > k\right)$, we have:

$$\mathbb{E}\left(T_1\right) = \mathbb{P}r\left(T_1 > 1\right) + \sum_{k\geq 2}\left[1 - \left(1-\tfrac{1}{2^k}\right)^n + (-1)^{n+1}\left(\left(\tfrac{1}{2^{n-1}}\right)^k - \left(\tfrac{1}{2^n}\right)^k\right)\right].$$

Without loss of generality, we consider the case of odd n since even n simply increases the value by 1.

$$\begin{aligned}\mathbb{E}\left(T_1\right) &= 1 + \sum_{k\geq 2}\left[1 - \left(1-\tfrac{1}{2^k}\right)^n\right] + \tfrac{1}{2^n} - \tfrac{1}{2^{n-1}} + \tfrac{1}{1-\frac{1}{2^{n-1}}} - \tfrac{1}{1-\frac{1}{2}^n} \\ &= \sum_{k\geq 0}\left[1 - \left(1-\tfrac{1}{2^k}\right)^n\right] - 1 + \tfrac{1}{1-\frac{1}{2^{n-1}}} - \tfrac{1}{1-\frac{1}{2}^n}.\end{aligned}$$

On the other hand, we have the exponential approximation $(1-a)^n \sim e^{-an}$, then, with:

$$F(x) = \sum_{k\geq 0}\left(1 - e^{-x/2^k}\right),$$

one finds elementarily $\mathbb{E}\left(T_1\right) \sim F(n) - 1$. Then, the first claim is proved using Lemma 2. □

We can also derive the distribution of the r.v. T_1:

Lemma 4. *Let $G = (V, E)$ be a ring graph of size $n \geq 3$ and let T_1 be the r.v. defined above. Then:*

- $\mathbb{P}r\left(T_1 = 0\right) = 0$.
- *If n is odd, then:* $\mathbb{P}r\left(T_1 = 1\right) = 0$, *and* $\mathbb{P}r\left(T_1 = 2\right) = \frac{3^n-3}{4^n}$.
- *If n is even, then:* $\mathbb{P}r\left(T_1 = 1\right) = \frac{1}{2^{n-1}}$, *and* $\mathbb{P}r\left(T_1 = 2\right) = \frac{3^n+3}{4^n} - \frac{1}{2^{n-1}}$.

and, for any $k > 2$:

$$\begin{aligned}\mathbb{P}r\left(T_1 = k\right) &= \left(1-\frac{1}{2^k}\right)^n - \left(1-\frac{1}{2^{k-1}}\right)^n \\ &+ (-1)^{n+1}\left(\left(\frac{1}{2^{n-1}}\right)^{k-1} - \left(\frac{1}{2^n}\right)^{k-1} - \left(\frac{1}{2^{n-1}}\right)^k + \left(\frac{1}{2^n}\right)^k\right).\end{aligned}$$

PROOF. For any $k \geq 1$, we have $\mathbb{P}r\left(T_1 = k\right) = \mathbb{P}r\left(T_1 > k-1\right) - \mathbb{P}r\left(T_1 > k\right)$. Then it suffices to apply (17) to obtain the lemma. □

We have also the following lemma:

Proposition 1. *The ratio between T_1 and $\log_2 n$ tends to 1 in probability as $n \to \infty$.*

PROOF. We start by calculating the second moment of the r.v. T_1 :

$$\begin{aligned}\mathbb{E}\left(T_1^2\right) &= \sum_{k\geq 0} k^2 \mathbb{P}r\left(T_1 = k\right) \\ &= \sum_{k\geq 1} k^2 \left[\left(1-\tfrac{1}{2^k}\right)^n - \left(1-\tfrac{1}{2^{k-1}}\right)^n\right] \\ &= \sum_{k\geq 0}(2k+1)\left(1-\left(1-\tfrac{1}{2^k}\right)^n\right).\end{aligned}$$

Using the same exponential approximation as before, we have $\mathbb{E}\left(T_1^2\right) \sim G(n)$ where:

$$G(x) = \sum_{k\geq 0}(2k+1)\left(1-e^{-x/2^k}\right).$$

By Lemma 2, we finally obtain the value of the second moment:

$$\begin{aligned}\mathbb{E}\left(T_1^2\right) &= (\log_2 n)^2 + \left(1+\tfrac{2\gamma}{\log 2}\right)\log_2 n \\ &\quad +\tfrac{1}{3} + \tfrac{\gamma}{\log 2} + \tfrac{\pi^2}{6(\log 2)^2} + \left(\tfrac{\gamma}{\log 2}\right)^2 + \Delta\left(\tfrac{\log n}{\log 2}\right) + O\left(\tfrac{1}{n^2}\right).\end{aligned}$$

Since $\mathbb{V}ar\left(T_1\right) = \mathbb{E}\left(T_1^2\right) - \left(\mathbb{E}\left(T_1\right)\right)^2$, we obtain:

$$\begin{aligned}\mathbb{V}ar\left(T_1\right) &= \left(\tfrac{1}{\log 2} - 1\right)\log_2 n + \tfrac{1}{12} + \tfrac{\pi^2}{6(\log 2)^2} \\ &\quad -P\left(\log_2 n\right) + O\left(\tfrac{1}{n^2}\right),\end{aligned}$$

where $P(u) = Q(u)^2 + \left(2u + \tfrac{2\gamma}{\log 2} - \tfrac{2}{\log 2}\right)Q(u)$.

Now, define the r.v. $R_n = \frac{T_1}{\log_2 n}$, then:

$$\mathbb{E}\left(R_n\right) = 1 + \left(\frac{1}{2} + \frac{\gamma}{\log 2}\right)/\log_2 n + \frac{1}{\log_2 n}\left(Q\left(\log_2 n\right) - O\left(\frac{1}{n^2}\right)\right),$$

and

$$\begin{aligned}\mathbb{V}ar\left(R_n\right) &= \mathbb{V}ar\left(T_1\right)/(\log_2 n)^2 \\ &= \left(\tfrac{1}{\log 2} - 1\right)/\log_2 n + \left(\tfrac{1}{12} + \tfrac{\pi^2}{6(\log 2)^2}\right)/(\log_2 n)^2 \\ &\quad -\tfrac{1}{(\log n)^2}P\left(\log_2 n\right) + O\left(\tfrac{1}{n^2}\right).\end{aligned}$$

Using the Tchebychev inequality, we have:

$$\forall \varepsilon > 0, \mathbb{P}r\left(\mid R_n - 1 \mid \geq \varepsilon\right) \leq \frac{\mathbb{V}ar\left(R_n\right)}{\varepsilon^2} \to 0 \text{ as } n \to \infty,$$

which ends the proof. □

Remark 5. Vertices exchange messages containing 1 bit. Thus the results in Theorem 3, Lemma 4 and Proposition 1 are valid for the bit complexity of the algorithm.

4. The 3-Colouring Algorithm

The 3-colouring algorithm is composed of the colouring algorithm and a decreasing colours algorithm which we describe now.

4.1. The Decreasing Colours Algorithm

The algorithm is given as Algorithm 2. When a vertex u ends the colouring algorithm, it obtains a proper colour. Then u calculates a binary fraction $x(u)$ using the word $colour_u$ as follows: if $colour_u = b_1b_2b_3\cdots$, then $x(u) = 0.b_1b_2b_3\cdots$. For any neighbour v, if $x(u) < x(v)$, then the edge $\{u, v\}$ is oriented from u to v; otherwise, it is oriented from v to u. Let:

$IN_u = \{v \in N(u) \,|$(u,v) is oriented from v to u$\}$
and
$OUT_u = \{v \in N(u) \,|$(u,v) is oriented from u to v$\}$.

If IN_u is empty, u chooses its colour, namely the smallest numbered colour not already chosen by a neighbour, and sends its colour to every neighbour in OUT_u. While IN_u is not empty, if u receives a colour c from a neighbour v, then it removes v from IN_u and removes c from its possible colours.

Remark 6. Note that the messages sent and received (using instructions send and receive) do not contain any information on the vertices' identities but only one bit 0 or 1.

4.2. The 3-Colouring Algorithm

Now, we can give a precise description of the 3-colouring algorithm. It is given as Algorithm 3, and it consists of the composition of Algorithm 1 and of Algorithm 2. In this section, we study the time complexity of the $3-$colouring of the ring graph. Métivier et al. [MRSDZ10] show that for graphs with bounded degrees, the time complexity of the colouring is $O(\log n)$ on average and w.h.p. By using Mellin transform, we get a more precise result on ring graphs. We prove the following theorem:

Theorem 5. *The 3-colouring algorithm computes a $3-$colouring for any ring graph of size n in $3\log n$ rounds on average and w.h.p..*

To prove this theorem, we need to study the time complexity of the decreasing colours algorithm. We have:

Lemma 6. *The time complexity of the decreasing colours algorithm is less than $e + 2\log_2 n$ with high probability.*

Proof. Let $t \geq 1$. A vertex v is coloured within t steps after the end of the colouring algorithm, unless there is a path of length $t + 1$ starting at v with monotonically decreasing x values.

There are 2 paths of length $t + 1$ starting at v and each one has probability $1/(t + 1)!$ of having monotonically decreasing x values. Thus, if we denote by

Algorithm 2 The Decreasing Colours Algorithm.

1: **var:**
2: $FC_u : Integer$ **Init** $c(u)$; (* The final colour of u *)
3: IN_u: $\subseteq N(u)$ **Init** $\emptyset$;
4: OUT_u: $\subseteq N(u)$ **Init** $\emptyset$;
5: $Colours_u$: set of colours **Init** $\{0, 1, 2\}$;
6: **for each** $v \in N(u)$
7: **if**$(x(u) < x(v))$ **then** //This can be decided as soon as u and v generate two different bits
8: $OUT_u = OUT_u \cup \{v\}$
9: **else**
10: $IN_u = IN_u \cup \{v\}$
11: **end if**
12: **end for**;
13: **while** $\exists v \in IN(u)$ **do**
14: receive $< c, w >$ from a neighbour w;
15: $IN_u = IN_u \setminus \{w\}$;
16: $Colours_u = Colours_u \setminus \{c\}$
17: **end while**
18: $FC_u = \min\{c \in Colours_u\}$;
19: **for each** $v \in OUT_u$
20: send $< FC_u, v >$
21: **end for**

Algorithm 3 The 3-colouring Algorithm

1: Do Algorithm 1;
2: Do Algorithm 2.

$p_{\geq t}(v)$ the probability that v is not coloured within t steps after the end of the colouring algorithm, then:

$$p_{\geq t}(v) \leq \frac{2}{t!}.$$

Using the Stirling formulae, we have:

$$p_{\geq t}(v) \leq \left(\frac{e}{t}\right)^t.$$

Hence, if $t > e + 2\log_2 n$, we get:

$$p_{\geq t}(v) \leq \left(1 - \frac{2\log n}{e + 2\log_2 n}\right)^{e+2\log_2 n} \leq e^{-2\log_2 n} = \frac{1}{n^{2.88\ldots}}.$$

Summing over all the vertices, this gives that the probability of existence of any such path is $o(n^{-1})$ and the average length of the longest decreasing path is at most $t + 2$. We conclude that the algorithm terminates in time $e + 2\log_2 n$ w.h.p. □

Remark 7. The result on the complexity of the decreasing colours algorithm is obtained by the fact that we first apply the colouring algorithm and by the fact that the colours obtained after the colouring algorithm are interpreted as *random* real values. The analysis of the decreasing colours algorithm is based on the uniformity of colours assigned to vertices and cannot be applied in general cases (for example, for the algorithm of [Joh99]). Note as usual that the messages contain only one bit and so, the results hold for the bit complexity.

5. Time Bounds for the 3-Colouring Algorithm

Although the colouring algorithm takes $\log_2 n$ on average and up to $2\log_2 n$ and the decreasing colours algorithm takes up to $2\log_2 n$ and it cannot start *locally* until the colouring algorithm is complete, the decreasing colours algorithm may start in some vertices before the colouring algorithm is complete *globally* so that the two algorithms can overlap giving a total time less than the obvious upper bounds of approximately $3\log_2 n$ on average and $4\log_2 n$ w.h.p.

We have the following theorem :

Theorem 7. *The total time for the* 3*-colouring algorithm is* $\log_2 n + o(\log n)$ *on average and* $2\log_2 n + o(\log n)$ *w.h.p.*

Proof. We need to be more precise about the messages sent after the end of the colouring algorithm. At the end of the colouring algorithm on an edge (u, v), the two vertices know which will be the first to be ready to start the decreasing colours algorithm; suppose that it is u. The vertex u will now send 0 to v at each step until it is ready to start the decreasing colours algorithm at which point it will send a 1-bit to signal that it is starting the decreasing colours algorithm. It will then send its colour c in unary as c 0-bits followed by one 1-bit at which

time the algorithm is complete on this edge. We note that v is ready to start the decreasing colours algorithm as soon as it has received the 1 from u (and in general its other neighbours with smaller x values) and that, although it does not, at this moment, know its own colour, after $t+1$ further steps, it knows whether u (and any of the others) has the colour t and so can decide whether its own colour should be t. Thus, in the case of the ring, a vertex finishes the decreasing colours algorithm at most four steps after being ready to start it.

If a vertex v has not completed the decreasing colours algorithm at time t, then there exist a time $t' \leq t$ and a chain $v_0 = v, v_1, \cdots, v_{t-t'}$ such that $v_{t-t'}$ had not completed the colouring algorithm at time t', and the x values of the vertices in the chain are monotonic decreasing.

We consider the probability that these two events occur for a given t and t'. Given the set S of values $x_0, x_1, \cdots x_{t-t'}$, the probability that they occur in monotonic decreasing order on the chain is $1/(t-t'+1)!$. Given this monotonicity, the probability that $v_{t-t'}$ had not completed the decreasing colours algorithm at time t', is the probability of at least one of two independent events: that the other neighbour of $v_{t-t'}$ has an x which agrees with $x_{t-t'}$ on its first t' bits and that the two smallest elements of the set S agree on their first t' bits. The first of these has probability $2^{-t'}$. We now consider the probability that, in a set of m reals (independently uniform on $(0 \cdots 1)$) the two smallest elements agree on their first b bits. This is the case exactly if, in putting m objects uniformly into 2^b boxes, the first occupied box contains at least two elements. It is easily seen that when the number of boxes is doubled, the expected number of other elements in the same box as the first element is divided by at least 2; thus after b such doublings, this expected number is at most $(m-1)/2^b$, so that the probability that it is positive is also at most $(m-1)/2^b$. In our case, this is $(t-t')/2^{t'}$. Adding in the probability for v's other neighbour, we have probability at most $(t-t'+1)/2^{t'}$ that v is not ready to start the decreasing colours algorithm at time t'.

Hence, the probability we are looking for is at most $(t-t'+1)/(2^{t'}(t-t'+1)!)$ that is $1/(2^{t'}(t-t')!)$. The sum of this probability over all t' is less than $e^2/2^t$. Taking into account the $2n$ chains of a given length, we see that the expected number of vertices which have not completed the decreasing colours algorithm after time $t+8$ is less than $2n/2^t$. We conclude finally that the probability that there exists some vertex which has not completed the decreasing colours algorithm after $\log_2 n + 9 + i$ steps is less than 2^{-i} so that the average is less than $\log_2 n + 10$ and the probability is $o(n^{-1})$ when $t = 2\log_2 n + f(n)$ for any unbounded increasing function f. □

6. Lower Bounds for the Randomised Ring Colouring Problem

Kothapalli et al [KOSS06] show that any Las Vegas algorithm computing a proper vertex colouring on rings takes time $\Omega(\log n)$ with high probability. Their proof is for algorithms which have three possibilities for the action on an edge at each step, namely *send* 0, *send* 1 and *send nothing*. We present here a modified version of their proof for algorithms which do not have the third

possibility of *send nothing*. We obtain exact values for the constant hidden in the $\Omega()$ result and, thereby, show that our algorithm is optimal in two respects.

Theorem 8.
- *A Las Vegas algorithm A which computes a proper colouring on rings takes time at least $\log_2 n - o(\log n)$ with high probability.*
- *No Las Vegas algorithm A which computes a proper colouring on rings runs in time $k \log_2 n$ with high probability for any $k < 2$.*

PROOF. Consider the cycle of n nodes and let $S_l = (u_l, \cdots, u_1, v_1, \cdots, v_l)$ be the set of nodes along a path of length $2l$ of the cycle. Initially, every node in S_l is in the same state s_0, with the only difference that for every $i \in \{1, \cdots, l-1\}$, u_i considers its left connection to go to u_{i+1} whereas v_i considers its left connection to go to v_{i+1}. (Notice that the cycle is non-oriented, so we can choose any orientation we want for the individual nodes.) Associated with s_0 is a fixed probability distribution $P_\epsilon = (p_x^\epsilon)_{x\in\{0,1\}}$ for sending bit x along the right edge, where ϵ represents the empty history. Since P_ϵ has only two probability values, u_1 and v_1 send the same bit with probability at least $1/2$. Let E_1 be the event that this happens. Then u_1 and v_1 receive the same bit from their right neighbour. Let $P_y = (p_x^y))_{x\in\{0,1\}}$ be the probability distribution for sending bit x along the right edge in the second round given that bit y was received along the right edge in the first round. If E_1 occurred, P_{x_0} applies to u_1 and v_1 for the same x_0. Since P_{x_0} has only two probability values, u_1 and v_1 send the same bit with probability at least $1/2$. Let E_2 be the event that this happens. Then u_1 and v_1 again receive the same bit from their right neighbour.

Continuing with this argument, it follows that there are events $E_1, \cdots, E_l$ with E_i having a probability of at least $1/2$ for all i (conditional on $E_1, \cdots, E_{i-1}$ having occurred) that u_1 and v_1 have received the same information from their right neighbours. Algorithm A cannot terminate in this case because in this case the same probability distribution for choosing a colour applies to u_1 and v_1, and hence, the probability that u_1 and v_1 choose the same colour is non-zero.

The probability that $E_1, \cdots, E_l$ occur is at least 2^{-l}. Moreover, notice that $E_1, \cdots, E_l$ only depend on the nodes in S_l because information can only travel a distance of l edges in l rounds. Hence, we can partition the n-node cycle into $n/(2l)$ sequences S where each sequence has a probability of at least 2^{-l} of running into the events $E_1, \cdots, E_l$ that is independent of the other sequences.

For the first claim of the theorem, we take $l = \log_2(n/2\log_2^2 n)$, so that $2^{-l} = 2\log_2^2 n/n$. Thus the probability that all node sequences avoid the event sequence $E_1, \cdots, E_l$ which is necessary for A to terminate is at most $(1 - 2\log_2^2 n/n)^{n/2l} \leq 1/n$, which implies that A needs $\log_2 n - O(\log_2 \log n)$ bit-rounds, with high probability, to finish.

For the second claim, we take $l = 2\log_2(n/\log_2 n)$, so that $2^{-l} = (\log_2 n/n)^2$. Taking n large enough for the probability of two sequences running into $E_1, \cdots, E_l$ to be negligible, we find that the probability that some sequence runs into these events is at least $\log_2 n/4n$. Thus, if algorithm A terminates in time l with probability $1 - o(n^{-1})$, it sometimes fails to compute a proper colouring.□

From this result and Theorem 7, we deduce :

Theorem 9. *The* 3*-colouring algorithm is achieved in times matching the lower bounds with no restriction on the number of colours used.*

7. Conclusion

This paper analyses the time complexities of a randomised colouring algorithm and a randomised 3-colouring algorithm which uses messages containing 1 bit. We study the complexities on average and w.h.p. of these problems. Then, we prove that the complexities, on average or w.h.p., of the presented algorithms match lower bounds. These results have been obtained using the Mellin transform.

References

[AW04] H. Attiya and J. Welch. *Distributed Computing*. Wiley, 2004.

[BMW94] H. L. Bodlaender, S. Moran, and M. K. Warmuth. The distributed bit complexity of the ring: from the anonymous case to the non-anonymous case. *Inf. and comput.*, 114(2):34–50, 1994.

[CV86] R. Cole and U. Vishkin. Deterministic coin tossing and accelerating cascades: micro and macro techniques for designing parallel algorithms. In *Proceedings of the 18th ACM Symposium on Theory of computing (STOC)*, pages 206–219, 1986.

[DMR08] Y. Dinitz, S. Moran, and S. Rajsbaum. Bit complexity of breaking and achieving symmetry in chains and rings. *Journal of the ACM*, 55(1), 2008.

[FRS85] Ph. Flajolet, M. Régnier, and R. Sedgewick. Some uses of the Mellin integral transform in the analysis of algorithms. In A. Apostolico and Z. Galil, editors, *Combinatorial Algorithms on Words*, volume 12 of *Series F: Computer and Systems Sciences*, pages 241–254. NATO Advance Science Institute Series, Springer Verlag, 1985.

[FS96] Ph. Flajolet and R. Sedgewick. The average case analysis of algorithms: Mellin transform asymptotics. Number RR-2956. INRIA, 1996.

[Joh99] Ö. Johansson. Simple distributed $(\Delta + 1)$-coloring of graphs. *Information Processing Letters*, 70(5):229–232, 1999.

[KN99] E. Kushilevitz and N. Nisan. *Communication complexity*. Cambridge University Press, 1999.

[KOSS06] K. Kothapalli, M. Onus, C. Scheideler, and C. Schindelhauer. Distributed coloring in $O(\sqrt{\log n})$ bit rounds. In *20th International Parallel and Distributed Processing Symposium (IPDPS 2006), Proceedings, 25-29 April 2006, Rhodes Island, Greece*. IEEE, 2006.

[KW06] F. Kuhn and R. Wattenhofer. On the complexity of distributed graph coloring. In *Proceedings of the 25 Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pages 7–15. ACM Press, 2006.

[Lin92] N. Linial. Locality in distributed graph algorithms. *SIAM J. Comput.*, 21:193–201, 1992.

[MRSDZ10] Y. Métivier, J. M. Robson, N. Saheb-Djahromi, and A. Zemmari. About randomised distributed graph colouring and graph partition algorithms. *Inf. Comput.*, 208(11):1296–1304, 2010.

[Pel00] D. Peleg. *Distributed computing - A Locality-sensitive approach*. SIAM Monographs on discrete mathematics and applications, 2000.

[Ros00] *Handbook of Discrete and Combinatorial Mathematics*. CRC Press, 2000.

[Yao79] A. C. Yao. Some complexity questions related to distributed computing. In *Proceedings of the 11th ACM Symposium on Theory of computing (STOC)*, pages 209–213. ACM Press, 1979.