Fast, Responsive Decentralised Graph Colouring

Alessandro Checco       Doug J. Leith

May 28, 2014

Abstract

We solve, in a fully decentralised way, the classic problem of colouring a graph. We propose a novel algorithm that is automatically responsive to topology changes, and we prove that it converges to a proper colouring in \( O(N \log N) \) time with high probability for generic graphs (and in \( O(\log N) \) time if \( \Delta = o(N) \)) when the number of available colours is greater than \( \Delta \), the maximum degree of the graph.

Moreover, this algorithm is significantly simpler and easier to implement than the state-of-the-art of stochastic learning algorithms: this gives some insight into which parts of such algorithms are essential to provide fast convergence.

We show how the proposed algorithm can be efficiently implemented in realistic industrial tasks, in particular a warehouse served by RFID robots, and in a smart electronic bookshelf, without the need to modify the RFID protocol or the readers, and retaining backward compatibility with standard RFID tags.

1 Introduction

Many fundamental wireless network allocation tasks can be formulated as a colouring problem, including channel and sub-carrier allocation [9], TDMA scheduling [1, 10], scrambling code allocation [5], network coding [9] and so on. Importantly, these tasks must often be solved while respecting strong communication constraints due, for example, to the range over which devices can communicate being smaller than the range over which they interfere or otherwise interact. Moreover, the network topology can change over
time, requiring the nodes to dynamically adapt to this. Recently, fully decentralised Communication-Free Learning (CFL) algorithms have been proposed for solving general constraint satisfaction problems without the need for message-passing \[9\] and with the ability to respond automatically to topology changes. These CFL algorithms exploit local sensing to infer satisfaction/dissatisfaction of constraints, thereby avoiding the need for message-passing and use stochastic learning to converge to a satisfying assignment. The extension to colouring problems with strong sensing restrictions is investigated in \[6\].

However, this class of algorithms, while quite simple, are nevertheless too complex to implement on highly resource constrained devices, such as Radio-Frequency Identification tags. Further, although very fast in simulations, these algorithms have not been proven to be fast: the only analytic bound available is exponential in the number of nodes.

In this paper we address both of these issues. We show that through a simple extension of the algorithm proposed in \[22\] (to make it responsive to topology changes) we can obtain results that are comparable, if not better, to the more complex Communication-Free Learning (CFL) algorithm in \[9\], with the advantage of being simple to implement (as explained in Section \[5.2\]) and to be provably fast (\(O(N \log N)\) when the number of available colours is greater than \(\Delta\) (the maximum degree of the graph), as shown in Section \[4.1\]). Since we obtain, by means of simulations, performance that is comparable to the CFL algorithm but using less memory and processing power (see Section \[5.4\]), this work also gives some insight into which parts of the CFL algorithm are essential to provide fast convergence. We illustrate the application of our algorithm in two Radio-Frequency Identification (RFID) applications: a warehouse served by RFID robots, and a smart electronic bookshelf. Importantly, there is no need to modify the RFID protocol or the readers, thus maintaining backward compatibility with standard RFID tags, and coexistence between multiple tag families.

1.1 Related Work

In the graph theory and computer science literature, the problem of colouring with \(\Delta + 1\) colours has been thoroughly studied \[12, 15, 19, 25\]. In particular, the family of *locally iterative algorithms* has received much attention. This family of algorithms makes use of the following strong assumptions:
1. the algorithm can use an unbounded number of colours (typically it will reduce them as the algorithm progresses);

2. the graph topology is assumed to be fixed;

3. each graph vertex needs to know which colours are not used by its neighbours.

Szegedy and Vishwanathan [25] use an heuristic argument to show that no locally iterative ($\Delta + 1$)-colouring algorithm is likely to terminate in less than $\Omega(\Delta \log \Delta)$ rounds (lower bound).

But in the wireless networking field, these assumptions may not be acceptable. To our knowledge, Assumption 1 has been discarded by all works in the wireless networking field, because it is inappropriate for such applications. Assumption 2 has been relaxed in two ways: by using network-wide stopping/restarting techniques in annealing-like algorithms [13], and by use of learning algorithms [2, 3, 9, 14, 15, 25].

Assumption 3 (either centralised or gossiping-like message passing) has been retained in [8, 13, 24, 27].

But even if, in certain conditions, communication between nodes may be possible, this cannot be relied upon in the design of a robust algorithm in cases where wireless nodes belong to different administrative domains or when the devices are too simple to be able to realize such communication (see, for example, RFID devices). The more challenging problem of graph colouring in which no message passing is possible (so all three assumptions are discarded) has only recently been studied in [2, 9, 22].

- The Learning-BEB algorithm, proposed by Barcelo et al. [2] is an algorithm devised for achieving collision-free scheduling in 802.11 networks. It is a modification of the CSMA/CA mechanism of truncated exponential backoff: after a successful transmission, the transmitter uses a fixed backoff interval $P$, while after a collision it selects an interval uniformly at random (u.a.r.) in the contention window range. Within the terminology of graph theory, this corresponds to a colouring algorithm in which each node selects the same colour after being locally satisfied, and selects a colour u.a.r. otherwise. This algorithm is known to suffer from slow convergence rates [10], but it has the advantage of being easy to implement.
• The algorithm proposed by Motskin et al. [22] is similar to [2], with the advantage of being provably fast ($O(\log N)$ when $\Delta = o(N)$) but with the major disadvantage of not being adaptive to topology changes, since after a correct local choice, the node keeps the chosen colour forever.

• The CFL algorithm proposed by Duffy et al. [9] uses a stochastic learning mechanism to update the probability of choosing each colour based on local sensing. In simulations it is fast, and it is provably adaptive to topology changes. The main disadvantage is that it is hard to prove good convergence rate bounds and it is too complicated to implement in simple hardware such as RFID tags.

It is worth noting that all three algorithms share the common property of initially selecting colours u.a.r. and staying with the same colour when locally satisfied. They also all belong to the family of locally iterative algorithm, even if they do not use assumptions 1-3. The difference between them lies in the way they respond to a loss of local satisfaction: Learning-BEB will go back to u.a.r. selection, the algorithm proposed by Motskin et al. [22] will keep the same choice even if locally unsatisfied, and CFL will distribute the probability mass amongst all colours, decreasing the probability of choosing the current unsatisfying colour. Learning-BEB is equivalent to CFL when the latter uses parameters $a = b = 1$ (in the terminology of [9]).

With our algorithm, we provide a simplified CFL algorithm (or an improved algorithm from [22], it depends on the point of view), that is provably fast (Section 4.1) and easy to implement (Section 5.2). Since we obtain very similar convergence rate performance to the CFL algorithm but using less memory and processing power (Section 5.4), this work also gives some insight into which part of the CFL algorithm is the essential part for ensuring fast convergence.

2 Problem Definition

We use the notation introduced in [9] for the more general decentralised constraint satisfaction problem, applying it to graph colouring problem as in [6].

Let $G = (\mathcal{N}, \mathcal{E})$ denote an undirected graph with set of vertices $\mathcal{N} = \{1, \ldots, N\}$ and set of edges $\mathcal{E} := \{(i, j) : i, j \in \mathcal{N}, i \leftrightarrow j\}$, where $i \leftrightarrow j$ denotes the existence of an edge between $i$ and $j$. A Colouring Problem (CP)
on graph $G$ with $D \in \mathbb{N}$ colours is defined as follows. Let $x_i \in \mathcal{D}$ denote the colour of vertex $i$, where $\mathcal{D} = \{1, \ldots, D\}$ is the set of available colours, and $\vec{x}$ denote the vector $(x_1, \ldots, x_N)$. Define clause $\Phi_m : \mathcal{D}^N \mapsto \{0, 1\}$ for each edge $m = (i, j) \in \mathcal{E}$ with:

$$
\Phi_m(\vec{x}) = \Phi_m(x_i, x_j) = \begin{cases} 
1 & \text{if } x_i \neq x_j \\
0 & \text{otherwise}.
\end{cases}
$$

We say clause $\Phi_m(\vec{x})$ is satisfied if $\Phi_m(\vec{x}) = 1$. An assignment $\vec{x}$ is said to be satisfying if for all edges $m \in \mathcal{E}$ we have $\Phi_m(\vec{x}) = 1$. That is

$$\vec{x} \text{ is a satisfying assignment iff } \min_{m \in \mathcal{E}} \Phi_m(\vec{x}) = 1. \quad (1)$$

Equivalently, $\vec{x}$ is a satisfying assignment if and only if $x_i \neq x_j$ for all edges $(i, j) \in \mathcal{E}$ i.e. if $i \leftrightarrow j$. For any colour allocation, we say a vertex is unsatisfied if at least one of its neighbours has the same colour; otherwise the vertex is said to be satisfied. A satisfying assignment for a colouring problem is also called a proper colouring.

**Definition 1** (Chromatic Number). The chromatic number $\chi(G)$ of graph $G$ is the smallest number of colours such that at least one proper colouring of $G$ exists.

We require the number of colours $D$ in our palette to be greater than or equal to $\chi(G)$ for a satisfying assignment to exist.

### 2.1 Decentralized CP Solvers

**Definition 2** (CP solver). Given a CP, a CP solver realizes a sequence of vectors $\{\vec{x}(t)\}$ such that for any CP that has a satisfying assignment

**\((D1)\)** for all $t$ sufficiently large $\vec{x}(t) = \vec{x}$ for some satisfying assignment $\vec{x}$;

**\((D2)\)** if $t'$ is the first entry in the sequence $\{\vec{x}(t)\}$ such that $\vec{x}(t')$ is a satisfying assignment, then $\vec{x}(t) = \vec{x}(t')$ for all $t > t'$.

In order to give criteria for classification of decentralized CP solvers, we re-write the LHS of Equation (1) to focus on the satisfaction of each variable

$$\vec{x} \text{ is a satisfying assignment iff } \min_{i \in \mathcal{V}} \min_{m \in \mathcal{E}_i} \Phi_m(\vec{x}) = 1. \quad (2)$$
where $\mathcal{E}_i$ consists of all edges in $\mathcal{E}$ that contain vertex $i$, i.e.

$$
\mathcal{E}_i = \{(j, i) : (j, i) \in \mathcal{E}\}.
$$

A decentralized CP solver is equivalent to a parallel solver, where each variable $x_i$ runs independently an instance of the solver, having only the information on whether all of the clauses that $x_i$ participates in are satisfied or at least one clause is unsatisfied. The solver located at variable $x_i$ must make its decisions only relying on this information.

**Definition 3** (Decentralized CP solver). A decentralized CP solver is a CP solver that for each variable $x_i$, must select its next value based only on the evaluation of

$$
\min_{m \in \mathcal{E}_i} \Phi_m(\vec{x}). \tag{3}
$$

That is, the decision is made without knowing

(D3) the assignment of $x_j$ for $j \neq i$.

(D4) the set of clauses that any variable, including itself, participates in, $\mathcal{E}_j$ for $j \in \mathcal{N}$.

(D5) the clauses $\Phi_m$ for $m \in \mathcal{E}$.

### 3 Proposed Algorithm

We consider the following decentralised algorithm, called Simplified Communication-Free Learning (SCFL):
Algorithm 1 Simplified Communication-Free Learning

1: Initialise vector $p = \frac{1}{D}1$ and counters $k = S, m = 0$
2: repeat
3: \hspace{1em} if $k = 0$ then
4: \hspace{2em} $k = S, m = 0$
5: \hspace{1em} end if
6: \hspace{1em} Select channel $c$ with probability $p_c$
7: \hspace{1em} if Satisfied OR $m = 1$ then
8: \hspace{2em} $p = \delta_c$ \hspace{1em} $\triangleright$ Choose same colour with probability 1
9: \hspace{2em} $m = 1$ \hspace{1em} $\triangleright$ Permanent state
10: \hspace{1em} else \hspace{1em} $\triangleright$ Unsatisfied
11: \hspace{2em} $p = \frac{1}{D}1$ \hspace{1em} $\triangleright$ Back to uniform selection
12: \hspace{1em} end if
13: \hspace{1em} $k = k - 1$ \hspace{1em} $\triangleright$ Decrease counter
14: until Forever

where $D$ is the number of colours available, and $S \in \mathbb{N}^+$ is a design parameter. The vertices have a common sense of time. For a round consisting of $S$ iterations, they select a colour u.a.r. until they become satisfied. At that point, they will enter what we call the permanent state ($m = 1$), i.e. they will not change their colour even if they become unsatisfied again. This permanent state lasts only until the round (of $S$ iterations) ends, then all vertices in the permanent state will start again to select colours u.a.r. if unsatisfied.

When all vertices are satisfied, the graph will have a proper colouring and keep it indefinitely. But as soon as the graph changes, for example upon the appearance of a new vertex, the vertices will start again to change colours after a delay of at most $S$ iterations, when they will go back to the non-permanent state\footnote{This differs from the satisfied and unsatisfied states because a permanent vertex is guaranteed to be satisfied only at the round in which it becomes permanent.}.

3.1 Role of Parameter $S$

When $S = 0$, the SCFL algorithm becomes that used in \cite{2}, while it is equivalent to that in \cite{22} when $S \to \infty$.\footnote{This differs from the satisfied and unsatisfied states because a permanent vertex is guaranteed to be satisfied only at the round in which it becomes permanent.}
Figure 1: Expected drift for unsatisfied vertices for SCFL algorithm when $S = 0$ on a complete graph.
Intuitively, parameter $S$ plays an important role in the performance of the algorithm. One the one hand, too small a value of $S$ will cause the vertices to be overly reactive to dissatisfaction, and change even when the graph is almost completely coloured, causing the system fluctuate around an allocation that is not a proper colouring. For example consider a complete graph with $D = N, N > 3$, in which $N - 2$ vertices are already coloured and only two vertices are unsatisfied. These two vertices will select their colour. Only in the unlikely event in which they choose the remaining two available colours will the system converge to a proper colouring. Otherwise the number of unsatisfied vertices will not decrease, making this algorithm slow to find a proper colouring.

This is illustrated in Figure 1 which shows the expected variation of unsatisfied vertices at next step (drift) vs. the number of unsatisfied vertices for different values of $N$. The drift here is computed exploiting the fact that when the graph is complete and $S = 0$, the corresponding Markov chain (where the state is the number of unsatisfied vertices) is easy to calculate. For $N > 3$, we can see a region (growing with $N$) of states with positive drift: the expected next state when inside this region is a state farther from the absorbing state.

On the other hand, it is important to keep $S$ small, because it also determines the maximum delay that can occur in the vertices reactivity to topology changes. This is because a proper coloured graph will have all vertices in the permanent state, and so it will require at least $S$ iterations to react to a change. Selecting $S \to \infty$, as in [22] would mean that the algorithm cannot react at all to topology changes.

In Section 4.1, we prove that it is enough to choose $S = \Delta + 1$ (where $\Delta$ is the maximum degree of the graph) to guarantee fast convergence.

3.2 Loose Bound for any Number of Colours

We can obtain similar bounds on the convergence time obtained in [9] by the CFL algorithm when the number of colours used is greater than or equal to the chromatic number $\chi$. Namely we have:

**Theorem 1.** Consider a feasible CP on a graph $G = \{N, E\}$, with palette $D$. Given any unsatisfied assignment $\vec{x}(0) \in D^N$, then with probability greater than $1 - \epsilon \in (0, 1)$, the number of iterations for SCFL algorithm to find a
satisfying assignment is less than

\[(S+1)N \exp\left(\frac{(S+1)N(N+1)}{2} \log(D)\right) \log(\epsilon^{-1}).\]

**Proof.** See Appendix \(\square\)

## 4 Fast Colouring – Performance Analysis

### 4.1 Main Result – Fast colouring with \(\Delta + 1\) colours

If at least \(\Delta + 1\) colours are available, SCFL is provably fast: it converges to a proper colouring in \(\mathcal{O}(N \log N)\) time with high probability for generic graphs, and in \(\mathcal{O}(\log N)\) time if \(\Delta = o(N)\). Moreover, this is achieved while keeping the parameter \(S\) small (of the order of \(\Delta + 1\)), allowing the algorithm to respond quickly to topology changes (see Section 3.1).

**Theorem 2.** Consider a CP on a graph \(G = \{N, \mathcal{E}\}\) with maximum degree \(\Delta\) and \(D > \Delta + 1\) available colours. Let \(|N| = N \geq 2\) and \(Z_t\) be the set of vertices in the non-permanent state at time \(t \in \{0, 1, 2, \ldots\}\), with \(|Z_t| = Z_t \in \{0, 1, 2, \ldots, N\}\). Let \(R\) be the first time in which all vertices reach the permanent state. For SCFL algorithm with \(S = \Delta + 1\) we have

\[\mathbb{P}(R \geq \log N + \log (\epsilon^{-1}) + K) \leq \epsilon,\]

for \(-1 < K \leq \log \frac{1}{1+\log 4}\). Given \(\epsilon\), we denote the bound \(B(N, \Delta, 1 - \epsilon) = \frac{\log N + \log (\epsilon^{-1}) + K}{\log \frac{\Delta+1}{\Delta} + K}\) when \(K = \log \frac{1}{1+\log 4}\).

**Corollary 1.** If \(\Delta = o(N)\), then for \(N \to \infty\), the order of steps before convergence is \(R = \mathcal{O}(\log N)\), or more precisely

\[\mathbb{P}(R \geq \log N + o(1)) \leq \epsilon, \quad \text{for } N \to \infty.\]

**Corollary 2 (Complete graphs).** If \(\Delta = \Theta(N)\), then for \(N \to \infty\), the order of steps before convergence is \(R = \mathcal{O}(N \log N)\), or more precisely

\[\mathbb{P}(R \geq N \log N + o(1)) \leq \epsilon, \quad \text{for } N \to \infty.\]

**Proof.** See Appendix. \(\square\)
4.2 Discussion

Theorem 2 means that if at least \( \Delta + 1 \) colours are available, SCFL algorithm achieves fast \( \mathcal{O}(N \log N) \) convergence to a proper colouring, while, from Theorem 1, when \( D \) is arbitrary we maintain a similar exponential bound to CFL. Simulations suggest that the latter bound is loose, but the drift analysis used to prove fast convergence when \( D \geq \Delta + 1 \) cannot be easily extended to the general case of an arbitrarily small number of colours. Szegedy and Vishwanathan [25] use an heuristic argument to show that no locally-iterative \((\Delta + 1)\)-colouring algorithms is likely to terminate in less than \( \Omega(\Delta \log \Delta) \) rounds, so it follows that the SCFL algorithm is order optimal in the case of complete graphs (when \( \Delta = N \)).

4.3 Differences with the State-of-the-art

As we will see by means of simulations in Section 4.4, the SCFL algorithm has very similar performance to the more complex CFL algorithm.

The main advantage of the SCFL algorithm over CFL is its simplicity: it does not require each vertex to update, for each iteration, a probability value over each colour. Its simplicity allows us to prove its fast convergence properties, and makes it more appealing for real world applications, see for example Section 5.2.

Moreover, the SCFL algorithm can obtain a similar convergence rate to CFL, but without using complicated stochastic learning techniques. This suggests that the main characteristic of the CFL algorithm that underpins its good performance is the "stickyness" to previously successful choices, rather than the ability to move the probability mass among the color vector.

A limitation of SCFL algorithm is that it requires a degree of synchronisation: vertices need to agree on when each round starts, so a global clock is needed. Note that we believe this assumption can be relaxed, and this is supported by simulation results. Unfortunately, relaxing it makes the proof of Theorem 2 considerably more involved, hence it is left for future work.

4.4 Simulations

In Figure 2, a comparison of the convergence time over 10 000 random graphs of the CFL algorithm, the Learning-BEB algorithm and SCFL is shown, when \( \Delta + 1 \) colours are available. Random graphs are created by selecting each
Figure 2: Comparison of the convergence time on random graphs for CFL algorithm, Learning-BEB algorithm and Algorithm 1 for densities of 0.8 and 0.4 respectively, using $\Delta + 1$ colours.
edge with probability 0.8 and 0.4 respectively. The number of iterations is plotted on a logarithmic scale.

It can be seen that, while Learning-BEB performs poorly (see Section 3.1 and Figure 1 for an explanation), CFL and SCFL exhibit a sub-polynomial convergence rate. From now on, the Learning-BEB algorithm results will not be shown anymore, because its consistently poor performance makes them uninteresting (and hard to compute) in the present context.

In Figure 7a, the empirical PDF (over 1000 samples) of the number of iterations to colour a 48 vertex complete graph are shown, for the CFL algorithm and the SCFL algorithm. The upper bound $B(N, \Delta, 0.9)$ on the number of iterations obtained using Theorem 2 with confidence $1 - \epsilon = 0.9$ is 2167 iterations.

In Figure 3, the empirical CDF of the convergence time for the CFL and SCFL algorithms is shown. Also the upper bound obtained using Theorem 2 is shown for a complete graph of 12 vertices. We can see that the two algorithms have similar performance and also that the bound seems to be
rather loose, even if we know from the heuristic argument of Szegedy and Vishwanathan [25] that no locally-iterative \((\Delta + 1)\)-colouring algorithm is likely to terminate in less than \(\Omega(\Delta \log \Delta)\) rounds, so the looseness of this bound is likely to be due to a pre-factor rather than the exponent.

To corroborate this intuition, in Figure 4 we show the ratio between the empirical estimate of the number of iterations and the bound \(B(N, \Delta, 1/2)\). We can see that the ratio tends to a constant value.

We obtain similar qualitative results for bipartite and random graphs.
5 Use Case – RFID robot/smart bookshelf

The SCFL algorithm can be applied in many different real life scenarios. We will show two examples:

**Warehouse with RFID robot** A RFID robot is a mobile reader, able to identify items in an environment (e.g. a warehouse) equipped with passive RFID tags [4, 7, 28]. The robot moves within the warehouse to locate and move items.

**Smart bookshelf with antenna array** A smart bookshelf is a device in which small items, like CDs or books, are managed in real time. An antenna array continuously reads RFID tags attached to the items using one antenna at a time, facilitating administrative tasks like renting or collection [16, 17, 21].

We will refer to both the robot and the antenna array as the *reader*. We are interested only in the RFID identification phase of the problem. In Figure 5 a schematic of both examples is shown, in which the reader (either a RFID robot or one antenna in the bookshelf array) is in the range of detection of 12 tags per time slot (these tags are indicated by a gray background when the robot is in the depicted position or the central antenna of the array is used).

Communication from the items to the reader can fail when there is a collision, i.e. when at least two RFID tags within the coverage of the reader transmit at the same time. To mitigate this, the RFID protocol implements a basic slotted Aloha collision resolution mechanism [11, 23, 26]. When the reader needs to identify a tag, it issues a QUERY command, and each tag in the coverage area selects an integer u.a.r. in interval \([0, D - 1]\), where \(D - 1\) is set by the reader. All tags that select 0 reply immediately; tags that select another number record those numbers in a counter and don’t transmit. A tag replies by sending a 16 bit random number. If the reader hears the random number, it echoes that number back as an acknowledgement, causing the tag to send its Electronic Product Code (EPC). The reader can then send commands specific to that tag, or continue to inventory other tags. In case of collision or the need for another identification, the reader can issue a QUERY REP command, causing all of the tags to decrement their counters by 1; again, any tag reaching a counter value of 0 will respond. After \(M\) steps, the procedure can start again with a QUERY command.
Figure 5: A schematic example of a grid of tags, in which the reader (either a RFID robot or an antenna in an array) is in the range of detection of 12 tags per time slot (tags with gray background when the reader is in central position).
Usually during inventory operation, the reader can set a flag (flag B) on successfully read tags, so they will not answer anymore to subsequent queries until a new command (set flag A) is broadcast to all tags.

**Problem Definition**

For both the warehouse and the smart bookshelf applications, we want to design a collision resolution mechanism that possesses the following properties:
(i) allow tags to be detected quickly (reading time comparable with Aloha);
(ii) allow subsequent read per tag to be faster; (iii) allow the reader to correctly read all of the tags when their relative positions change (for example when a batch is moved to a new warehouse); (iv) the new mechanism cannot require a change in the RFID protocol and has to be backward compatible, *i.e.* able to work with standard RFID tags and new tags together.

### 5.1 Collision-Free Scheduling

In the problem just defined, the RFID detection needs to be repeated in time, so a collision-free schedule would dramatically improve the efficiency of the medium access protocol. Moreover, the work of Melià-Seguí et al. [20] showed that the random number generator used in most RFID tags is biased towards certain values: the effect of the increase of collisions due to this bias would be heavily mitigated with a collision-free schedule.

The problem of assigning a different time slot (different counter value when the QUERY command is issued) to each RFID tag can be mapped to a CP on a graph, where the structure of the graph depends on the location of the tags. Graph $G = (\mathcal{N}, \mathcal{E})$ is built such that $\mathcal{N}$ is the set of tags, and an edge $e = (i, j) \in \mathcal{E}$ iff the tags $i$ and $j$ are near enough for their transmissions to potentially collide.

When all tags are within the coverage range of the reader, the problem is mapped to colouring of a complete graph. More generally (e.g. when the reader can cover at most $k$ tags per time), most RFID applications can be modeled as a CP on a complete $k$-partite graph $G_{s_1, \ldots, s_k}$, *i.e.* the graph composed of $k$ independent sets of (possibly different) size $s_i, i = 1, \ldots, k$, such that each set is connected with all the vertices of the other sets. This graph is $k$-colourable. For example, the collision-free schedule problem in the scenario represented in Figure 5 can be modeled as a CP on a 12-partite graph.
The SCFL algorithm is a natural candidate for this task, because

- It requires changing the behaviour of the tags only, without changing the RFID protocol or changing the (usually expensive) reader;
- It is backward compatible and can coexist with standard tags;
- It provides a significant speed-up, as shown in Section 5.3.

### 5.2 Implementation

We show how to implement SCFL algorithm in an existing RFID infrastructure ensuring backward compatibility.

The idea is to modify the behaviour of the tag, to allow it to enter the permanent state after successful QUERY, and to possibly exit it every \( S \) periods by extending the meaning of the \( \text{QueryAdjust} \) command. The \( \text{QueryAdjust} \) command is normally used to modify the range \([0, D-1]\) in the tags, to reduce the collision probability (by increasing the time period \( D \)) when many tags are present, or to reduce the expected backoff (by decreasing \( D \)) when few of them are present. The reader should be programmed to send a \( \text{QueryAdjust} \) command every \( S \) period, i.e. every \( S \cdot D \) queries.

Modified tags will thus have the following additional capabilities

1. If the reader sets flag B, the tag will enter the **permanent** state, and thus will select the same random number (same time slot) when flag A is broadcast again.

2. If the tag receives a \( \text{QueryAdjust} \) command and the tag flag is A, it will exit the permanent state.

We also assume the reader broadcasts flag A at the beginning of the process, and sets flag B on each tag that is correctly detected in a time slot not used by other tags. In this way already identified tags will not cause collisions, and tags that are correctly identified but that would cause a collision (with a previous identified tag) when a new inventory would be started will actually continue to change.

This implementation will still work together with non-modified tags at the expense of having some collisions, because those non-modified tags will choose a new (possibly different) time slot at every new QUERY, but each non modified tag can at most affect one modified tag, so the overall performance
should still be superior than original slotted Aloha mechanism. This intuition is confirmed by simulation in next section.

5.3 Comparison with Slotted Aloha

In this section the average time needed by the SCFL algorithm to identify correctly all tags in a complete graph is computed using simulations and compared with each of the following algorithms [18], which are based on slotted Aloha (with the additional capability of flagging a tag that has been correctly identified, so a flagged tag will not try to transmit anymore):

**BFSA** Basic Framed Slotted Aloha, with standard superframe size of 256 slots.

**DFSA** Dynamic Framed Slotted Aloha, where the superframe size doubles when the number of slots with collisions is larger than 70% of the current superframe size, and halves when the number of slots with collisions is less than 30%.

**EDFSA** Enhanced Dynamic Framed Slotted Aloha, see [18] for more details of this enhanced version of DFSA.

In this notation, the superframe size is equivalent to the parameter $D$, i.e. the number of slots after which the reader starts a new QUERY (forcing tags to select a new slot u.a.r.). These algorithms, different from SCFL, have the property of being memoryless, in the sense that for each superframe they behave statistically in the same way. On the other hand, the SCFL algorithm has a transient period in which a collision-free schedule is being determined, while after convergence the tags will deterministically select the same slot at every subsequent superframe. As shown in Table 1, the SCFL algorithm is comparable with classic slotted Aloha during the transient period, while at steady state performs better than classic slotted Aloha (83% reduction), and also better than the state-of-the-art dynamically adjusted slotted Aloha (66% reduction). Using the ISO15693 high tag data rate [26], the reader needs at each slot 1 ms to send the QUERY (or QUERY REP) command, and the tag needs 6 ms to complete the identification procedure with the reader (for transmission of the random number and reception of the echo acknowledgment). This would allow to read 1000 tags in around 7 seconds instead of the more than 40 seconds required for classic slotted Aloha. In the
Table 1: Median of the number of time slots needed to identify correctly all tags in a complete graph topology with $N = 200$ and $N = 1000$ for different algorithms. SCFL algorithm has a different reading time after convergence because it reaches collision-free operation, while the other algorithms are memoryless.

In the case of a tag grid in which 12 tags can interfere at each antenna, we simulated the time of convergence, time of first inventory and time of identification at steady state of the SCFL algorithm, compared to standard slotted Aloha with flagging enabled and superframe size equal to $\Delta + 1$ (as SCFL) varying the number of tags. In Figure 6, we can see that the time of first inventory is comparable for the two algorithms, but after convergence (less than 5 minutes for a shelf of 1000 items) the SCFL algorithm will be able to check the status of the whole library in 7 seconds, instead of the 32 seconds required for Aloha every time. In other words, a shelf with no new books will need 7 seconds to discover the status of the items (taken or not), while if new books are introduced or the topology of the grid changes, the SCFL algorithm will have the same performance as Aloha for at most 5 minutes, converging again to a collision-free schedule.

5.4 Memory and Computation Footprint vs. CFL

The implementation of the CFL would require the usage, in each tag, of a register of length $D$, to store the probability vector of choosing each slot at next QUERY. Moreover, it would be necessary to perform some basic operations (one multiplication, one division and one addition) at each QUERY.

Both implementation constraints are not required in the implementation of SCFL algorithm, while it keeps comparable, if not better, performance as illustrated in Figure 7, where the empirical PDF of the number of iterations
Figure 6: Reading time of SCFL algorithm and Aloha for a 12-partite complete graph. For SCFL algorithm time of convergence and time of reading at steady state are also shown.
to colour a graph are shown, in the case of a complete 12-partite graph with 10 vertices for each independent set over 10,000 samples for the CFL algorithm and the SCFL algorithm. The upper bound $B(N, \Delta, 0.9)$ on the number of iterations obtained using Theorem 2 with confidence $1 - \epsilon = 0.9$ is 784 iterations.

6 Conclusions and Future Work

The SCFL algorithm is responsive to topology changes, and converges to a proper colouring in $O(N \log N)$ time with high probability for generic graphs (and in $O(\log N)$ time if $\Delta = o(N)$) when the number of available colours is greater than $\Delta$.

The SCFL algorithm can be efficiently implemented in realistic industrial tasks, in particular in a warehouse served by RFID robots, and in a smart electronic bookshelf, without the need to modify the RFID protocol or the readers, and retaining backward compatibility with standard RFID tags.

The performance of the algorithm during its transient time is comparable with the standard slotted Aloha implementation, while the performance after convergence is one order of magnitude better. We note that the SCFL algorithm gives a clear advantage only to the systems in which the same set of tags is required to be read multiple times, otherwise an improved slotted Aloha algorithm may be preferable.

The main limitation of this work is the requirement for a form of central synchronisation, that we believe can be relaxed in a future work. This would make the algorithm suitable for more complex applications, such as 802.11 networks.

Acknowledgments

We want to thank Jaume Barcelo, Joan Meli-Segu and Marc Morenza for their thoughtful insights. Their expertise on RFID and Learning-BEB substantially improved this work.

This material is based upon works supported by Science Foundation Ireland under Grant No. 11/PI/1177.
Figure 7: Empirical PDF of the number of iterations to colour a complete graph and a complete 12-partite graph respectively, with 10 vertices for each independent set over 10,000 samples, for the CFL algorithm and the SCFL algorithm.
Appendix – Proofs

Consider graph $G = (\mathcal{N}, \mathcal{E})$. Let $A$ denote the set of assignments which are absorbing for SCFL algorithm, i.e. the set of proper colourings. All absorbing assignments are also satisfying. When the colouring problem is feasible (the number of colours available is greater than or equal to $\chi$) then $A \neq \emptyset$ (at least one satisfying assignment exists). Let $a \in A$ be a target satisfying assignment. We will refer to the assignment at time step $t$ as $\vec{x}(t)$.

Let $U_{\vec{x}(t)}$ denote the set of unsatisfied vertices and $\mathcal{D}$ the set of available colour. Define $\gamma = 1/D$.

**Lemma 1.** If a vertex is unsatisfied, when using SCFL algorithm the probability that the vertex chooses any colour $j$ at the next step is greater than or equal to $\gamma$.

**Proof.** This follows from step 11 of SCFL algorithm. □

**Proof of Theorem 1** Consider SCFL algorithm starting from an assignment $\vec{x}(0)$. Select an arbitrary valid solution $a \in A$. Since the CP is satisfiable, we have that $A \neq \emptyset$. We will exhibit a sequence of events that, regardless of the initial configuration, leads to a satisfying assignment with a probability for which we find a lower bound: SCFL algorithm will reach, in $(S+1)N$ steps, an assignment $\vec{x}((S+1)N)$ such that $U_{\vec{x}((S+1)N)} = \emptyset$ with probability greater than $\gamma^{(S+1)N(N+1)/2}$.

At the first step we consider the chain of events that changes the assignment, after $(S+1)$ steps, to

$$x_i((S+1)) = \begin{cases} a_i & \text{if } i \in U_{\vec{x}(0)}, \\ x_i(0) & \text{otherwise}. \end{cases} \quad (4)$$

This is feasible since SCFL algorithm ensures that all satisfied vertices at step 0 will remain unchanged for $(S+1)$ steps and each unsatisfied vertex may change its colour at step 1, and keep the same colour for $(S+1)$ steps with probability at least $\gamma^{(S+1)}$. The probability that this event happens is greater than $\gamma^{(S+1)|U_{\vec{x}(0)}|}$. Now, the set of unsatisfied variables $U_{\vec{x}((S+1))}$ could have changed. If $U_{\vec{x}((S+1))} = \emptyset$, we have finished, otherwise we consider again the event that changes the assignment similarly to equation (4), i.e. at generic step $\tau(S+1)$ we have

$$x_i(\tau(S+1)) = \begin{cases} a_i & \text{if } i \in U_{\vec{x}(\tau(S+1)-1)}, \\ x_i(\tau(S+1) - 1) & \text{otherwise}. \end{cases}$$
The probability of this happening is greater than $\gamma^{(S+1)\log\frac{(S+1)N(N+1)}{2}}$. The lower bound on the probability of this sequence is obtained when at each step that is a multiple of $(S + 1)$, only one new vertex choose the target colouring, giving us the bound of $(S + 1)N$ steps, with probability greater than $\gamma^{(S+1)1 \cdot \gamma^{(S+1)2} \cdot \gamma^{(S+1)N} = \gamma^{(S+1)N(N+1)/2}}$.

Due to the Markovian nature of SCFL algorithm and the independence of the probability of the above sequence on its initial conditions, if this sequence does not occur in $(S + 1)N$ iterations, it has the same probability of occurring in the next $(S + 1)N$ iterations. The probability of convergence in $k \cdot (S + 1)N$ steps is greater than $1 - \left(1 - \frac{\gamma^{(S+1)N(N+1)}}{2}\right)^k$. For

$$1 - \left(1 - \frac{\gamma^{(S+1)N(N+1)}}{2}\right)^k \geq 1 - \epsilon \text{ we require } k \leq \frac{\log\epsilon}{\log\frac{(S+1)N(N+1)}{2}} \leq \frac{\log\epsilon}{\gamma}.$$ 

Lemma 2. If vertex $i$ is in non-permanent state at the end of iteration $t$, then

$$\mathbb{P}(i \text{ becomes permanent at time } t+1) \geq \frac{M - \Delta}{M} \geq \frac{1}{\Delta + 1}.$$

Proof. A non-permanent vertex has at least $M - \Delta$ available colour, and its choice is uniform, so it has a probability at least equal to $\frac{M - \Delta}{M}$ to choose a colour not used by any neighbour. Now $\frac{\Delta}{M} \leq \frac{\Delta}{\Delta + 1}$, because $M \geq \Delta + 1$; so we have $\frac{M - \Delta}{M} = 1 - \frac{\Delta}{M} \geq 1 - \frac{\Delta}{\Delta + 1} = \frac{1}{\Delta + 1}$. \qed

Lemma 3. If all vertices are in permanent state, then they are all satisfied.

Proof. First let us note that in the first round in which a vertex becomes permanent, it is satisfied and it cannot cause dissatisfaction to its neighbours; the neighbours can still be unsatisfied, but only because of other vertices.

By contradiction, assume all vertices are in permanent state but there is at least one vertex $i$ unsatisfied. So there must be at least another neighbour $j$ unsatisfied and with same colour of $i$, by symmetry of dissatisfaction sensing. Now let us call $t_i, t_j$ the (last) time in which $i$ and $j$ became permanent, respectively. Assume, w.l.o.g. that $t_i < t_j$ (note that equality is not possible,
because at first round a vertex becomes permanent it is necessarily satisfied. Now at time $t_j$, $j$ became permanent, so it chose a colour different from $i$, causing a contradiction.

**Corollary 3.** A vertex in permanent state can be unsatisfied only by non-permanent neighbours.

**Lemma 4.** If a vertex $i$ is in permanent state and the counter $k$ is equal to zero, then

$$
\mathbb{P}(\text{vertex } i \text{ remains permanent}) = \left(1 - \frac{1}{M}\right)^{n(i,t)} \geq \left(\frac{\Delta}{\Delta + 1}\right)^{n(i,t)},
$$

where $n(i,t)$ is the number of neighbours of $i$ that are in non-permanent state at time $t$.

**Proof.** Let $x_i$ be the colour of vertex $i$. When the counter $k$ reaches zero, permanent vertex $i$ will still keep the same colour $x_i$. By Corollary 3 other permanent vertices cannot affect the satisfaction of vertex $i$, but $i$ could lose its (permanent) state if at least one of its non-permanent neighbours chooses $x_i$.

The probability that a non-permanent neighbour chooses a different colour from $x_i$ is $1 - \frac{1}{M}$, and since the choice of each vertex is independent, the probability all non-permanent vertices choose a colour from $x_i$ is $(1 - 1/M)^{n(i,t)}$.

Now, since $M \geq \Delta + 1$, we have $\frac{1}{M} \leq \frac{1}{\Delta + 1}$ and so $1 - \frac{1}{M} \geq 1 - \frac{1}{\Delta + 1} = \frac{\Delta}{\Delta + 1}$.

**Lemma 5.** Let $Z, N, \Delta$ positive integer numbers, with $N - Z \geq 1$, and $\Delta \geq 2$. The function

$$
f(Z) = (N - Z) \left(1 - \left(\frac{\Delta}{\Delta + 1}\right)^{\frac{\Delta Z}{N-Z}}\right)
$$

is concave with respect to $Z$.

**Proof.** This function is twice differentiable, and the second derivative is negative in its domain:

$$
f''(Z) = \frac{\Delta^2 \left(\log \left(\frac{\Delta}{\Delta + 1}\right)\right)^2 N^2}{\left(\frac{\Delta}{\Delta + 1}\right)^{\frac{\Delta Z}{N-Z}} (Z - N)^3}
$$

26
Lemma 6. For any choice of the integers $N > 2$, $1 \leq \Delta \leq N - 1$, $\tau \geq 1$, and the real $1 + \log 4 < k < e$ we have

$$
\left(\frac{\Delta}{\Delta+1}\right)^{\tau(\Delta+1)+1}k^{\tau-1}N + \left(1 - \left(\frac{\Delta}{\Delta+1}\right)^{\tau(\Delta+1)}k^{\tau-1} \right) \left( N - \left(\frac{\Delta}{\Delta+1}\right)^{(\Delta+1)k^{\tau-1}}N \right) \leq k^\tau N \left(\frac{\Delta}{\Delta+1}\right)^{\tau(\Delta+1)+1}.
$$

(5)

Proof. We first notice that we want $k < e$, to keep the bound limited when $\Delta = N - 1$ and $N \to \infty$ ($\lim_{N \to \infty} \left(\frac{N-1}{N}\right)^N = 1/e$). The function $X = \left(\frac{\Delta}{\Delta+1}\right)^{\tau(\Delta+1)}k^{\tau-1}$ is bounded from above by $1/e$ (because, for $k < e$, $X$ is increasing with $\Delta$ and decreasing with $\tau$, so we take $\Delta = N - 1$ and then taking the limit for $N \to \infty$ we get $1/e$ when $\tau = 1$) and from below by $0$. Now we consider function $Y = \frac{1-X}{X}$, decreasing with $X$ for $k < e$, and with image equal to $[e-1, \infty)$, and notice that (5) is equivalent to

$$
\frac{1}{k} F(\Delta, Y) \leq 1,
$$

with

$$
F(\Delta, Y) = \left( 1 + Y \frac{\Delta+1}{\Delta} \left( 1 - \left( \frac{\Delta}{\Delta+1} \right)^{\Delta/Y} \right) \right).
$$

Clearly $F$ is increasing with $Y$, so we can bound it from above with $F^*(\Delta) = \lim_{Y \to \infty} F(\Delta, Y) = 1 - \log \left( \left( \frac{\Delta}{\Delta+1} \right)^{\Delta+1} \right)$. It is easy to show that $F^*$ is decreasing with $\Delta$, so we just obtain the bound choosing $\Delta = 1$, that brings to $F(\Delta, Y) \leq F^*(1) = 1 + \log 4$. This guarantees that (5) is satisfied when $1 + \log 4 < k < e$. 

\[\square\]
Lemma 7. Let $\mathcal{N} \supseteq \mathcal{Z}$ two integer sets of cardinality $N$ and $Z$ respectively, and $N > 1$ and $Z \leq N$. Let $\Delta > 1$ be an integer and $n$ a integer vector of length $N$. If
\[ n(i) = 0, \quad \text{when } Z = \mathcal{N}, \]
and
\[ \sum_{i \in \mathcal{N} \setminus \mathcal{Z}} n(i) \leq \Delta Z, \] (6)
then the following holds
\[ f(Z, n) := \sum_{i \in \mathcal{N} \setminus \mathcal{Z}} \left( 1 - \left( \frac{\Delta}{\Delta + 1} \right)^{n(i)} \right) \leq \left( 1 - \left( \frac{\Delta}{\Delta + 1} \right)^{N - Z} \right)(N - Z). \] (7)

Proof. We maximise over $n$ the concave function $f(Z, n)$ subject to constraint (6). Since we want an upper bound, we can work on the relaxed problem in which we allow $n(i) \in \mathbb{R}$, because the maximum over this wider set will be greater than or equal to the maximum over $\mathbb{N}$. The optimisation is then convex. The Slater condition is satisfied, because $\Delta > 1$ and $Z \geq 1$ and so the point $n(i) = 0 \ \forall i$ is in the interior of the constraint set. Hence strong duality holds, and from the KKT conditions we obtain that at an optimum $n(i) = n(j)$ for all $i, j$ (because for each $i$ we get the very same condition $\nabla_{n(i)} f(Z, n) = 0 \Leftrightarrow \mu = -\log \left( \frac{\Delta}{\Delta + 1} \right)^{n(i)}$, where $\mu$ is the (unique) multiplier, because $1 - 1/\Delta \neq 0$), and from complementary slackness we obtain that constraint (6) is tight (because we get that $\mu = 0$ if constraint (6) is not tight, but this is not possible because $\frac{\Delta}{\Delta + 1}$ and $n(i)$ are finite and thus $\mu = 0$ contradicts first KKT condition). Hence, the $n(i)$ maximising $f$ is
\[ n(i) = \frac{\Delta Z}{N - Z}, \quad i \in \mathcal{N} \setminus \mathcal{Z}, \quad Z < N, \] (8)
and $n(i) = 0$ when $Z = N$ (by definition of $n(i)$).

Proof of Theorem \[.\] Let $Z_t$ the number of permanent vertices at time $t$. Observe that, by Lemma \[.\] $Z_t = 0$ is an absorbing state i.e. $Z_{\tau} = 0$ \ \forall $t \geq \tau$, and since $Z_t$ is non-negative we have that $E[Z_t] = 0$ implies $Z_t = 0$. It follows that $P(R \geq \tau \cdot S) = P(Z_{\tau} S \geq 1)$ and by Markov’s inequality,
\[ P(Z_{\tau} S \geq 1) \leq E[Z_{\tau} S]. \]
We divide the rest of the proof in two parts. First let us analyse the behaviour of the algorithm when time is not a multiple of $S$ and get a bound for the first $S - 1$ steps.

**First Part: $t < S$**

We define the random variable

$$X_i(t) = \begin{cases} 1, & \text{if vertex } i \text{ is permanent at time } t \\ 0, & \text{otherwise.} \end{cases}$$

Then $\mathbb{P}(i \text{ is permanent at time } t) = \mathbb{P}(X_i(t) = 1)$. We can define the random variable $M(t + 1)$ that represents the number of non-permanent vertices that become permanent at next step as

$$M(t + 1) = \sum_{i \in Z_t} X_i(t + 1).$$

So now we have

$$\mathbb{E}[Z_{t+1}|Z_t] = Z_t - \mathbb{E}[M(t + 1)] = Z_t - \sum_{i \in Z_t} \mathbb{E}[X_i(t + 1)] = Z_t - \sum_{i \in Z_t} 1 \cdot \mathbb{P}(i \text{ is permanent at time } t + 1), \quad (9)$$

and we can use Lemma 2 to obtain that in expectation, at least a fraction $\frac{\Delta}{\Delta + 1}$ of non-permanent vertices will become permanent at each step

$$\mathbb{E}[Z_{t+1}|Z_t] \leq \left(1 - \frac{1}{\Delta + 1}\right) Z_t, \quad t \neq S \mod S. \quad (10)$$

Since the RHS is strictly decreasing, for the first $S - 1$ steps we can bound it with

$$\mathbb{E}[Z_{t+1}|N] \leq \left(\frac{\Delta}{\Delta + 1}\right)^t N \quad t < S.$$  

**Second Part: $t > S$**

At time $\tau \cdot S + 1$, $\tau \geq 1$, we have to consider the number of vertices that exit from permanent state. By Lemma 4 and using the same construction used
in (9) and (10), the probability a vertex \( i \) remains permanent after a reset is:

\[
P(\text{no neighbours choose colour of } i) \geq \left(1 - \frac{1}{\Delta + 1}\right)^{n(i, t)},
\]

where \( n(i, t) \) is the number of neighbours of \( i \) that are in the non-permanent state, \( i.e. \) the number of edges that make \( i \) unsatisfied. So the expected number of vertices that exit from the permanent state, conditioned on \( Z_t \) is:

\[
e(Z_t) \leq \sum_{i \in \mathcal{N} \setminus Z_t} \left(1 - \left(\frac{\Delta}{\Delta + 1}\right)^{n(i, t)}\right) =: f(Z_t, n(\cdot, t)). \tag{11}
\]

Each non-permanent vertex can affect at most \( \Delta \) permanent vertices, and because of Corollary 3 a permanent vertex cannot affect any other permanent vertex, so the set \( \mathcal{N} \setminus Z_t \) can be affected by at most a number of edges equal to

\[
\sum_{i \in \mathcal{N} \setminus Z_t} n(i, t) \leq \Delta Z_t. \tag{12}
\]

To bound \( e(Z_t) \), we maximise over \( n(\cdot, t) \) the concave function \( f(Z_t, n(\cdot, t)) \) subject to constraint (12). Using Lemma 7 we get for any \( \tau > 1 \), combining (10), (7) and (8)

\[
\mathbb{E}[Z_{\tau S+1}|Z_{\tau S}] \leq (\frac{\Delta}{\Delta + 1})Z_{\tau S} + \left(1 - \left(\frac{\Delta}{\Delta + 1}\right)^{\frac{\Delta Z_{\tau S}}{N-Z_{\tau S}}}\right)(N - Z_{\tau S}), \text{ for } Z_{\tau S} > 0.
\]

Applying iterated expectation rule,

\[
\mathbb{E}[Z_{\tau S+1}] = \mathbb{E}\left[\mathbb{E}[Z_{\tau S+1}|Z_{\tau S}]\right] \leq (\frac{\Delta}{\Delta + 1})\mathbb{E}[Z_{\tau S}] + \mathbb{E}\left[\left(1 - \left(\frac{\Delta}{\Delta + 1}\right)^{\frac{\Delta Z_{\tau S}}{N-Z_{\tau S}}}\right)(N - Z_{\tau S})\right].
\]

We can now use Lemma 5 to apply Jensen’s inequality:

\[
\mathbb{E}[\phi(Z_t)] \leq \phi(\mathbb{E}[Z_t]),
\]

with \( \phi(Z_t) = (N - Z_t)\left(1 - \frac{\Delta}{\Delta + 1}\right)^{\frac{\Delta Z_{\tau S}}{N-Z_{\tau S}}} \). So we obtain

\[
\mathbb{E}[Z_{\tau S+1}] = \mathbb{E}\left[\mathbb{E}[Z_{\tau S+1}|Z_{\tau S}]\right] \leq (\frac{\Delta}{\Delta + 1})\mathbb{E}[Z_{\tau S}] + \left(1 - \left(\frac{\Delta}{\Delta + 1}\right)^{\frac{\Delta \mathbb{E}[Z_{\tau S}]}{N-\mathbb{E}[Z_{\tau S}]}}\right)(N - \mathbb{E}[Z_{\tau S}]). \tag{13}
\]
We set now, \( S = \Delta + 1 \), and we start considering the case \( \tau = 1 \). From (10), we have that \( \mathbb{E}[Z_{\Delta+1}] \leq \left( \frac{\Delta}{\Delta+1} \right)^{\Delta+1} N < \frac{N}{e} \). We want to bound the RHS of (13) substituting \( \mathbb{E}[Z_{\Delta+1}] \) with \( \left( \frac{\Delta}{\Delta+1} \right)^{\Delta+1} N \), but to do that we have to prove that the RHS of (13) is increasing with \( \mathbb{E}[Z_{\Delta+1}] \). The first addend of the RHS of (13) is affine and increasing with \( \mathbb{E}[Z_{\Delta+1}] \). Let us call second addend \( f \). For Lemma 5, \( f \) is strictly concave, so the subgradient property is:

\[
 f'(x) \leq f'(y) + (x - y) \partial f(y),
\]

and for \( x \in [0, N/e] \) and \( \partial f(N/e) \geq 0 \) we have \( f(x) \leq f(N/e) \). Thus it is enough to show that \( f' > 0 \) when \( \mathbb{E}[Z_{\Delta+1}] = \frac{N}{e} \). To simplify the analysis, we do the following strictly monotonic change of variable (that thus preserve the stationary points):

\[
 x = \frac{\mathbb{E}[Z_{\Delta+1}]}{N - \mathbb{E}[Z_{\Delta+1}]},
\]

\[
 f(x, N, \Delta) := \frac{N}{1 + x} \left( 1 - \left( \frac{\Delta}{\Delta + 1} \right)^{\Delta x} \right).
\]

Now the derivative of \( f \) computed in \( x(N/e) \) is positive iff

\[
 g(\Delta) = -e^{\Delta} \left( \frac{\Delta}{\Delta + 1} \right)^{\frac{\Delta}{\Delta + 1}} \log \left( \frac{\Delta}{\Delta + 1} \right) + e^{\Delta} \left( \frac{\Delta}{\Delta + 1} \right)^{\frac{\Delta}{\Delta + 1}} - \left( \frac{\Delta}{\Delta + 1} \right)^{\frac{\Delta}{\Delta + 1}} - e + 1 > 0.
\]

We can easily prove it splitting \( g \) in the sum of two functions

\[
 g_1(\Delta) = +e^{\Delta} \left( \frac{\Delta}{\Delta + 1} \right)^{\frac{\Delta}{\Delta + 1}} - \left( \frac{\Delta}{\Delta + 1} \right)^{\frac{\Delta}{\Delta + 1}} - e + 1 > 0
\]

\[
 g_2(\Delta) = -e^{\Delta} \left( \frac{\Delta}{\Delta + 1} \right)^{\frac{\Delta}{\Delta + 1}} \log \left( \frac{\Delta}{\Delta + 1} \right).
\]

- \( g_1 \) is decreasing so we bound it with \( \lim_{\Delta \to \infty} g_1 = -e^{-\frac{1}{\Delta}} \left( (e - 1) e^{-\frac{1}{\Delta}} - e + 1 \right) \),
- \( g_2 \) is increasing so we bound it with \( g_2(1) = \frac{e^{\log(2)}}{2^e} \).

So \( g \) is positive and thus \( f \) is increasing from 0 to \( N/e \), and we can then
bound it with the value in $N/e$:

$$
\mathbb{E}[Z_{\Delta+2}] \leq \left(\frac{\Delta}{\Delta+1}\right)^{\Delta+2} N + \left(1 - \left(\frac{\Delta}{\Delta+1}\right)^{\Delta+1} N\right)^{N} \left(N - \left(\frac{\Delta}{\Delta+1}\right)^{\Delta+1} N\right).
$$

(14)

We can now use Lemma 6 with $\tau = 1$ to bound the RHS of (14) with $kN(\Delta + 1)^{\Delta+1}$ (we note that this quantity is again smaller than $kN/e$). Applying again (10) for the next $N - 1$ steps, we get that $\mathbb{E}[Z_{2(\Delta+1)}] \leq kN(\Delta + 1)^{2(\Delta+1)}$, that is thus smaller than $N/e$ (because $k < e$). So we can use the same reasoning that brought from (13) to (14) and then apply Lemma 6 again with $\tau = 2$. Iterating this procedure we get, at any step $\tau(\Delta + 1)$,

$$
\mathbb{E}[Z_{\tau(\Delta+1)}] \leq k^{\tau-1} N(\Delta + 1)^{\tau(\Delta+1)}.
$$

Now, setting a target small probability $\epsilon$, we need to choose

$$
\tau \geq \frac{\log N + \log (\epsilon^{-1}) + \log (k^{-1})}{(\Delta + 1) \log \left(\frac{\Delta+1}{\Delta}\right) + \log (k^{-1})},
$$

to obtain the thesis. Corollary 1 and 2 are obtained taking the limit for $N \to \infty$ (and, regarding Corollary 2, $\frac{1}{N}$ is the first term of Taylor series at $\infty$ of $\log \left(\frac{\Delta}{\Delta+1}\right)$).

References

[1] J. Barcelo, B. Bellalta, C. Cano, A. Sfairopoulou, M. Oliver, and K. Verma. Towards a collision-free WLAN: Dynamic parameter adjustment in CSMA/E2CA. *EURASIP Journal on Wireless Communications and Networking*, 2011.

[2] Jaume Barcelo, Boris Bellalta, Cristina Cano, and Miquel Oliver. Learning-BEB: Avoiding collisions in WLAN. *Other IFIP Publications*, (1), 2011.
[3] Leonid Barenboim and Michael Elkin. Distributed $\Delta + 1$-coloring in linear in $\Delta$ time. In *Proceedings of the 41st annual ACM symposium on Theory of computing*, pages 111–120. ACM, 2009.

[4] Anna Carreras, Marc Morenza-Cinos, Rafael Pous, Joan Melià-Seguí, Kamruddin Nur, Joan Oliver, and Ramir De Porrata-Doria. STORE VIEW: pervasive RFID & indoor navigation based retail inventory management. In *Proceedings of the 2013 ACM conference on Pervasive and ubiquitous computing adjunct publication*, pages 1037–1042. ACM, 2013.

[5] A. Checco, R. Razavi, D. J. Leith, and H. Claussen. Self-configuration of scrambling codes for WCDMA small cell networks. In *IEEE 23rd International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, Sydney, Australia, September 2012.

[6] Alessandro Checco and Douglas J. Leith. Learning-based constraint satisfaction with sensing restrictions. *IEEE Journal of Selected Topics in Signal Processing*, 7:811–820, 2013.

[7] Harry KH Chow, King Lun Choy, WB Lee, and KC Lau. Design of a RFID case-based resource management system for warehouse operations. *Expert Systems with Applications*, 30(4):561–576, 2006.

[8] Jorge Crichigno, Min-You Wu, and Wei Shu. Protocols and architectures for channel assignment in wireless mesh networks. *Ad Hoc Networks*, 6 (7):1051–1077, 2008.

[9] K. R. Duffy, C. Bordenave, and D. J. Leith. Decentralized constraint satisfaction. *Networking, IEEE/ACM Transactions on*, 21(4):1298–1308, 2013.

[10] M. Fang, D. Malone, K. R. Duffy, and D. J. Leith. Decentralised learning MACs for collision-free access in WLANs. *Wireless Networks*, pages 1–16, 2010.

[11] K. Finkelzeller. *The RFID handbook*. John Wiley & Sons, 2003.

[12] Öjvind Johansson. Simple distributed $\Delta + 1$-coloring of graphs. *Information Processing Letters*, 70(5):229–232, 1999.
[13] Bruno Kauffmann, François Baccelli, Augustin Chaintreau, Konstantina Papagiannaki, Christophe Diot, et al. Self organization of interfering 802.11 wireless access networks. 2005.

[14] Kishore Kothapalli, M Onus, C Scheideler, and Christian Schindelhauer. Distributed coloring in $o(\sqrt{\log n})$–bits. In Proc. of IEEE International Parallel and Distributed Processing Symposium (IPDPS), 2006.

[15] Fabian Kuhn and Rogert Wattenhofer. On the complexity of distributed graph coloring. In Proceedings of the twenty-fifth annual ACM symposium on Principles of distributed computing, pages 7–15. ACM, 2006.

[16] Jun Lang and Liang Han. Design of library smart bookshelf based on RFID. Applied Mechanics and Materials, 519:1366–1372, 2014.

[17] Pui-Yi Lau, KK-O Yung, and Edward Kai-Ning Yung. A low-cost printed CP patch antenna for RFID smart bookshelf in library. Industrial Electronics, IEEE Transactions on, 57(5):1583–1589, 2010.

[18] Su-Ryun Lee, Sung-Don Joo, and Chae-Woo Lee. An enhanced dynamic framed slotted ALOHA algorithm for RFID tag identification. In Mobile and Ubiquitous Systems: Networking and Services, 2005. MobiQuitous 2005. The Second Annual International Conference on, pages 166–172. IEEE, 2005.

[19] Michael Luby. Removing randomness in parallel computation without a processor penalty. In Foundations of Computer Science, 1988., 29th Annual Symposium on, pages 162–173. IEEE, 1988.

[20] Joan Melià-Seguí, Joaquín García-Alfaro, and Jordi Herrera-Joancomartí. On the similarity of commercial EPC gen2 pseudorandom number generators. Transactions on Emerging Telecommunications Technologies, 2012.

[21] Joan Melià-Seguí, Rafael Pous, Anna Carreras, Marc Morenza-Cinos, Raúl Parada, Zeinab Liaghat, and Ramír De Porrata-Doria. Enhancing the shopping experience through RFID in an actual retail store. In Proceedings of the 2013 ACM conference on Pervasive and ubiquitous computing adjunct publication, pages 1029–1036. ACM, 2013.
[22] Arik Motskin, Tim Roughgarden, Primoz Skraba, and Leonidas J. Guibas. Lightweight coloring and desynchronization for networks. In INFOCOM, pages 2383–2391, 2009.

[23] Dong-Her Shih, Po-Ling Sun, David C. Yen, and Shi-Ming Huang. Taxonomy and survey of RFID anti-collision protocols. Computer communications, 29(11):2150–2166, 2006.

[24] Anand Prabhu Subramanian, Himanshu Gupta, Samir R Das, and Jing Cao. Minimum interference channel assignment in multiradio wireless mesh networks. Mobile Computing, IEEE Transactions on, 7(12):1459–1473, 2008.

[25] Mário Szegedy and Sundar Vishwanathan. Locality based graph coloring. In Proceedings of the twenty-fifth annual ACM symposium on Theory of computing, pages 201–207. ACM, 1993.

[26] Roy Want. An introduction to RFID technology. Pervasive Computing, IEEE, 5(1):25–33, 2006.

[27] Haitao Wu, Fan Yang, Kun Tan, Jie Chen, Qian Zhang, and Zhensheng Zhang. Distributed channel assignment and routing in multiradio multi-channel multihop wireless networks. Selected Areas in Communications, IEEE Journal on, 24(11):1972–1983, 2006.

[28] Bo Yan, Yiyun Chen, and Xiaosheng Meng. RFID technology applied in warehouse management system. In Computing, Communication, Control, and Management, 2008. CCCM’08. ISECS International Colloquium on, volume 3, pages 363–367. IEEE, 2008.