On Fast Randomized Colorings in Sensor Networks

N. Mitton† E. Fleury‡ I. Guérin Lassous† B. Sericola⋆ S. Tixeuil⋄

† INRIA ARES / CITI – INSA de Lyon, Lyon, France,
Nathalie.Mitton@insa-lyon.fr, Isabelle.Guerin-Lassous@inria.fr
‡ CITI – INSA de Lyon / INRIA ARES, Lyon, France,
Eric.Fleury@inria.fr
⋆ ARMOR / INRIA – IRISA, Rennes, France
Bruno.Sericola@irisa.fr
⋄ LRI - CNRS UMR 8623 / INRIA Grand Large, Orsay, France
Sebastien.Tixeuil@lri.fr

1This work is supported by the FNS of the French Ministry of Research through the FRAGILE
Abstract

We present complexity analysis for a family of self-stabilizing vertex coloring algorithms in the context of sensor networks. First, we derive theoretical results on the stabilization time when the system is synchronous. Then, we provide simulations for various schedulings and topologies. We consider both the uniform case (where all nodes are indistinguishable and execute the same code) and the non-uniform case (where nodes make use of a globally unique identifier).

Overall, our results show that the actual stabilization time is much smaller than the upper bound provided by previous studies. Similarly, the height of the induced DAG is much lower than the linear dependency to the size of the color domain (that was previously announced). Finally, it appears that symmetry breaking tricks traditionally used to expedite stabilization are in fact harmful when used in networks that are not tightly synchronized.

Résumé

Nous présentons des analyses de complexité pour une famille d’algorithmes auto-stabilisants de coloriage de nœuds dans le contexte des réseaux de capteurs. Tout d’abord, nous présentons des résultats théoriques sur le temps de stabilisation quand le système est synchrone. Puis, nous proposons des simulations pour plusieurs types d’ordonnancements ou de topologies. Nous considérons à la fois le cas uniforme (où tous les nœuds sont indistingubles et exécutent le même code) et le cas non-uniforme (où les nœuds peuvent faire usage d’un identifiant globalement unique).

En résumé, nos résultats montrent que le temps de stabilisation véritable est largement plus petit que les bornes supérieures présentées dans les études antérieures. De manière similaire, la hauteur du DAG induit est beaucoup plus petite que la dépendance linéaire à la taille du domaine des couleurs (ce qui correspond aux résultats précédents). Enfin, les astuces habituellement utilisées pour accélérer la stabilisation sont en fait néfastes quand les réseaux considérés ne sont pas fortement synchronisés.
Chapter 1

Introduction

In the context of computer networks, resuming correct behavior after a fault occurs can be very costly \cite{14}: the whole network may have to be shut down and globally reset in a good initial state. While this approach is feasible for small networks, it is far from practical in large networks such as forecast sensor networks. Self-stabilization \cite{2, 3} provides a way to recover from faults without the cost and inconvenience of a generalized human intervention: after a fault is diagnosed, one simply has to remove, repair, or reinitialize the faulty components, and the system, by itself, will return to a good global state within a relatively short amount of time. The vertex coloring problem, issued from classical graph theory, consists in choosing different colors for any two neighboring nodes in a graph. This problem can easily be generalized to distance $k$, requiring that any two nodes that are up to $k$ hops away must have different colors. In Distributed Computing, vertex coloring algorithms are mainly used for resource allocation (see \cite{11} for more details). Self-stabilizing distributed vertex coloring was previously studied for planar \cite{6}, bipartite \cite{16, 15}, and general \cite{8, 9} graphs, trying to minimize the number of used colors. A common drawback to those approaches is that the stabilization time depends on $n$, the size of the network, so that they do not scale well. A self-stabilizing vertex coloring at distance two that has expected constant stabilization time was presented in \cite{7}, but requires $O(n^2)$ colors in a $n$-sized network, so it also depends on some global network parameter.

Recently, in the context of sensor networks, a simple distance-$k$ coloring algorithm was presented with $k = 3$ in \cite{10} and $k = 2$ in \cite{12}. Intuitively, the algorithm runs as follows: every node repetitively collects colors chosen by its neighbors, and if it detects a conflict with its own color randomly chooses a fresh color (not taken by its distance-$k$ neighborhood). The algorithm does not really try to minimize the number of used colors (it is $\delta^6$ in \cite{10}, and $\delta^2$ in \cite{12}, where $\delta$ denotes the maximum degree of the graph), but when the graph degree is bounded by a small constant (as it is the case in sensor networks), the expected local stabilization time (i.e. the stabilization time in any neighborhood) of the algorithm is also constant. This makes the algorithm independent of $n$ and thus scalable to large networks. Not only this algorithm is simple enough that it is likely to be implemented on limited devices such as those in sensor networks, it also provides a fault containment radius of about $k$ to unbounded byzantine (i.e. malicious) failures \cite{13}. Also, the directed acyclic graph
that is induced by the colors is of constant height, so that self-stabilizing algorithms that are composed with the coloring can stabilize also in constant time. This property was used in [10] to construct a minimal distance-2 coloring, and in [12] to self-organize the network into clusters, both locally stabilizing in expected constant time.

While this distance-\( k \) coloring algorithm has provenly fast local stabilization time, the influence of system parameters remains unknown. In this paper, we study its stabilization time in various settings that are relevant to sensor networks. We provide a theoretical study in synchronous networks. Using simulations, we consider various topologies (grids, and random graphs), different kinds of scheduling hypothesis (synchronous and probabilistically asynchronous), and variants of the algorithm that uses network wide identifiers so that priorities can be derived for neighboring nodes.

The remaining of the paper is organized as follows: Chapter 2 formally presents the system model and coloring algorithm. In Chapter 3, we analytically study the stabilizing time considering a synchronous setting, while in Chapter 4 we provide extensive simulations and comments on various parameters. Chapter 5 gives concluding remarks.
Chapter 2

Preliminaries

2.1 System model

The system is composed of a set $V$ of nodes in a multi-hop wireless network, and each node has a unique identifier. Communication between nodes uses a low-power radio. Each node $p \in V$ can communicate with a subset $N_p \subseteq V$ of nodes determined by the range of the radio signal $R$; $N_p$ is called the neighborhood of node $p$. Note that $p$ does not belong to $N_p$ ($p \notin N_p$). In the wireless model, transmission is omni-directional: each message sent by $p$ is effectively broadcast to all nodes in $N_p$. We also assume that communication capability is bidirectional: $q \in N_p$ iff $p \in N_q$. Define $N^1_p = N_p$ and for $i > 1$, $N^i_p = N^{i-1}_p \cup \{ r \mid (\exists q : q \in N^{i-1}_p : r \in N_q) \}$ (let’s call $N^i_p$ the distance-$i$ neighborhood of $p$). We assume that the distribution of nodes is sparse: there is some known constant $\delta$ such that for any node $p$, $|N_p| \leq \delta$. Note that when sensor nodes are randomly spread on the area to be monitored, the average degree of a node is $\bar{\delta} = \lambda \pi R^2$ where $\lambda$ is the spatial intensity. Sensor networks can control density by adjusting their radius $R$ and/or powering off nodes in areas that are too dense, which is one aim of topology control algorithms.

2.2 Notation

We describe algorithms using the notation of guarded assignment statements: $G \rightarrow S$ represents a guarded assignment, where $G$ is a predicate of the local variables of a node, and $S$ is an assignment to local variables of the node. If predicate $G$ (called the guard) holds, then assignment $S$ is executed, otherwise $S$ is skipped. We assume that all such guarded assignments execute atomically when a message is received. At any system state, where a given guard $G$ holds, we say that $G$ is enabled at that state.
2.3 Execution and scheduling

The life of computing at every node consists of the infinite repetition of evaluating its guarded actions. The scheduler is responsible for choosing enabled processors for executing their guarded rules. In this paper, we consider three possible schedulers: the synchronous scheduler, the probabilistic central scheduler, and the probabilistic distributed scheduler. With the synchronous scheduler, nodes operate in lock steps, and at every step, every node is activated by the scheduler. At every step, the probabilistic central scheduler randomly activates exactly one node. With the probabilistic randomized scheduler, at each step, each node is activated with probability $1/n$. The last two schedulers model the fact that although nodes execute their actions at the same speed on average, there is a chance that their clocks or speeds are not uniform, so that the system is slightly asynchronous. The distributed scheduler is more realistic than the central one, but the latter is often used for proving self-stabilizing algorithms.

2.4 Shared Variable Propagation

Nodes communicate with their neighbors using shared variables. To keep the analysis simple, we assume that there exists an underlying shared variable propagation scheme that permit nodes to collect shared variables in their neighborhood at distance $k$, for a fixed $k$. A possible implementation can be found in [10]. For our purpose, we simply assume that a node is able, in one step, to read all shared variables in its neighborhood at distance $k$.

2.5 Coloring Algorithm

The coloring algorithm that we consider uses a single shared variable for each node. Let $C_p$ be a shared variable that belongs to the domain $\Delta$; variable $C_p$ is the color of node $p$. The $C_{C_p}$ predicate refers to the set of colors that have been used in the neighborhood at distance $k$ of $p$: $C_{C_p} = \{C_q \mid q \in N_k^p\}$. Let $\text{random}(S)$ choose with uniform probability some element of set $S$. Node $p$ uses the following function to compute $C_p$:

$$\text{newC}(C_p) = \begin{cases} C_p & \text{if } C_p \notin C_{C_p} \\ \text{random}(\gamma \setminus C_{C_p}) & \text{otherwise} \end{cases}$$

The algorithm for vertex coloring is the following: \textbf{N1:} true $\rightarrow C_p := \text{newC}(C_p)$

2.6 Local Stabilization

With respect to any given node $v$, a solution for the coloring problem at distance $k$ is locally stabilizing for $v$ with convergence time $t$ if, for any initial system state, after at most $t$ time units, every subsequent system state satisfies the property that any node $w$ at distance less than $k$ from $v$ is such that $C_w \neq C_v$. For randomized algorithms, this definition is modified
to specify expected convergence times (all stabilizing randomized algorithms we consider are probabilistically convergent in the Las Vegas sense).

**Theorem 1 ([10])** Algorithm N1 self-stabilizes with probability 1 and has constant expected local stabilization time.

### 2.7 Uniform vs. Non Uniform Networks

In theory, the coloring algorithm N1 could work in uniform and anonymous networks (where node do not have unique identifiers and execute the same code), collecting the neighborhood at distance $k$ generally requires identifiers. Then, it is possible to tweak the algorithm to use these identifiers to break symmetry and expedite convergence: when two neighbors have a conflicting color, the node with the lowest identifier never changes its color. Thus, we have the guarantee that at any step, at least one of the two nodes gets a stable color. In the remaining of the paper, we distinguish two operating modes for the algorithm: the **all** mode refers to the mode where all conflicting nodes draw a new color, while the **all but one** mode refers to the latter version of the algorithm.
Chapter 3

Analysis

In this chapter, we compute the expected stabilization time of the coloring protocol \texttt{N1}, i.e. the expected number of steps before a node has a color that is not in its distance-$k$ neighborhood. From \texttt{[10]}, we already know that when the degree is upper bounded by a constant, the expected local stabilization time is also upper bounded by a constant. However, the actual constant is not given in \texttt{[10]}, and the one that can be derived from the algorithm is high (about $\delta^6$ for a distance-3 coloring).

The other metric of interest for our purpose is the height of the DAG that is induced by the colors (orienting every edge from the higher color to the lower color). Indeed, when the coloring algorithm is used as a building block for subsequent algorithms, the stabilization time of those algorithms is generally in the order of the color DAG height, a lower DAG height inducing a smaller stabilization time.

**Theorem 2** (\texttt{[10]}) The height of the DAG is bounded by $|\Delta| + 1$, where $\Delta$ denotes the color domain.

For the theoretical study, we consider only the synchronous scheduler, and we model the coloring protocol by a successive set of random draws. The main goal is to assign a color to each node. A way to model this problem is to consider that the color domain is represented by a set of $M$ urns in which one must randomly distribute $L$ balls which are going to represent the nodes. As stated before, in each neighborhood, the goal is to have only one ball (Node) associated to a given urn (Color).

This model has already been used in \texttt{[1]} (for the NAP protocol), in which the authors analyze the stabilization time of a self-addressing network where two links must receive a unique prefix in the network. In this model, the urns symbolize the prefix and the balls the links. Each link chooses a random prefix in a prefix domain. If two links have chosen the same prefix, the one with the lowest ID keeps it while the other one(s) choose(s) a new prefix among the ones not already taken by the other links. The analysis and the calculus carried out in \texttt{[1]} thus roughly correspond to our all but one mode. In fact, this analytical model matches our sensor model only in complete graphs. Otherwise, it is possible that two neighboring nodes with no conflicting colors simultaneously draw a new identical color, when
they each have another conflicting neighbor (not visible to the first neighbor). Overall, this theoretical study gives a lower bound on the actual stabilization time (this is further refined in Chapter 4). The algorithm can be modeled in terms of urns/balls as follows:

**Algorithm 3.1 Coloring Process \((L, M)\)**

- **Input:** \(M\) urns and \(L\) balls
- **Pre-condition:** \(M \geq L\)

  ```java
  if \((L \neq 0)\) then
    Randomly throw the \(L\) balls in the \(M\) urns;
    if (case = 'All') then
      Keep aside all urns with their balls inside that contain exactly one ball;
    end
    if (case = 'All but One') then
      Keep aside all urns containing at least one ball and one of the balls contained inside;
    end
    Let note \(c \leq M\) the number of "correct" urns that we keep aside;
    Call Coloring Process \((L - c, M - c)\);
  end
  
  By repeating the process, eventually every ball will be stored in a correct urn and every urn will contain at most one ball. Let \(N\) denote the number of iterations needed to reach such a configuration, i.e. the number of calls of the recursive procedure of coloring. We are interested in computing the distribution and the expectation of the random variable \(N\). We consider the homogeneous discrete-time Markov chain \(X = \{X_n, n \in \mathbb{N}\}\) on the finite state space \(S = \{0, 1, \ldots, L\}\) where the event \(\{X_n = i\}\) represents the fact that, after \(n\) transitions (or calls to the coloring procedure), the procedure Coloring Process \((M - i, L - i)\) is currently executed. In other words, \(\{X_n = i\}\) represents the fact that, after \(n\) transitions, exactly \(i\) urns and balls have been kept aside, i.e. exactly \(i\) urns contain exactly one ball. The Markov chain starts in state 0 with probability 1, which means that at the beginning, none of the urns and balls have been kept aside. The random variable \(N\) can be defined more formally, for \(L \geq 1\), as \(N = \sum_{n \geq 0} \sum_{i=0}^{L-1} 1\{X_n = i\}\). \(N\) is the number transient states of \(X\) visited before absorption.

We denote by \(\mathcal{P}(L, M) = (p_{i,j}(L, M))_{(i,j) \in S^2}\) the transition probability matrix of the Markov chain \(X\), where \(p_{i,j}(L, M)\) represents the probability to have exactly \(j\) urns and balls kept aside at the time \(n + 1\) given that \(i\) urns and balls are kept aside at time \(n\). \(p_{i,j}(L, M)\) is thus the probability to obtain exactly \(j - i\) urns each with one ball when throwing \(L - i\) balls into \(M - i\) urns. For all \(i \leq j\), we thus have

\[
p_{i,j}(L, M) = p_{0,j-i}(L - i, M - i).
\]  

(3.1)

The Markov chain \(X\) is clearly acyclic and state \(L\) is the absorbing state of \(X\). This means that for every \(i \in S - \{L\}\) and \(j \in S\), we have \(p_{i,j}(L, M) = 0\) for \(i > j\) and \(p_{L,L}(L, M) = 1\).

The transition probability matrix \(\mathcal{P}(L, M)\) has thus the following form, where we write
$p_{i,j}$ instead of $p_{i,j}(L, M)$:

$$
P(L, M) = \begin{pmatrix}
p_{0,0} & p_{0,1} & p_{0,2} & \cdots & \cdots & p_{0,L} \\
0 & p_{1,1} & p_{1,2} & \cdots & \cdots & p_{1,L} \\
\vdots & \vdots & \ddots & \cdots & \ddots & \vdots \\
0 & 0 & \cdots & 0 & p_{L-1,L-1} & p_{L-1,L} \\
0 & 0 & \cdots & 0 & 1 & 0
\end{pmatrix}.
$$

The computation of the matrix $P(L, M)$ depends on the drawing hypothesis ('All but one' or 'All' modes). But, once the transition probability matrix is computed, the way of evaluating the stabilization time of the coloring algorithm is the same whatever the drawing hypothesis. The mode 'All but one' is a little bit simpler because in that case we have $p_{i,i}(L, M) = 0$ for every $0 \leq i \leq L - 1$, which means that the stabilization time $N$ of the coloring algorithm is bounded i.e. we have $N \leq L$. We give in the following section the transition probability matrix of each mode before giving the distribution and the expectation of the stabilization time $N$ of the coloring algorithm.

### 3.1 Computation of the transition matrix $P(L, M)$.

We are now going to explicit the transition probability $p_{i,j}(L, M)$ of the Markov chain $X$ for all $i \leq j$ for both modes.

In the all mode, all nodes in conflict choose a new color among the free ones. At each step, we keep aside the urns and their balls if they contain exactly one ball. Unlike the ”All but one” mode, we may keep none urn and ball at a step and thus the diagonal values of $P(L, M)$ are not null.

From relation (3.1), we only have to compute the transition probability $p_{0,j-i}(L-i, M-i)$ for $i \leq j$. The computation of the transition probabilities $p_{i,j}(L, M)$ yields to the computation of the transition probabilities $p_{0,j}(L, M)$.

By definition of the transition matrix as stated above, the transition probability $p_{0,j}(L, M)$ is the probability to obtain exactly $j$ urns containing only one ball when throwing randomly $L$ balls into $M$ urns. The case $j = L$ yields to the generalized birthday problem and thus we have:

$$p_{0,L}(L, M) = \frac{M!}{(M-L)!M^L} \quad (3.2)$$

For $j < L$, we proceed as follows. We throw $L$ balls in $M$ urns. We denote by $K_0(L, M)$ the number of empty urns and by $K_1(L, M)$ the number of urns with exactly one ball inside. Let us denote by $a_{L,M}(k, j)$ the joint distribution of the two random variables $K_0(L, M)$ and $K_1(L, M)$, that is $a_{L,M}(k, j) = \mathbb{P}[K_0(L, M) = k, K_1(L, M) = j]$. We can thus express the transition probability $p_{0,j}(L, M)$ as

$$p_{0,j}(L, M) = \mathbb{P}[K_1(L, M) = j] = \sum_{k=0}^{M} a_{L,M}(k, j).$$
In order to compute all the probabilities \( a_{L,M}(k, j) \) we can proceed by recursion on integer \( L \) by conditioning on the result of the throw of the last ball. More precisely, in order to obtain \( k \) empty urns and \( j \) urns with only one ball inside when throwing \( L \) balls in \( M \) urns we need:

1. Either obtaining \( k + 1 \) empty urns and \( j - 1 \) urns with exactly one ball inside at the end of the \( L - 1 \) first throws of \( L - 1 \) balls in \( M \) urns and throwing the last ball in one of the \( k + 1 \) empty urns.

2. Either obtaining \( k \) empty urns and \( j + 1 \) urns with exactly one ball inside at the end of the \( L - 1 \) first throws of \( L - 1 \) balls in \( M \) urns and throwing the last ball in one of the \( j + 1 \) urns containing exactly one ball.

3. Or obtaining \( k \) empty urns and \( j \) urns with exactly one ball inside at the end of the \( L - 1 \) first throws of \( L - 1 \) balls in \( M \) urns and throwing the last ball in one of the \( M - (j + k) \) urns that contain at least 2 balls.

Thus, from that decomposition and for \( L \geq 2 \), we have:

\[
a_{L,M}(k, j) = \frac{k + 1}{M} a_{L-1,M}(k+1, j-1)1_{\{j \geq 1\}} + \frac{j + 1}{M} a_{L-1,M}(k, j+1) + \frac{M - (j + k)}{M} a_{L-1,M}(k, j)
\]

where \( 1_{\{c\}} \) is the indicator function equal to 1 if condition \( c \) is true and 0 otherwise. For \( L = 1 \) we trivially have as initial value for the recursion:

\[
a_{1,M}(k, j) = 1_{\{k=M-1, j=1\}}.
\]

It is also easy to check that \( a_{L,M}(k, j) = 0 \) if either \( j > L \) either \( k = M \) or \( j + k > M \). Note that for \( j = L \), we have \( a_{L,M}(k, L) = 0 \) for \( k \neq M - L \) and \( a_{L,M}(M - L, L) = p_{0,L}(L, M) \) which is given by relation (3.2) and that clearly satisfies the recurrence relations.

In the all but one mode, at each step, we keep aside at least one urn together with one of the balls contained inside. This means that the transition probability \( p_{i,j}(L, M) = 0 \) for every \( 0 \leq i \leq L - 1 \). The Markov chain \( X \) is thus strictly acyclic (acyclic without loops) and the stabilization time \( N \) of the coloring algorithm is bounded by \( L \). This case is the one given in [1]. We only give here the results and a brief explanation.

Based on a well-known result about the number of ways of throwing \( r \) different balls in \( n \) different urns such that exactly \( m \) urns are non-empty [4], the transition probability matrix is given according to (3.1), for \( i < j \), by

\[
p_{i,j}(L, M) = \binom{M-i}{j-i} \sum_{k=0}^{j-i} \binom{j-i}{k} (-1)^k \frac{(j-i-k)^{L-i}}{M-i}.
\]

### 3.2 Distribution and expectation of \( N \).

Now given the transition probability matrix of the Markov chain \( X \), we are able to determine the distribution \( \mathbb{P} [N = n] \) for \( n = 0, \ldots, \infty \) and the expected value \( \mathbb{E} [N] \) of the random
variable $N$. The mean value $\mathbb{E}[N]$ also actually is the mean number of steps needed before stabilization of our coloring algorithm and so the stabilization time.

These calculus are detailed in [1]. We directly use them here. They are easily derived from the classical results on Markov chains.

Let $Q$ denote the sub-matrix obtained from $\mathcal{P}(L, M)$ by removing the last line and the last column which correspond to the absorbing state $L$. We denote by $\alpha$ the row vector containing the initial probability distribution of the transient states of $X$ i.e. $\alpha = (\mathbb{P}[X_0 = i])_{i=0,\ldots,L-1}$. As already seen, the Markov chain starts in state 0 with probability 1 and, thus we have $\alpha = (1,0,\ldots,0)$. With these notations, we have

$$\mathbb{P}[N = n] = \alpha Q^{n-1}(I - Q) \mathbb{1}, \text{ for } n \geq 1,$$

$$\mathbb{P}[N > n] = \sum_{k=n+1}^{\infty} \alpha Q^{k-1}(I - Q) \mathbb{1} = \alpha Q^n \mathbb{1}, \text{ for } n \geq 0,$$

and

$$\mathbb{E}[N] = \sum_{n=0}^{\infty} \mathbb{P}[N > n] = \alpha (I - Q)^{-1} \mathbb{1},$$

where $I$ is the identity matrix and $\mathbb{1}$ is the column vector with all entries equal to 1, both of dimension $L$. If $V$ is the column vector $V = (V_i)_{0 \leq i \leq L-1}$ defined by $V_i = \mathbb{E}[N|X_0 = i]$, we have $\mathbb{E}[N] = V_0$. The vector $V$ of conditional expectations is given by $V = (I - Q)^{-1} \mathbb{1}$, which means that it is solution to the linear system $(I - Q)V = \mathbb{1}$, which can also be written as $V = 1 + QV$ or equivalently, since the matrix $\mathcal{P}(L, M)$ is acyclic, as

$$V_i = \frac{1}{1 - p_{i,i}} \left( 1 + \sum_{j=i+1}^{L-1} p_{i,j} V_j \right) \text{ for } i = L - 2, \ldots, 0,$$

with $V_{L-1,L-1} = 1/(1 - p_{L-1,L-1})$.

We are now able to compute $\mathbb{E}[N] = V_0$ by recurrence from $V_i$. We use these theoretical results in the next chapters to compare with the simulation outcomes.
Chapter 4

Simulations

We performed simulations to evaluate the stabilization time of the different coloring algorithms, to analyze the directed acyclic graph deriving from these colorings, and to compare those values to the theoretical ones in order to validate our analytical model. As mentioned in the previous chapters, we are mainly interested in the stabilization time under several scheduling and coloring hypothesis but also on the height of the derived DAG.

4.1 Simulation model.

We use a simulator we developed. In each case, each statistic is the average over 1000 simulations and we fix a minimum radius and/or number of nodes such that the network is connected. All coloring algorithms studied below are analyzed over a same node distribution.

We say that a node executes an action (or acts, for short) when it checks its neighbors’ color and if needed, chooses another color among the available ones. For a distance-\(k\) coloring, a color \(c \in \Delta\) is said available for a node \(u\) if \(\forall v \in N^k_u, C_v \neq c\). As mentioned in Chapter 2, we studied the coloring algorithm for different kinds of scheduling hypothesis: Synchronous (every node acts at every step), Probabilistic Central (exactly one random node acts at every step) and Probabilistic Distributed (at each step, each node acts with probability \(\frac{1}{n}\)). Each scheduling hypothesis is studied for both modes of coloring: all but one mode (for each pair of conflicting nodes, every node but one chooses another color) and all mode (every conflicting node chooses another color).

We ran Algorithm N1 for distance-1 and distance-2 coloring with these scheduling hypothesis and modes over a geometric topology and a grid with a varying number of neighbors per node and different sizes of color domain \(\Delta\). The initial stabilizing algorithm has been designed using \(|\Delta| = (\max_{p \in V}|N_p|)^{2^k}\), \(k\) being the coloring distance, but as we are interested in the impact of this domain size over the stabilization time and the DAG height, we also run Algorithm N1 for \(|\Delta| = 2 \times (\max_{p \in V}|N_p^k|)\). Moreover, as in sensor networks, nodes do not have a general view of the network and thus have no way a priori to know the maximum degree of the graph, we also run simulations where each node has its own color domain size such that \(\forall p \in V |\Delta_p| = (|N_p|)^{2^k}\). In these settings, we collect the stabilization time of the
coloring algorithm as well as the height of the induced DAG. In order to be able to compare
the three schedulers, the number of steps before stabilization in probabilistic modes is di-
vided by \( n \). In the geometric approach, nodes are randomly deployed using a Poisson process
in a \( 1 \times 1 \) square with various levels of intensity \( \lambda \) (from 500 to 1000). In such processes, \( \lambda \)
represents the mean number of nodes per surface unit. The communication range \( R \) is set
to 0.1 in all tests. We thus have a mean number of neighbors per node ranging from 15 to
32. In the grid, we consider topologies where each central node has 4 and 8 neighbors.

### 4.2 Theory vs. Experience.

As mentioned in Chapter 3, we expect that our theoretical results give an accurate lower
bound on the stabilization time of the coloring algorithm. Indeed, when considering a com-
plete graph, once a node \( u \) has chosen an available color, every other node in the network
knows that color (as a neighboring node of \( u \)) and thus will not choose it anymore. However,
in a non-complete graph, a situation such as illustrated in Figure 4.1 (for distance-1 coloring)
may occur and thus, the average stabilization time would be a bit higher than the one in a
complete graph, therefore our theoretical analysis only leads to a lower bound.

For example, let’s run the algorithm over the graph plotted on Figure 4.1. Every node
has to choose a color from 1 to 4 until getting a locally unique color. Node \( B \) has two
neighbors, \( A \) and \( C \). The example shows the colors drawn at each step by each node. As
there still exists conflicts at the end of the three steps, the algorithm would not stop yet.
But, from theoretical analysis, at the end of the three steps, every urn has been kept aside,
so the system should have stabilized.

![Figure 4.1: A possible execution for distance-1 coloring in a synchronous network.](image)

In order to evaluate the difference between the analytical lower bound and the simulated
stabilization time, we compute by simulation the number of times that two neighboring
nodes not originally in mutual conflict have to both choose another color and both get the
same color. Results for a distance-1 synchronous scheduling coloring are given in Figure 4.2.
As we can see, the greater the color domain size, the more unlikely this case is to appear
(almost 0 percent of the cases when the color domain size is quadratic in the maximum
degree of the nodes). We also note that, even for a small color domain size (such as twice
the maximum degree), this case does not occur very often (less than 18 percent in the worst
Thus, we can expect that in practice, our analytical result gives a very accurate lower bound on the stabilization time of the coloring protocol.

![Graph showing proportion of nodes creating a new conflict over the number of nodes choosing another color.](image)

Figure 4.2: Proportion of nodes creating a new conflict over the number of nodes choosing another color.

Tables 4.2 and 4.1 compare analytical and simulation results for the Synchronous scheduling in both modes, when using two different color domain sizes $|\Delta|$. As expected, theoretical results give tight lower bounds of the simulation outcome.

### 4.3 Stabilization time.

Figures 4.3 presents the stabilization time for each scheduling hypothesis and both modes when using different sizes of color domain, for a random spatial distribution of the nodes. The most striking results compared to those of [10] is that the stabilization time is much lower than expected. From the results of [10], the stabilization time is at least linear in $|\Delta|$ (being upper bounded by a constant when $\delta$ is also a constant). In contrast, our simulation results show a sub-linear (in $|\Delta|$) stabilization time, since considering $|\Delta| = 2 \times \max_{p \in V} |N_p|$ vs. $|\Delta| = \max_{p \in V} |N_p|^2$ merely divides by two the stabilization time. Also, doubling the degree of the nodes (when the network grows from 500 nodes to 1100 nodes) does not double the stabilization time. Especially, with the synchronous and probabilistic distributed schedulers, the stabilization time remains upper bounded by a constant.
4 neighbors

<table>
<thead>
<tr>
<th></th>
<th>all but one</th>
<th>all</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$2 \times \text{Max}^2$</td>
<td>$\text{Max}^2$</td>
</tr>
<tr>
<td>Theory</td>
<td>Simulation</td>
<td>Theory</td>
</tr>
<tr>
<td>1.88</td>
<td>1.89</td>
<td>1.51</td>
</tr>
</tbody>
</table>

8 neighbors

<table>
<thead>
<tr>
<th></th>
<th>all but one</th>
<th>all</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$2 \times \text{Max}^2$</td>
<td>$\text{Max}^2$</td>
</tr>
<tr>
<td>Theory</td>
<td>Simulation</td>
<td>Theory</td>
</tr>
<tr>
<td>1.51</td>
<td>1.56</td>
<td>1.14</td>
</tr>
</tbody>
</table>

Table 4.1: Theory and simulations results for the stabilization time for synchronous distance-1 coloring with $|\Delta| = (\max_{p \in V} |N_p|)^2$ and $|\Delta| = 2 \times (\max_{p \in V} |N_p|)$ in a grid.

<table>
<thead>
<tr>
<th>500 nodes</th>
<th>600 nodes</th>
<th>700 nodes</th>
<th>800 nodes</th>
<th>900 nodes</th>
<th>1000 nodes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean degree</td>
<td>15.7</td>
<td>18.8</td>
<td>22.0</td>
<td>25.1</td>
<td>28.3</td>
</tr>
<tr>
<td>Theory</td>
<td>2.35</td>
<td>2.40</td>
<td>2.44</td>
<td>2.50</td>
<td>2.53</td>
</tr>
<tr>
<td>Simulation</td>
<td>2.78</td>
<td>2.91</td>
<td>2.87</td>
<td>2.94</td>
<td>2.99</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>500 nodes</th>
<th>600 nodes</th>
<th>700 nodes</th>
<th>800 nodes</th>
<th>900 nodes</th>
<th>1000 nodes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Theory</td>
<td>2.83</td>
<td>2.91</td>
<td>2.94</td>
<td>2.95</td>
<td>3.05</td>
</tr>
<tr>
<td>Simulation</td>
<td>3.25</td>
<td>3.31</td>
<td>3.29</td>
<td>3.28</td>
<td>3.42</td>
</tr>
</tbody>
</table>

Table 4.2: Theory and simulations results for the stabilization time for synchronous distance-1 coloring with $|\Delta| = 2 \times (\max_{p \in V} |N_p|)$ random geometry topology.

In all cases, whatever the coloring distance and $\Delta$, we can note that the behavior of both probabilistic schedulers is similar. With the probabilistic scheduling hypothesis, in order to stabilize, the scheduler has to choose a node in conflict. In the all but one mode, only one node per pair of conflicting nodes actually chooses another color. The probabilistic schedulers thus have less chance to elect a conflicting node in the all but one mode than in the all mode (almost half chances less). Therefore, with the all mode, these probabilistic scheduling achieve better stabilization time (almost half time) than with the all mode. In the synchronous mode, at each step, every node acts. As in the all but one mode, in any pair of conflicting nodes, already one of those nodes has a stable color, so the stabilization time is lower than in the all mode. Note that for sensor networks, nodes are rarely tightly synchronized, so that the most realistic model is the distributed probabilistic scheduler, so the all mode is to be preferred in this context.

Results for distance-2 coloring are not plotted here but are similar to distance-1 coloring results in shape and lead to the same conclusions.
4.4 Influence of the size of the color domain.

Since the most realistic model for sensor networks is the distributed probabilistic one, Figures 4.4(a) and 4.4(b) plot the stabilization time for the distance-1 coloring algorithm with all mode as well as the height of the induced DAG, using different sizes of color domain. Nevertheless, results for the DAG height are similar whatever the coloring hypothesis.

Results clearly show that a higher domain size $|\Delta|$ induces a lower stabilization time and a higher DAG. There is thus a trade-off between these two characteristics depending of the application that will use the coloring. However, and although theoretical results show that the DAG height can be up to $\Delta - 1$, simulation results show that the actual height is in fact much lower, and most certainly sub-linear in $\Delta$.

Results for distance-2 coloring are not plotted here but are similar to distance-1 coloring results in shape and lead to the same conclusions.
Figure 4.3: Stabilization time of the distance-1 coloring process for different modes and scheduling hypothesis over a geometric node distribution with different sizes of color domain.
Figure 4.4: Influence of the color domain size on the stabilization time and the DAG height.
Distance-$k$ coloring is a useful mechanism for sensor networks. In [10], distance-3 coloring was used to construct a TDMA schedule, and in [12], distance-2 coloring permitted to expedite density based cluster construction. Further applications could be derived, e.g. distance-$k$ maximal independent set construction, by having nodes that have locally minimal color in their $k$ neighborhood be part of the independent set, and remaining nodes that do not see distance-$k$ neighbors (with lower color) in the independent set join the independent set.

In this paper, we show analytically and by simulation that the stabilization time of such coloring protocols is low. Nevertheless, we did not take into account the delay induced by the communication at distance-$k$ of the color of each node, reducing it to a given constant. Further studies are needed to get a more realistic bound that takes into account the probabilistic nature of the MAC layer of ad hoc and sensor networks. Indeed, as $k$ grows, the size of the messages get longer (in the order of $\delta^{k-1}$, since a node needs to communicate information about its neighborhood at distance $k - 1$ to each of its neighbors), so the probability that collisions occurs between neighboring grows, and the delay increases.
Bibliography


