An Analysis of an Optimal Bit Complexity Randomised Distributed Vertex Colouring Algorithm
(Extended Abstract)

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

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

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.

In this paper we discuss how efficient (time and bits) vertex colouring may be accomplished by exchange of bits between neighbouring vertices. 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 graph \( G = (V, E) \) where the vertices of \( V \) 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.

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

⋆ This work was supported by grant No ANR-06-SETI-015-03 awarded by Agence Nationale de la Recherche
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.** As is explained by Santoro in [San07] (Chapter 6) (see also [Gho06], Chapter 3) the cost of a synchronous distributed algorithm is both time and bits (whether a message contains 1 bit or it contains the Encyclopedia Britannica does not have the same cost). By definition, the bit complexity of a distributed algorithm (per channel) is the total number of bits exchanged (per channel) during its execution. Thus 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].

**Network and Processes Knowledge.** The network is anonymous: unique identities are not available to distinguish the processes. We do not assume any global knowledge of the network, not even its size or an upper bound on its size. The processors do not require 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.

### 1.3 Our Contribution

We present and analyse a randomised distributed vertex colouring algorithm. In this paper, we prove Theorem 1:

*There exists a randomised distributed colouring algorithm for arbitrary graphs of size $n$ that halts in time $O(\log n)$ with probability $1 - o(n^{-1})$ each message containing 1 bit.*

Kothapalli et al. show in [KOSS06] that if only one bit can be sent along each edge in a round, then every distributed vertex colouring algorithm (in which every node has the same initial state and initially only knows its own edges) needs at least $\Omega(\log n)$ rounds with high probability\(^1\) (w.h.p. for short) to colour the cycle of size $n$ with any finite number of colours. From this result we deduce that our algorithm is optimal (bits and time) modulo multiplicative constants.

### 1.4 Related Work: Comparisons and Comments

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.

\(^1\) With high probability means with probability $1 - o(n^{-1})$. 
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 $\Delta$ 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 $O(\log n)$ bit identifier. More recently, Kuhn and Wattenhofer [KW06] have obtained 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]. This 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 $\{0, \ldots, d\}$ 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.

Johanson proves that this algorithm runs in $O(\log n)$ rounds with high probability on graphs of size $n$. The size of each message is $\log n$ thus the bit complexity per channel of this algorithm is $O(\log^2 n)$.

In fact this algorithm may be viewed as a particular case of a general family of randomised algorithms in which vertices wake up with a certain probability to participate in a round (see [Lub93,FPS04]).

The table below summarises the comparison between Johanson’s Algorithm and the algorithm presented in this paper: Algorithm $\mathcal{B}$.

<table>
<thead>
<tr>
<th></th>
<th>Time</th>
<th>Order of the colour of a vertex of degree $d$</th>
<th>Message size (number of bits)</th>
<th>Bit complexity (per channel)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Johansson’s algorithm</td>
<td>$O(\log n)$</td>
<td>$d + 1$ (guaranteed)</td>
<td>$\log n$</td>
<td>$O(\log^2 n)$</td>
</tr>
<tr>
<td>Algorithm $\mathcal{B}$</td>
<td>$O(\log n)$</td>
<td>$O(d)$ (on average)</td>
<td>1</td>
<td>$O(\log n)$</td>
</tr>
</tbody>
</table>

## 2 Algorithm $\mathcal{B}$

We describe the algorithm executed by each node. Our algorithm operates in synchronous rounds. At each round, some vertices are permanently coloured, and stop executing the algorithm, some edges are removed from the graph (symmetry
is broken) 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 symmetry is not yet broken; initially $active_v$ is equal to $N(v)$ (the set of 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 $b_v \oplus colour_v$; 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 all its neighbours and hence the vertex has obtained its final colour.

Remark 1. The colour of a node $v$ is the concatenation of all the bits generated since from the start of execution of the algorithm to the time where $v$ has no active neighbours. In a natural way, it may be interpreted as an integer.

3 Analysis of the Algorithm

The Expected Time Complexity. Each vertex $v$ which is not already coloured generates a bit $b_v$, sends $b_v$ to all its still active neighbours and receives $b_u$ from each active neighbour $u$. If $b_v \neq b_u$ then $v$ updates its list of active neighbours by dropping $u$ from this list. In terms of the graph structure, this means that the edge $\{u, v\}$ is removed from the graph. Since the event $b_v \neq b_u$ occurs with probability $1/2$, it is easy to prove that:

Lemma 1. Let $(G_i)_{i \geq 0}$ be the sequence of residual graphs obtained with Algorithm B. The expected number of edges removed from the residual graph $G_i$ after the $(i + 1)^{th}$ round is half the number of its edges.

Then, we have:

Corollary 1. There is a constant $k_1$ such that for any graph $G$ of $n$ vertices, the number of rounds to remove all edges from $G$ is less than $k_1 \log n$ on average.

Now we prove a more precise result:

Lemma 2. There is a constant $K_1$ such that for any graph $G$ of $n$ vertices, the number of rounds to remove all edges from $G$ is less than $K_1 \log n$ w.h.p.

Proof. Initially the number of edges is less than $n^2/2$. Therefore after $r$ rounds the expected number of edges remaining is less than $n^2/2^{r+1}$. In particular after $4 \log n - 1$ rounds it is less than $n^{-2}/2$ so that the probability that any edges remain is less than $n^{-2}/2$. $\Box$

Finally:
Theorem 1. Algorithm B computes a colouring for any arbitrary graph of size $n$ in time $O(\log n)$ w.h.p., each message containing 1 bit.

Local Complexity. In this section, we study the expected number of bits generated for a given vertex $v$ with degree $d(v) = d$. Let $L_d$ denote the number of bits generated on the vertex $v$ and let $l_d = E(L_d)$ its expected value. Let $I(v)$ denote the set of edges incident to $v$. At each round, any $e \in I(v)$ is removed with probability $1/2$, then the same arguments as in the previous section can be used to prove that $E(L_d)$ is $O(\log d)$. However, if $d \rightarrow \infty$, using the Mellin transform, we prove a more precise result:

Proposition 1. Let $G = (V, E)$ be a connected graph and $v \in V$ with $d(v) = d \rightarrow \infty$. Let $l_d$ denote the expected number of bits generated by $v$ before it obtains its final colour. Then $l_d = \log_2 d + 1/12 + \frac{\pi^2}{6} - P(\log_2 d) + O(d^{-2})$, where $P(u) = 1 - 2\log_2 u + 2\log_2 u$ is a Fourier series with period 1 and with an amplitude which does not exceed $10^{-6}$.

Corollary 2. Let $v$ be a vertex; let $c(v)$ be the colour of $v$ interpreted as an integer; then $E(c(v)) = O(d)$.

More exactly, we have the following result on the distribution of the r.v. $L_d$:

Lemma 3. Let $d \geq 1$. We have: $\Pr(L_d = 0) = 0$, and $\Pr(L_d = k) = (1 - \frac{1}{2^k})^d - (1 - \frac{1}{2^k})^d$, if $k \geq 1$.

Lemma 3 gives the probability distribution of the r.v. $L_d$. Thus, we can derive its variance:

Lemma 4. Let $\text{Var}$ denote the variance, if $d \rightarrow \infty$, then we have $\text{Var}(L_d) = (\frac{1}{\log 2} - 1) \log_2 d + \frac{1}{12} + \frac{\pi^2}{6} - P(\log_2 d) + O(d^{-2})$, where $P(u) = Q(u)^2 + 2u + \frac{\pi^2}{2}\log_2 u$ and $Q$ is the Fourier series defined in Proposition 1.

We then have:

Proposition 2. The ratio $R_d$ between $L_d$ and $\log_2 d$ tends in probability to 1 as $d$ tends to $\infty$.

Now we can state a result more precise than Corollary 2:

Corollary 3. Let $v$ be a vertex; let $c(v)$ be the colour of $v$ interpreted as an integer. Then for any $\varepsilon > 0$, $\lim_{d \rightarrow \infty} \Pr(c(v) \leq d^{1+\varepsilon}) = 1$.

4 Conclusion and Further Developments

Algorithm B is a very simple and natural vertex colouring. In this paper we analyse its time and bit complexity (on an edge, on a vertex and over all the graph). To our knowledge this kind of analysis has never been done before. This analysis is non trivial and to obtain precise results we use some tools like the Mellin transform. It needs no initial knowledge and from the work of Kothapalli
et al. [KOSS06] we deduce that it is optimal (modulo a multiplicative constant). Johansson’s algorithm ensures a $d+1$ colouring for a vertex of degree $d$, while our algorithm needs a priori $O(d)$ colours (on average) for the same vertex. There are two natural questions, with the same initial knowledge: (1) Is it possible to estimate the number of different colours Algorithm $B$ needs? (2) Is it possible to improve the number of colours, using an optimal algorithm (for the bit complexity)?

Acknowledgement

The authors are grateful to Christian Lavault for the corrections he suggested in the proofs using the Mellin transform.

References


