ALESSANDRO CHECCO

DECENTRALISED ALGORITHMS FOR WIRELESS NETWORKS

NUI MAYNOOTH

Ollscoil na hÉireann Má Nuad

DECENTRALISED ALGORITHMS FOR WIRELESS NETWORKS

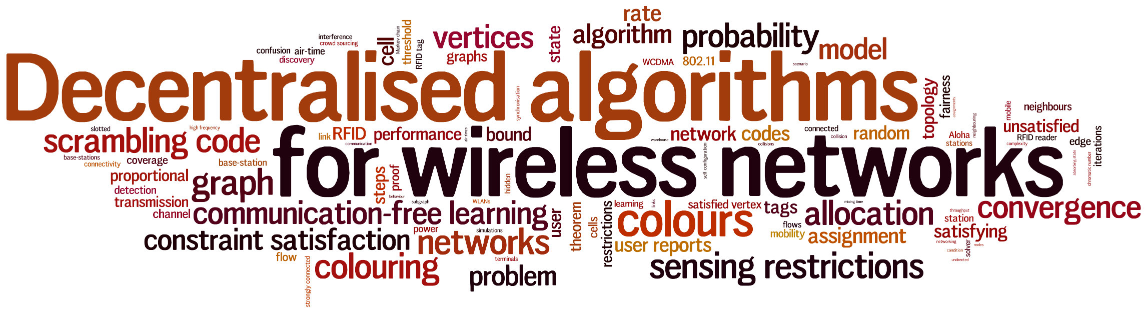
ALESSANDRO CHECCO

A DISSERTATION SUBMITTED FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

SUPERVISOR: PROF. DOUGLAS J. LEITH
HAMILTON INSTITUTE
NATIONAL UNIVERSITY OF IRELAND, MAYNOOTH

AUGUST 2014

email: alessandro.checco@nuim.ie, website: http://www.hamilton.ie/achecco/
Hamilton Institute, National University of Ireland, Maynooth



CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ALGORITHMS

ACRONYMS

WLAN	Wireless Local Area Network	1
CSMA/CA	Carrier Sense Multiple Access with Collision Avoidance	2
LTD	Local Topology Discovery	3
TD	Topology Discovery	4
FK	Full Knowledge	40
AP	Access Point	1
MC	Markov chain	42
MTU	Maximum Transmission Unit	2
TXOP	Transmission Opportunity	9
KKT	Karush–Kuhn–Tucker	11

DCS	Decentralised Constraint Satisfaction	62
PSC	Primary Scrambling Code	4
UE	User Equipment	24
UMTS	Universal Mobile Telecommunications System	
HSPA	High Speed Packet Access	
NCL	Neighbour Cell List	24
3GPP	3rd Generation Partnership Project	4
SGA	Single-step Greedy Algorithm	27
IGA	Iterative Greedy Algorithm	28
CFL	Communication-Free Learning	v
WCDMA	Wideband Code Division Multiple Access	24
RSCP	Received Signal Code Power	26
CPICH	Common Pilot Channel	26
WiSE	Wireless System Engineering	29
MIPS	Million Instruction Per Second	35
CP	Coloring Problem	v
TDMA	Time Division Multiple Access	
CFL	Communication-Free Learning	v
DBM	Directed Boolean Model	viii
WiGLE	Wireless Geographic Logging Engine	65
CP	Colouring Problem	v
RFID	Radio-Frequency Identification	6
EPC	Electronic Product Code	90
SCFL	Simplified Communication-Free Learning	82

1 INTRODUCTION

Designing and managing wireless networks is challenging for many reasons. Two of the most crucial in 802.11 wireless networks are: (a) variable per-user channel quality and (b) unplanned, ad-hoc deployment of the Access Points (APs). Regarding (a), a typical consequence is the selection, for each user, of a different bit-rate, based on the channel quality. This in turn causes the so-called performance "anomaly", where the users with lower bit-rate transmit for most of the time, causing the higher bit-rate users to receive less time for transmission (*air time*). Regarding (b), an important issue is managing interference. This can be mitigated by selecting different channels for neighbouring APs, but needs to be carried out in a decentralised way because often APs belong to different administrative domains, or communication between APs is unfeasible. Tools for managing unplanned deployment are also becoming important for other small cell networks, such as femtocell networks, where decentralised allocation of scrambling codes is a key task.

1.1 PERFORMANCE ANOMALY

One way to mitigate the performance anomaly is to artificially increase the air-time of high rate users and in this way to increase the *fairness* amongst users. While initially addressed in the literature in a heuristic way, the need has arisen for more soundly based approaches and a clear definition of fairness and air-time. It is notable that these approximate approaches in the literature share in common the idea that proportional fairness is related to some form of air-time fairness. We provide the first rigorous analysis of proportional fairness in 802.11 Wireless Local Area Networks (WLANs). For a WLAN (*i. e.* a single wireless hop) with no hidden terminals and noise losses we show that there exists a unique proportional fair rate allocation and completely characterise the allocation in terms of an air-time quantity. Importantly, we find that the correct air-time quantity, the *flow total air-time*, differs from the quantities previously considered in the literature. Our analysis is general enough to encompass both per station fairness and per flow fairness, and does not assume symmetric network load (stations may have different numbers of flows and these may have finite offered load, leading to stations being unsaturated).

1.1.1 *Performance Anomaly – Related Work*

Proportional fairness has been the subject of considerable attention in the literature on multi-rate 802.11 WLANs since it can be used to address the performance "anomaly" [Heusse et al., 2003, Herzen et al., 2011] in a principled manner and is closely related to so-called time-based fairness [Jiang and Liew, 2005, Tan and Guttag, 2004]. Well established utility fairness techniques from wired networks cannot, however, be directly applied to Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) wireless networks due to the presence of collision losses and the coupling of station transmissions via carrier sense.

For Aloha wireless networks, proportional fairness is rigorously analysed in [Liu et al., 2009, Kar et al., 2004, Wang and Kar, 2005]. In [Kar et al., 2004, Wang and Kar, 2005] it is established that there exists a unique proportional fair rate allocation and local message-passing algorithms that converge to this allocation are proposed. In [Liu et al., 2009] an alternative distributed algorithm based on back-pressure is studied. However, there are no corresponding rigorous results for 802.11 WLANs and the literature is confined to approximate approaches [Jiang and Liew, 2005, Siris and Stamatakis, 2006, Banchs et al., 2006, Dunn et al., 2005, Nandagopal et al., 2000]. The work in [Jiang and Liew, 2005, Nandagopal et al., 2000] neglects collision losses while [Banchs et al., 2006] neglects collisions involving more than two stations. In [Siris and Stamatakis, 2006] an algorithm based on contention window tuning is proposed and in [Dunn et al., 2005] an algorithm using Maximum Transmission Unit (MTU) tuning is proposed. Notwithstanding this work, since the 802.11 rate region is non-convex, basic issues such as the existence and uniqueness of proportional fair rate allocations remained open before the present work.

1.2 INTERFERENCE REDUCTION

To mitigate interference in unplanned, ad-hoc deployed networks, it is necessary to design simple, decentralised algorithms for channel and power selection that are robust and adaptive to changes in the network topology.

Recently, fully decentralised algorithms have been proposed for solving general constraint satisfaction problems without the need for message-passing [Duffy et al., 2013] and with the ability to respond automatically to topology changes. These 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. Our main contribution in this topic is the analysis and extension of such algorithms, for the class of constraint satisfaction problems corresponding to graph colouring.

We begin Chapter 3 by applying these techniques to scrambling code selection in femtocell networks (Section 3.2). This application motivates the Local Topology Discovery (LTD) problem, because femtocells need to rely on user reports in order to estimate their interference neighbourhood. In Section 3.3 we define the LTD problem, introduce a Markov chain model and use this to upper bound performance vs. the number of users in the network. In Chapter 4 we then investigate situations where algorithms cannot rely on symmetrical sensing, a very typical case in wireless networks where hidden terminals are present. We characterise a class of problems with partial sensing that can still be solved by CFL algorithms. Finally, in Chapter 5 we provide a significantly simpler and easier to implement algorithm that retains similar convergence rate to the state-of-the-art, but has the advantage of being provably fast when sufficient colours are available. This gives some insight into which characteristics of such algorithms are essential to provide fast convergence.

We also show how this simplified algorithm can be efficiently implemented in two applications, namely in a warehouse served by RFID robots and in a smart electronic bookshelf. In these applications the algorithm can be implemented without the need to modify the RFID protocol or the readers, and retaining backward compatibility with standard RFID tags.

1.2.1 *Interference Reduction – Related Work*

Self-configuration of scrambling codes for WCDMA small cell networks (3.2)

Reducing cell size is one of the simplest yet most effective solutions for capacity improvement *e.g.* in [Webb, 2007] the spectral gain efficiency of wireless systems during the years 1950 to 2000 is analysed and it is shown that shrinking of cell size has resulted in a factor of 2700 spectral efficiency gain. Self-configuration and self-optimisation are considered as a key enabler for successful deployment of small cells. This is especially true considering most deployments of small cells (*e.g.* femtocells) are expected to be ad-hoc and unplanned. The 3rd Generation Partnership Project (3GPP) standard considers the optimal selection of Primary Scrambling Codes (PSCs) as one of the top 5 most important parameters for self-configuration of small cells [3GPP TR 25.967, 2011].

Achieving a proper PSC allocation is even more important than interference reduction, because if two or more neighbouring femtocells use the same PSC, *code confusion* occurs and the network cannot correctly distinguish between them. This is potentially an extremely severe problem as it can prevent correct handover and make the cells unserviceable. Nevertheless, this thesis is the first work that addresses this problem for small cell networks.

Local Topology Discovery (3.3)

Topology Discovery (TD) has previously been investigated because of its applications to geographical position discovery [Chouldhury et al., 2000], routing protocols problems [Marwaha et al., 2002], and ad-hoc networks configuration in general [Choudhury et al., 2000]. However, to the best of our knowledge, the use of user reports from wireless handset for neighbour discovery has not been studied in the literature so far.

Partial Sensing (4)

The graph colouring problem has been the subject of a vast literature, from cellular networks (*e.g.* [Raniwala and Chiueh, 2005]), WLANs (*e.g.* [Raniwala and Chiueh, 2005, Mishra et al., 2006a,b, Leung and Kim, 2003, Narayanan, 2002] and references therein) and graph theory (*e.g.* [Dousse, 2012, Kothapalli et al., 2006, Hedetniemi et al., 2002, Johansson, 1999]). Almost all previous work has been concerned either with centralised schemes or with distributed schemes that employ extensive message-passing. Centralised and message-passing schemes have many inherent advantages. In certain situations, however, these systems may not be applicable. For example, differing administrative domains may be present in a network of WLANs. Early work in this direction includes that of Kauffmann et al. [2007], which proposes a distributed simulated annealing algorithm for joint channel selection and association control in 802.11 WLANs. However, heuristics are used to both terminate the algorithm and to restart it if the network topology changes. Network-wide stopping/restarting in a distributed context can be challenging without some form of message-passing. A similar approach is chosen in [Herzen et al., 2013a,b], where a distributed Metropolis sampler is provided to jointly frequency and bandwidth selection.

Fast Colouring with $\Delta + 1$ colours (5)

In the graph theory and computer science literature, the problem of colouring with $\Delta + 1$ colours has been thoroughly studied [Johansson, 1999, Kuhn and Wattenhofer, 2006, Szegedy and Vishwanathan, 1993, Luby, 1988]. 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 [1993] 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 (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 [Kauffmann et al., 2005], and by use of learning algorithms [Duffy et al., 2013, Kothapalli et al., 2006, Kuhn and Wattenhofer, 2006, Szegedy and Vishwanathan, 1993, Barcelo et al., 2011, Barenboim and Elkin, 2009].

Assumption 3 (either centralised or gossiping-like message passing) has been retained in [Kauffmann et al., 2005, Wu et al., 2006, Crichigno et al., 2008, Subramanian et al., 2008]. 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 realise 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 [Duffy et al., 2013, Barcelo et al., 2011, Motskin et al., 2009].

- The Learning-BEB algorithm, proposed by Barcelo et al. [2011] 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 [Fang et al., 2010], but it has the advantage of being easy to implement.

- The algorithm proposed by Motskin et al. [2009] is similar to [Barcelo et al., 2011], with the advantage of being provably fast ($\mathcal{O}(\log N)$ 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 Clifford and Leith [2007] and extended in [Duffy et al., 2013,?, Leith et al., 2012] 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 Radio-Frequency Identification (RFID) tags.

It is worth noting that all three algorithms share the common property of initially selecting colours u.a.r. and holding the colour assignment constant when locally satisfied. They also all belong to the family of locally iterative algorithms, even if they do not use assumptions 1-3. The difference between them lies in the way they respond to the loss of local satisfaction: Learning-BEB will go back to u.a.r.

selection, the algorithm proposed by Motskin et al. [2009] 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 [Duffy et al., 2013]).

1.3 PUBLICATIONS

The following papers have been published/submitted reporting the work in this thesis:

I. Alessandro Checco and Douglas J. Leith. Learning-Based Constraint Satisfaction With Sensing Restrictions. *Selected Topics in Signal Processing, IEEE Journal of*, 7(5):811–820, 2013.

II. Alessandro Checco and Douglas J. Leith. Proportional Fairness in 802.11 Wireless LANs. *Communications Letters, IEEE*, 15(8):807–809, 2011.

III. Alessandro Checco, Rouzbeh Razavi, Douglas J. Leith, and Holger Claussen. Self-configuration of Scrambling Codes for WCDMA Small Cell Networks. In *Personal Indoor and Mobile Radio Communications (PIMRC), 2012 IEEE 23rd International Symposium on*, pages 149–154. IEEE, 2012.

IV. Alessandro Checco and Douglas J. Leith. Fast, responsive decentralised graph colouring. *CoRR*, abs/1405.6987, 2014. Submitted to *IEEE/ACM Transactions on Networking*.

V. Alessandro Checco, Carlo Lancia, and Douglas J. Leith. Using crowd sourcing for local topology discovery in wireless networks. *CoRR*, abs/1401.1551, 2014. Submitted to *IEEE Transactions on Vehicular Technology*.

2 PROPORTIONAL FAIRNESS IN 802.11 NETWORKS

In this chapter we provide the first rigorous analysis of proportional fairness in 802.11 WLANs. This analysis corrects prior approximate studies. We show that there exists a unique proportional fair rate allocation and completely characterise the allocation in terms of a new air-time quantity, the *total air-time*.

CONTENTS

2.1 NETWORK MODEL

We begin by introducing our network model, which follows closely the well established model in [Duffy et al., 2005].

Station Throughput

We consider an 802.11 WLAN with n stations, $n \geqslant 2$. Let τ_i denote the probability that station i attempts a transmission. The throughput of station i is [Duffy et al., 2005]:

$$S_i(\tau) = \frac{P_{succ,i} D_i}{\sigma P_{idle} + T_c(1 - P_{idle})}$$

where $P_{idle} = \prod_{k=1}^{n}(1-\tau_k)$, $P_{succ,i} = \tau_i \prod_{k=1,k\neq i}^{n}(1-\tau_k)$, $\tau = [\tau_1 \ ... \ \tau_n]^T$, σ is the PHY idle slot duration, T_c the mean duration of a transmission, D_i the mean number of bits sent by station i in a successful transmission. This model is from [Leith et al., 2010], with the mean duration of successful and colliding transmissions taken equal *i. e.* without Transmission Opportunity (TXOP) packet bursting.

The WLAN *rate region* is the set $\mathcal{R}$ of achievable throughput vectors $S(\tau) = [S_1 \dots S_n]^T$ as the vector τ of attempt probabilities ranges over domain $[0,1]^n$. We also define the log-transformed rate region $\tilde{\mathcal{R}}$ as the set of achievable vectors $\tilde{S}(\tau) = [\tilde{S}_1 \dots \tilde{S}_n]^T$, $\tilde{S}_i = \log S_i$.

It will prove useful to work in terms of $x_i = \tau_i/(1-\tau_i)$ rather than τ_i; observe that $x_i \in [0,\infty)$ for $\tau_i \in [0,1)$. We have that $P_{idle} = 1/\prod_{k=1}^{n}(1+x_k)$, $P_{succ,i} = x_i P_{idle}$ and

$$S_i(x) = \frac{x_i}{X(x)} \frac{D_i}{T_c} \tag{1}$$

where $X(x) = a + \prod_{k=1}^{n}(1+x_k) - 1$ with $a = \sigma/T_c$.

Flows

We assign the packets transmitted by each station to "flows". Let $\mathcal{P}_i$ be the set of flows carried by station i and $\mathcal{P} = \cup_{i=1}^{n} \mathcal{P}_i$ the set of flows in the WLAN. How we choose to define a flow is essentially a design decision, subject only to the constraint that $\sum_{p \in \mathcal{P}_i} s(p) = S_i(x)$, $s(p)$ the rate of flow p. For example, we might define a flow to consist of packets with the same source-destination address/port number quadruple. Alternatively, per-station fairness is subsumed within our formulation by defining all packets transmitted by the same station to be a flow.

Air-time

We clarify *air-time*, since in the literature various non-equivalent definitions are used.

Definition 1 (Transmission Duration)**.** The time T_c taken to transmit a frame; *e.g.*, used in [Dunn et al., 2005].

Definition 2 (Successful Station air-time)**.** The fraction of time spent by a station on *successful* transmissions; *e.g.*, used in [Banchs et al., 2006]. For station i, this is given by $T_{succ,i} = x_i/X(x)$.

Definition 3 (Flow Total air-time)**.** The fraction of time $T(p)$ used for transmissions by flow p, including both successful transmissions and collisions. We also define the *station total air-time* $T_i = \sum_{p \in \mathcal{P}_i} T(p)$ and note that

$$T_i = \frac{\tau_i P_{coll,i} T_c + \tau_i (1 - P_{coll,i}) T_c}{\sigma P_{idle} + T_c (1 - P_{idle})} = \frac{x_i}{X(x)} \left(1 + \frac{P_{coll,i}}{1 - P_{coll,i}} \right)$$

where $P_{coll,i} = 1 - \frac{\prod_{k=1}^{n} (1-\tau_k)}{1-\tau_i} = 1 - \frac{1+x_i}{\prod_{k=1}^{n} (1+x_k)}$ is the collision probability experienced by station i.

2.2 PROPORTIONAL FAIR RATE ALLOCATION

Assume that the WLAN offered load is unconstrained. This is similar to a saturated station (*i. e.* stations always have a packet to send) assumption, which we will later relax. To determine the proportional throughput allocation we need to solve the following utility optimisation problem:

$$\max_{x,s} \sum_{p \in \mathcal{P}} \log s(p) \text{ s.t.} \sum_{p \in \mathcal{P}_i} s(p) \leqslant \frac{x_i}{X(x)} \frac{D_i}{T_c},\; x_i \geqslant 0,\; i = 1, ..., n$$

The constraints ensure that the aggregate flow throughput at station i is feasible and so lies within the WLAN rate region. Since the WLAN rate region $\mathcal{R}$ is non-convex, the optimisation problem is non-convex. Fortunately, it has recently been shown that the log-transformed rate region $\tilde{\mathcal{R}}$ is, however, convex [Leith et al., 2010]. Changing variables to $\tilde{s}(p) = \log s(p)$, the optimisation can therefore be transformed into convex form:

Problem 2.1 (Proportional Fairness)**.**

$$\max_{x,s} \sum_{p \in \mathcal{P}} \tilde{s}(p) \text{ s.t.} \log \sum_{p \in \mathcal{P}_i} e^{\tilde{s}(p)} \leqslant \log \frac{x_i D_i}{X(x)\, T_c},\; x_i \geqslant 0,\; i = 1, .., n$$

In addition to the constraints now being convex, there also exists a strictly feasible point and so the Slater conditions [Rockafellar, 1997] are satisfied. Hence, strong duality holds and the Karush–Kuhn–Tucker (KKT) conditions [Rockafellar, 1997] are necessary and sufficient conditions for global optimality. However, since the objective function is not strictly concave (it is linear in $\tilde{s}(p)$), extra work is needed to establish that the optimisation has a unique solution. We will make use of the following Lemma:

Lemma 2.1. *The log-transformed rate region $\tilde{R}$ is strictly convex.*

Proof. Let $\partial\tilde{\mathcal{R}}$ denote the boundary of set $\tilde{\mathcal{R}}$. Since we already know that $\tilde{\mathcal{R}}$ is convex, to establish strict convexity it is sufficient to show that the tangent hyperplanes to any two boundary points $\tilde{S}(x^1), \tilde{S}(x^2) \in \partial\tilde{\mathcal{R}}$ are different whenever $x^1 \neq x^2$. Now $\tilde{S}_i(x) = \log \frac{x_i}{X(x)} \frac{D_i}{T_c}$ and

$$\frac{\partial \tilde{S}_i}{\partial x_j} = \frac{\delta_{ij}}{x_i} - \frac{1}{X} \prod_{\substack{k \neq j \\ k=1}}^{n} (1 + x_k),$$

where $\delta_{ij} = 1$ when $i = j$ and 0 otherwise. The normal vector $b(x)$ to the tangent hyperplane at point $\tilde{S}(x) \in \partial\tilde{\mathcal{R}}$ solves

$$\sum_{i=1}^{n} b_i(x) \frac{\partial \tilde{S}_i(x)}{\partial x_j} = 0 \;\forall j = 1, ..., n.$$

It can be verified (see appendix to this chapter) that $b_j(x) = \frac{x_j}{1+x_j}$ is one such normal vector. Hence, for any two points $\tilde{S}(x^1)$, $\tilde{S}(x^2)$ on the rate region boundary with $x^1 \neq x^2$, the corresponding tangent hyperplanes are different, as required. $\square$

We are now in a position to state the first of our main results.

Theorem 2.1. *There exists a unique proportional fair rate allocation, the solution to Problem 2.1, which: (i) assigns equal flow total air-times ($T(p) = T(q) \; \forall \; p, q \in \mathcal{P}$), (ii) the flow total air-times sum to unity ($\sum_{p \in \mathcal{P}} T(p) = 1$) and (iii) lies on the boundary of the WLAN rate region. Note that since the flow air-time usage overlaps due to collisions, (ii) does* not *imply that the channel idle probability $P_{idle} = 0$.*

Proof. We proceed by making use of strong duality. The Lagrangian is

$$L(x, s, \lambda) = \sum_{p \in \mathcal{P}} \tilde{s}(p) + \sum_{i=1}^{n} \lambda_i \left(\log \frac{x_i D_i}{X T_c} - \log \sum_{p \in \mathcal{P}_i} e^{\tilde{s}(p)} \right)$$

The main KKT conditions are

$$1 - \lambda_i \frac{e^{\tilde{s}(p)}}{\sum\limits_{q \in \mathcal{P}_i} e^{\tilde{s}(q)}} = 0 \qquad \forall \, p \in \mathcal{P}, \; i = r(p) \tag{2}$$

$$\frac{\lambda_i}{x_i} - \frac{1}{X} \frac{\partial X}{\partial x_i} \sum_{j=1}^{n} \lambda_j = 0, \qquad i = 1, \ldots n \tag{3}$$

From the first KKT condition (2), $s(p) = \sum_{q \in \mathcal{P}_i} s(q)/\lambda_i \ \forall\, p \in \mathcal{P}_i$ and so all flows carried by the same station are allocated the same throughput. Summing (2) over flows $p \in \mathcal{P}_i$ carried by station i yields

$$\lambda_i = n_i \tag{4}$$

where $n_i = |\mathcal{P}_i|$ is the number of flows carried by station i. It follows from complementary slackness that since $\lambda_i > 0$ the station throughput constraint in Problem 2.1 is tight. Thus station throughput $S_i(x) = n_i s(p)$ and $\tilde{S}_i(x) = \log(n_i s(p))$. The contribution to the optimisation objective function in Problem 2.1 by station i is $n_i \log s(p) = n_i \tilde{S}_i - n_i \log n_i$. Hence, the level sets form hyperplanes in rate region $\tilde{R}$. Any optimal solution $\tilde{S}^*$ must lie on the boundary of $\tilde{R}$, else the flow throughputs could be increased and the objective improved. Since $\tilde{R}$ is strictly convex by Lemma 2.1, each boundary point has a unique supporting hyperplane and so the optimum of Problem 2.1 is unique, as claimed.

Turning now to the second KKT condition, rearranging (3) and using (4) yields $T_i = \frac{\lambda_i}{\sum_{j=1}^{n} \lambda_j}$, where T_i is the total air-time used by station i. The total air-time used by each flow $p \in \mathcal{P}_i$ carried by station i is

$$T(p) = \frac{T_i}{n_i} = \frac{1}{\sum_{j=1}^{n} \lambda_j},$$

establishing the property (i) as claimed. Property (ii) follows from the fact that $\sum_{j=1}^{n} \lambda_j = \sum_{j=1}^{n} |\mathcal{P}_j| = |\mathcal{P}|$. $\square$

2.3 FINITE-LOAD

The foregoing analysis can be generalised to situations with finite offered-load. Problem 2.1 is then augmented with the additional (convex) constraint

$$\tilde{s}(p) \leqslant \log \bar{s}(p), \qquad \forall p \in \mathcal{P} \tag{5}$$

where $\bar{s}(p) > 0$ is the offered load of flow p. Let $\mathcal{S} = \{r \in \mathcal{P} \colon s(r) = \bar{s}(r)\}$ denote the set of offered-load constrained flows and $\mathcal{U} = \mathcal{P} \setminus \mathcal{S}$ the set of network constrained flows. We have the following intuitive extension of Theorem 2.1 demonstrating that the air-time left unused by the offered-load constrained flows is simply reallocated equally amongst the network constrained flows:

Theorem 2.2. *The proportional fair rate allocation with offered-load constraint (5): (i) assigns equal total air-time to all network constrained flows ($T(p) = T(q)\ \forall\ p,q \in \mathcal{U}$), (ii) the flow total air-times sum to unity ($\sum_{p\in\mathcal{P}} T(p) = 1$) and (iii) is on the boundary of its rate region.*

Proof. Strong duality also holds for the augmented problem. The Lagrangian is

$$L(x,s,\lambda,\vartheta) = \sum_{p\in\mathcal{P}} \tilde{s}(p) + \sum_{i=1}^{n} \lambda_i \left(\log \frac{x_i D_i}{X(x)T_c} - \log \sum_{p\in\mathcal{P}_i} e^{\tilde{s}(p)} \right) + \sum_{p\in\mathcal{P}} \vartheta_p (\tilde{\bar{s}}(p) - \tilde{s}(p))$$

with multiplier $\vartheta_p \geqslant 0\ \forall p \in \mathcal{P}$. The second KKT condition (3) is unchanged but the first KKT condition becomes

$$1 - \lambda_i \frac{e^{\tilde{s}(p)}}{\sum_{q\in\mathcal{P}_i} e^{\tilde{s}(q)}} - \vartheta_p = 0 \qquad \forall\ p \in \mathcal{P},\ i = r(p) \tag{6}$$

Let $\mathcal{S}_i = \mathcal{P}_i \cap \mathcal{S}$ be the set of offered-load constrained flows carried by station i and $\mathcal{U}_i = \mathcal{P}_i \setminus \mathcal{S}_i$ the set of network constrained flows. For a network constrained flow $p \in \mathcal{U}_i$ multiplier $\vartheta_p = 0$ by complementary slackness. Consider a station i with at least one network constrained flow (if there are no network constrained flows, the throughput allocation is trivial). By (6), $s(p) = \sum_{q\in\mathcal{P}_i} s(q)/\lambda_i\ \forall\ p \in \mathcal{U}_i$ (so all network constrained flows on station i have the same throughput) and

$$s(r) = \bar{s}(r) = (1-\vartheta_r)s(p) \qquad \forall\ r \in \mathcal{S}_i, p \in \mathcal{U}_i$$

The multiplier ϑ_r is equal to the relative throughput difference between offered-load constrained flow r and a network constrained flow. Combining these observations we have that

$$\lambda_i = \sum_{p\in\mathcal{U}_i} 1 + \sum_{r\in\mathcal{S}_i} (1-\vartheta_r) \tag{7}$$

Now from the second KKT condition, $T_i = \frac{\lambda_i}{\sum_{j=1}^{n} \lambda_j}$. Taking this together with (7), the total air-time allocated to flow $p \in \mathcal{P}$ is

$$T(p) = \begin{cases} \frac{1}{\sum_{j=1}^{n} \lambda_j} & p \in \mathcal{U} = \mathcal{P} \setminus \mathcal{S} \\ \frac{1-\vartheta_p}{\sum_{j=1}^{n} \lambda_j} & p \in \mathcal{S} \end{cases}$$

That is, every network constrained flow $p \in \mathcal{U}$ is allocated the same total air-time as claimed. Property (ii) follows from inspection of $\sum_{j=1}^{n} \lambda_j$. Operation at the rate region boundary is necessary as otherwise the flow throughputs could be increased and so the objective improved. □

2.4 EXAMPLE

Consider a WLAN with 10 stations and 24 UDP flows. The WLAN uses 802.11g MAC/PHY values (9 μ s slot time, short preamble, 6 Mbps PHY rate). The first three stations carry 2, 5 and 10 flows respectively. The remaining 7 stations carry one flow each. The offered load is not constrained, so Theorem 2.1 applies. Figure 1 shows the flow total air-times and flow success air-times for the proportional fair rate allocation. It can be seen that the proportional fair allocation assigns equal flow total air-times but that success air-times are unequal. In general, equalising flow total air-times is not equivalent to equalising flow success air-times whenever there is asymmetry in the network load (*e. g.* when stations do not all carry an identical number of flows) since this leads to stations experiencing different collision probabilities.

2.5 CONCLUSIONS

We provide the first rigorous analysis of proportional fairness in 802.11 WLANs. We show that a unique proportional fair rate allocation exists and, correcting previous studies, that this allocation assigns equal *total air-time* to flows. Total air-time is the time spent on both colliding and successful transmissions and differs from other air-time quantities proposed heuristically in the literature.

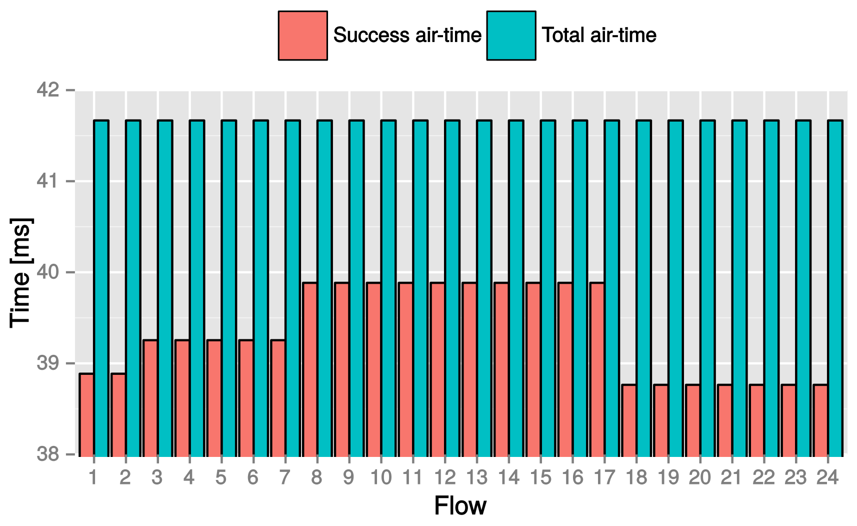


Figure 1: Flow air-times of the proportional fair rate allocation in a 802.11g WLAN with 24 UDP flows and 10 stations.

2.A APPENDIX - NORMAL VECTOR

We want to verify that $b_i = \frac{x_i}{1+x_i}$ is a normal vector on the boundary of the log-tranformed rate region. In other words, given

$$\frac{\partial \tilde{S}_i}{\partial x_j} = \frac{\delta_{ij}}{x_i} - \frac{1}{X} \prod_{\substack{k \neq j \\ k=1}}^{n} (1 + x_k),$$

we want to show that on the rate region boundary the following holds

$$\sum_{i=1}^{n} b_i(x) \frac{\partial \tilde{S}_i(x)}{\partial x_j} = 0 \; \forall j = 1, ..., n.$$

On the rate region boundary we have [Subramanian, 2012]

$$X = \prod_m (1 + x_m) \sum_i \frac{x_i}{1 + x_i}.$$

So we can write

$$\sum_i \left(\frac{x_i}{1+x_i} \left(\frac{\delta_{ij}}{x_i} - \frac{\prod_{k \neq j}(1+x_k)}{\sum_k \frac{x_k}{1+x_k} \prod_k (1+x_k)} \right) \right) =$$
$$\sum_i \left(\frac{x_i}{1+x_i} \left(\frac{\delta_{ij}}{x_i} - \frac{1}{(1+x_j) \sum_k \frac{x_k}{1+x_k}} \right) \right) =$$
$$\frac{x_j}{x_j(1+x_j)} - \frac{\sum_i \frac{x_i}{1+x_i}}{(1+x_j) \sum_k \frac{x_k}{1+x_k}} = 0.$$

as required. □

3 DECENTRALISED ALGORITHMS FOR SMALL CELL NETWORKS

In Section 3.1, we introduce the decentralised Colouring Problem (CP), and define the class of solvers we want to study. In Section 3.2, we show how this problem arises in small cell networks for channel/code allocation, and propose a decentralised solver for it. In Section 3.3, we show how crowd sourcing can be used for local topology discovery, to facilitate the design of such solvers.

CONTENTS

3.1 DECENTRALISED COLOURING ALGORITHMS

We begin by recalling the formal definition of a decentralised Colouring Problem (CP) solver, an example of which is the CFL algorithm introduced by [Leith et al., 2012].

3.1.1 *Colouring Problem (CP)*

Let $G = (V, \mathcal{M})$ denote an undirected graph with set of vertices $V = \{1, \ldots, N\}$ and set of edges $\mathcal{M} := \{(i, j) : i, j \in V, i \leftrightarrow j\}$, where $i \leftrightarrow j$ denotes the existence of a pair of directed edges $i \rightarrow j$, $i \leftarrow j$ joining vertices $i, j \in V$. Note that with this notation the edges in set $\mathcal{M}$ are directed, since this will prove convenient later when considering oriented subgraphs of G. However, since graph G is undirected we have $(i, j) \in \mathcal{M} \iff (j, i) \in \mathcal{M}$.

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 $\Phi_m \colon \mathcal{D}^N \mapsto \{0, 1\}$ for each edge $m = (i, j) \in \mathcal{M}$ 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 $m \in \mathcal{M}$ we have $\Phi_m(\vec{x}) = 1$. That is

$$\vec{x} \text{ is a satisfying assignment iff } \min_{m \in \mathcal{M}} \Phi_m(\vec{x}) = 1. \tag{8}$$

Equivalently, $\vec{x}$ is a satisfying assignment if and only if $x_i \neq x_j$ for all edges $(i, j) \in \mathcal{M}$ *i.e.* if $i \leftrightarrow j$. A satisfying assignment for a colouring problem is also called a *proper colouring*.

Definition 4 (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. That is, we require the number of colours D in our palette to be greater or equal than $\chi(G)$ for a satisfying assignment to exist.

3.1.2 *Decentralised CP Solvers*

Definition 5 (CP solver). Given a CP, a CP solver realises a sequence of vectors $\{\vec{x}(t)\}$ such that, if a satisfying assignment exists,

(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 decentralised CP solvers, we re-write the LHS of Equation (8) to focus on the satisfaction of each variable

$$\vec{x} \text{ is a satisfying assignment iff } \min_{i \in V} \min_{m \in \mathcal{M}_i} \Phi_m(\vec{x}) = 1. \tag{9}$$

where $\mathcal{M}_i$ consists of all edges in $\mathcal{M}$ that contain vertex i, *i. e.*

$$\mathcal{M}_i = \big\{(j,i) \colon (j,i) \in \mathcal{M}\big\}.$$

Note that we adopt the convention of including edges in $\mathcal{M}_i$ which are incoming to vertex i, but since $(i,j) \in \mathcal{M} \iff (j,i) \in \mathcal{M}$ then $\bigcup_{i \in V} \mathcal{M}_i = \mathcal{M}$.

A decentralised 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 6 (Decentralised CP solver)**.** A decentralised CP solver is a CP solver that for each variable x_i, must select its next value based only on the evaluation of

$$\min_{m \in \mathcal{M}_i} \Phi_m(\vec{x}). \tag{10}$$

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, $\mathcal{M}_j$ for $j \in V$.

(D5) the clauses Φ_m for $m \in \mathcal{M}$.

3.1.3 *Communication-Free Learning (CFL) Algorithm*

Pseudo-code for the CFL algorithm, that is the decentralised CP solver we will analyse and extend for the rest of this thesis, is given in Algorithm 1.

Algorithm 1 Communication-Free Learning for base-station i

```
1: Initialize $p_{i,j} = 1/D, j \in \{1, \dots, D\}$.
2: Realise a random variable, selecting $x_i = j$ with probability $p_{i,j}$.
3: loop
4:     Evaluate $\min_{m \in \mathcal{M}_i} \Phi_m(\vec{x})$, returning satisfied if its value is 1, and unsatisfied otherwise.
5:     Update: If satisfied,
           $$p_{i,j} = \begin{cases} 1 & \text{if } j = x_i \\ 0 & \text{otherwise.} \end{cases}$$
       If unsatisfied,
           $$p_{i,j} = \begin{cases} (1-b)p_{i,j} + a/(D-1+a/b) & \text{if } j = x_i \\ (1-b)p_{i,j} + b/(D-1+a/b) & \text{otherwise,} \end{cases}$$
       where $a, b \in [0, 1]$ are design parameters.
6:     Realise a random variable, selecting $x_i = j$ with probability $p_{i,j}$.
7: end loop
```

An instance of the CFL algorithm is run in parallel for every vertex of the graph to be coloured (*e. g.* when a vertex corresponds to a cell in a wireless network, an instance of the algorithm is run on every base-station/access-point). By construction, the only information used by the algorithm is whether or not any neighbour shares the same colour as i. Moreover, the CFL algorithm also satisfies the criterion that proper colourings are absorbing states (so once one is found, the vertices will settle on that assignment). The i-th instance of the algorithm maintains a vector p_i of length M. The j-th element $p_{i,j}$ gives the probability that instance i chooses colour j. Vector p_i is updated according to Algorithm 1.

In summary, when *satisfied i. e.* the current colour choice is distinct from that of all neighbours, vertex i keeps using this colour. Otherwise, it updates p_i to decrease the probability of selecting this colour again and then randomly chooses a new colour with probability given by vector p_i. The algorithm contains two parameters, a and b, which determine the algorithm's aversion to colours that lead to a failure: as a is made larger, failures are penalised more heavily and the "stickiness" of the algorithm towards previously successful allocations is reduced; b affects the speed of convergence.

The CFL algorithm is decentralised, meaning that it does not impose the requirement and overhead of communication between the vertices.

3.1.3.1 *Convergence Properties*

Define $\gamma = \min(a, b)/(D - 1 + a/b)$. We state now the theorem from Duffy et al. [2013].

Theorem 3.1. *With probability greater than* $1 - \varepsilon \in (0, 1)$, *the number of iterations for Algorithm 1 to find a satisfying assignment is less than*

$$(N) \exp(2N \log(\gamma^{-1})) \log(\varepsilon^{-1}).$$

We will extend this theorem in two different directions:

- in Chapter 4, we prove a similar convergence rate when each vertex has partial information;
- in Chapter 5, we prove a sub-exponential convergence rate when the number of colours is greater than Δ.

3.1.3.2 *Asynchronous Updates*

The CFL algorithm has been designed to run in parallel in a synchronous manner [Duffy et al., 2013]. To make the algorithm suitable for more general scenarios, we allow now the vertices to have asynchronous updates, with the constraint that the update period T of each vertex is the same. In other words, each vertex updates every T seconds, so the vertices will update one after the other according to an order induced by their relative time offset. We also assume the time of the execution of the algorithm steps is negligible, so no vertex update will happen during the update of another. We will show that the CFL algorithm is guaranteed to converge with high probability even in this case.

Comparing this version of the CFL algorithm with that presented by Duffy et al. [2013], we can notice that in Algorithm 1 the selection of a new colour has been moved to be *immediately after* the update of the probability vector (see step 6). This is necessary to ensure that when the updates are asynchronous the sampling of a new colour is made considering the most recent state possible, *i. e.* satisfaction is tested immediately before the sampling. This change does not affect the behaviour of the algorithm when the updates are synchronous, in the sense that Algorithm 1 is equivalent to the CFL algorithm when the updates are synchronous.

We refer to Theorem 4.2 for a proof of the convergence rate of Algorithm 2 (that generalises Algorithm 1 in case of partial sensing), where we can notice that we never consider updates of more than one vertex per time, so the order in which vertices update does not affect the proof.

3.2 SELF-CONFIGURATION OF SCRAMBLING CODES FOR WCDMA SMALL CELL NETWORKS

We now introduce the problem of Primary Scrambling Code (PSC) selection in Wideband Code Division Multiple Access (WCDMA) small cell networks, and show how the CFL algorithm can be used to allocate scrambling codes in a fully decentralised manner.

The PSC used by a base-station in UMTS and HSPA acts as an identifier and it is important that neighbouring base stations employ different scrambling codes in order to correctly manage handover and cell association. When the pilot signal power of a base-station received at the User Equipment (UE) is above a threshold, the UE will add the base-station to its monitored set and report the measurements to the serving base-station. If two or more base-stations in this set use the same PSC, *code confusion* occurs and the network cannot correctly distinguish between them. This is potentially an extremely severe problem as it can prevent correct handover and make the cells unserviceable. The 3GPP requirements for macrocells therefore specify that direct neighbours and second order neighbours use different PSCs.

In legacy macrocells, the PSC is chosen from a total of 512 available codes and, with such a large number of available codes, scrambling code allocation is a fairly straightforward task [Holma and Toskala, 2000]. This task is mostly carried out manually or through centralised algorithms using cluster reuse-based techniques [Chang et al.] or centralised graph colouring schemes [Jung and Lee, 2001]. While the Neighbour Cell List (NCL) can be constructed manually with potential enhancement through drive tests to include missing neighbours by constructing the cell overlapping matrix for legacy macrocell networks [Soldani and Ore, 2007], this would not be applicable to small cells considering their unplanned deployment. Therefore a number of PSCs are reserved specifically for small cells. Because small cells can be deployed within the coverage area of any macrocell, all macrocells need to add this set to their existing NCL. To avoid excessively long NCLs, the number of reserved PSCs is kept very small (3 or 4 in current implementations). A too long NCL leads to slower neighbour detection, measurement and cell acquisition [Guey et al., 2005].

For small cells, scrambling code allocation is much more challenging for four main reasons:

- Small cells are not only constrained to choose from amongst only very few reserved scrambling codes but also are typically more densely deployed: it is not even guaranteed a non-interfering allocation exists;
- Dynamic allocation is required due to the unplanned, organic nature of deployments;
- Allocation algorithms are constrained to use little or no message passing between base stations in order to ensure scalability to large network sizes;
- Since the reserved scrambling codes for small cells is a unique set that is added to all macrocells NCL, intelligent code planning techniques such as choosing the codes from the same code group or choosing identical codes from different groups becomes more difficult.

The main contributions of the present work are: (a) The formal definition of the dynamic scrambling code allocation problem for small cells; (b) The application of the CFL algorithm to dynamic scrambling code allocation; (c) Analysis and performance evaluation of this algorithm using numerical simulations. The results show a dramatic improvement (up to 150 % reduction in code confusion when a feasible solution exists) with respect to the 3GPP recommended scheme.

3.2.1 *Problem Statement*

We introduce the following notation:

- Let $\mathcal{B}$ denote the set of small base-stations, with $|\mathcal{B}| = N$.
- Let $\mathcal{S}$ denote the set of PSCs reserved for small cells, with $|\mathcal{S}| = M$.

- Let $x_i \in \mathcal{S}$ denote the scrambling code associated to base-station $i \in \mathcal{B}$, and $\vec{x} \in \mathcal{S}^N$ a global scrambling code allocation.
- Let $r_i(p)$ the Received Signal Code Power (RSCP) from the Common Pilot Channel (CPICH) of base-station $i \in \mathcal{B}$ measured at location $p \in \mathbb{R}^2$, and $r_{i,j} := r_i(p_j)$, where p_j is the location of device j (which might be a UE or another base-station). Note that the received power of pilot signal i measured by a device served by a cell j may be asymmetric, *i.e.* $r_i(p_j) \neq r_j(p_i)$.

Let $m_{i,j}$ denote the maximum ratio of received pilot power of base-station i to that of base-station j in the coverage area C_j of base-station j, *i.e.* $m_{i,j} := \max_{p \in C_j} \frac{r_i(p)}{r_j(p)}$. The *neighbour set* for base-station i is defined as $n_i = \{j : j \in \mathcal{B} \colon w_{j,i} > 0\}$, where

$$w_{j,i} = \begin{cases} m_{j,i} & \text{if } m_{j,i} > T \\ 0 & \text{otherwise,} \end{cases}$$

and T is an appropriate threshold value. We define the network *confusion graph* be the weighted directed graph $G := (\mathcal{B}, E, w)$ with vertices $\mathcal{B}$ and edges $E := \{(i, j) : j \in n_i\}$. That is, the graph with network base-stations as vertices and edge between each vertex/base-station i and every member of its neighbour set n_i. Each edge between two neighbours (i, j) is assigned weight $w_{i,j}$. Graph G is fundamental because if every vertex is assigned a different PSC from all of its neighbours then code confusion cannot occur. We call such an allocation a *proper* code allocation. Conversely, if any neighbour in this graph chooses the same PSC then code confusion may occur.

We define the *utility* of a code allocation as:

$$U(\vec{x}) = \frac{\sum_{i=1}^{N} u_i}{N}, \quad u_i = \begin{cases} 1 & \text{if } x_i \neq x_j \text{, for all } j \in n_i \\ 0 & \text{otherwise.} \end{cases}$$

The utility of a proper code allocation $\vec{x}$ is $U(\vec{x}) = 1$, while if all base-stations have at least one neighbour with the same PSC then $U(\vec{x}) = 0$. Given a confusion graph G and a set $\mathcal{S}$ of codes, the code allocation task is to find a code allocation $\vec{x}$ with maximal utility $U(\vec{x})$. We say that this code allocation task is *feasible* if at least one proper choice of PSCs exists (and so the maximal $U(\vec{x}) = 1$). Whether a feasible

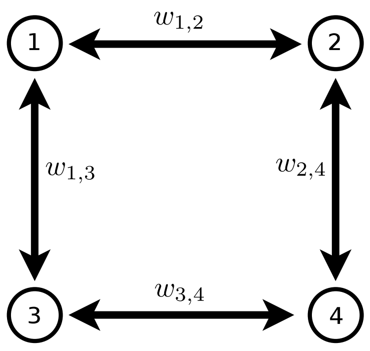


Figure 2: Simple example of a confusion graph G for a scenario consisting of 4 base-stations with symmetrical weights, and $w_{1,2} < w_{2,4}, w_{3,4} < w_{1,3}$.

allocation exists depends both on the graph G and the number M of codes available.

As an example, Figure 2 shows a simple case of a confusion graph G for a network consisting of 4 base-stations where the weights associated to the edges are symmetrical, *i. e.* $w_{i,j} = w_{j,i}$.

3.2.2 *Scrambling Code Selection*

In this section we introduce three decentralised algorithms for tackling the code allocation task. All algorithms are run by the base-stations in an asynchronous fashion, *i. e.* each base-station asynchronously updates its choice of PSC. This allows a simple implementation since femtocells are currently not synchronised. We will discuss how the ordering of updates can affect the performance of the algorithms.

3.2.2.1 *State of the Art: Single-step Greedy Algorithm (SGA)*

The Single-step Greedy Algorithm (SGA) is based on the current 3GPP recommendation [3GPP TR 25.967, 2011] that is envisioned as a potential solution while other vendor specific methods are concurrently encouraged. Each base-station $i \in \mathcal{B}$ scans the set of available PSCs and determines the set $\mathcal{S}_{\text{allocated}}(b) = \{x_j, j \in n_i\}$ of PSCs used by its neighbouring base-stations. From this it determines the set $\mathcal{S}_{\text{free}}(i) = \mathcal{S} \setminus \mathcal{S}_{\text{allocated}}$ of unallocated PSCs. If $\mathcal{S}_{\text{free}}(i)$ is non-empty, then base-station i picks a PSC uniformly at random from $\mathcal{S}_{\text{free}}(i)$. Otherwise, base-station i selects the code that is used by a neighbour of minimal weight $x_i = \underset{x_i \in S}{\operatorname{argmin}} \max_{j \in n_i} w_{j,i}$.

While appealingly simple, it is important to note that this greedy algorithm is not guaranteed to find a proper code allocation even in simple scenarios and in general its performance may be poor. For example, consider the confusion graph shown in Figure 2, with weights chosen such that $w_{1,2} < w_{2,4}, w_{3,4} < w_{1,3}$ and suppose $M = 2$ PSCs are available for use by small cells, $\mathcal{S} = \{A, B\}$. It can be readily verified that a proper code allocation exists, namely assigning code A to vertices 1 and 4 and code B to vertices 2 and 3. However, in the SGA it may happen that vertex 1 chooses code A, then vertex 4 chooses code B and now vertices 2 and 3 are unable to choose a code that ensures a proper allocation (vertex 2 will choose code A, conflicting with vertex 1 and vertex 3 will choose code B, conflicting with vertex 4). Indeed in this case the utility $U(s) = 0$.

3.2.2.2 *Iterative Greedy Algorithm (IGA)*

A refinement is to execute the SGA repeatedly rather than just in a one-shot manner, thereby yielding the Iterative Greedy Algorithm (IGA). However, in general this suffers from similarly poor performance to the SGA *e.g.* it is easy to verify that in the previous example for SGA the allocation assigning code A to vertices 1 and 2 and code B to vertices 3 and 4 is a reachable equilibrium point of IGA for the confusion graph in Figure 2, *i.e.* there is a significant probability this algorithm converges to an allocation with $U(s) = 0$.

3.2.2.3 *Communication-Free Learning (CFL) Algorithm*

Consider the unweighted, undirected version of graph G, *i.e.* $\tilde{G} := (\mathcal{B}, \tilde{E})$, with $\tilde{E} := \{(i,j) : (j \in n_i) \text{ OR } (i \in n_j)\}$. That is, the graph with network base-stations as vertices and an undirected edge between each vertex/base-station that can lead to code confusion.

We can then formulate the scrambling code allocation problem as a CP on graph $\tilde{G}$, where the colour set represents $\mathcal{S}$, and run Algorithm 1 asynchronously to solve it.

3.2.3 *Confusion Graph Estimation*

In order to decide the scrambling code, ideally each base-station i should know the RSCP levels $r_j(p), r_i(p), p \in C_i$ for all base-stations j. In practice, these quantities can be estimated based on UE reports.

However, if the UE reports are to be taken into account, the base-station requires to allow long enough time to collect the UE measurements before deciding the scrambling code, which can lead to slow convergence of the network[1].

A simpler way to construct the confusion graph is to rely on the CPICH RSCP measurements of other base-stations. This corresponds, in our notation, to constructing the approximate confusion graph G' measuring $m_{i,j} := \frac{r_i(p_j)}{r_j(p_j)}$ for each base-station j. Again, each base-station i needs to know only the values $m_{i,j}, j \in n(i)$. Similarly, the 3GPP recommendation [3GPP TR 25.967, 2011] suggests to use base-stations measurements for scrambling code selection.

While the algorithms introduced in this work can support both base-station based and UE-assisted graph construction, in order to make a fair comparison against the 3GPP scheme, the confusion graphs are constructed based on base-station measurements.

As an example, Figure 3-(a), illustrates the scenario where four base-stations are deployed within the conference hall at the Hynes convention centre in Boston. The heat-map shown in this graph refers to CPICH RSCP of base-station 1 (top-left) when the transmit pilot power is set to 10 dBm. The map is generated using the Wireless System Engineering (WiSE) [Fortune et al., 1995] software, which is a comprehensive 3D ray tracing based simulation package. The simulation results from WiSE have been validated by comparison against measurement data in a number of indoor and outdoor environments including the Hynes convention centre considered in this example. Figure 3-(b) shows the resulting approximate confusion graph G'.

3.2.4 *Results - Numerical Simulations*

Simulations were performed to evaluate and compare the performance of these three scrambling code allocation algorithms. Due to the complexity of the WiSE package and need for exhaustive global search for scrambling code allocation in many simulation scenarios, a simpler model was used. Varying number of base-stations are placed uniformly at random in a $100 \times 100\,\text{m}^2$ area. The update ordering of the base-stations is selected uniformly at random from the set of permutations of $\mathcal{B}$ for each sample. The maximum transmit power

1 We investigate the impact on convergence time when UE reports are used in Section 3.3.

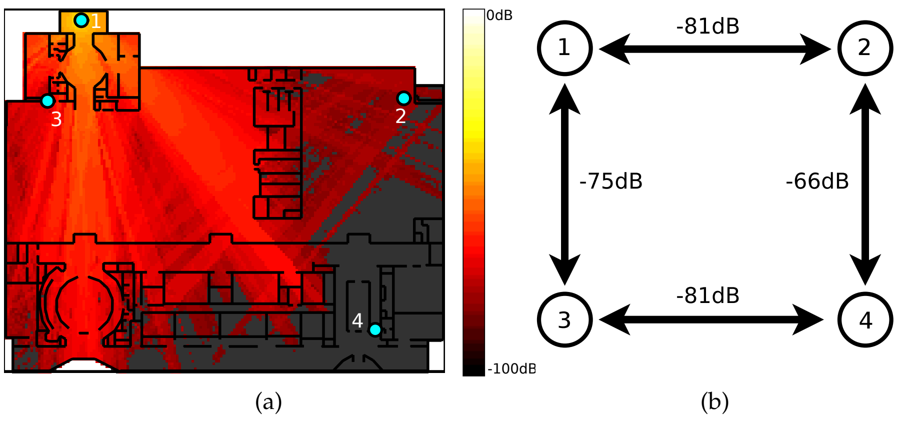


Figure 3: Example of small cell deployment within the Hynes convention centre in Boston: (a) Received pilot power for base-station 1, (b) Corresponding confusion graph G'.

is 100 mW and the pilot power is 10 % of this value [3GPP TS25.101, 2004]. Radio propagation path loss used is the 3GPP model for small cells [3GPP TR 36.814, 2010]. Simulation results are averaged over 100 independent runs.

3.2.4.1 *Comparison of Scrambling Code Selection Algorithms*

Figure 4 shows the boxplot of the utility function, $U(s)$, for all three algorithms and for scenarios where a feasible scrambling code allocation exist (choosing the number of available codes M equal to the chromatic number of the confusion graph). In this figure, markers indicate the mean value of utility function, the bottom and the top of the boxes indicate the 25th and 75th percentile of the utility function samples. The figure additionally indicates the minimum values underneath the bottom of the boxes. Without exception, the CFL always finds a proper solution ($U(\vec{x}) = 1$). In contrast, both IGA and SGA achieve mean utility of around 0.85 and also have relatively large degree of dispersion around this mean. The improvement of the utility function is up to 150 %.

Figure 5 shows the utility function versus the number of base-stations. The results are illustrated for the scenario when $M = 3$ scrambling codes are reserved for small cells (and so a proper code allocation may not exist). While the overall trend is expectedly descending, the CFL is shown to outperform the other two greedy algorithms by up to 50 %. Additionally, the optimal allocation has been calculated with exhaustive search and compared against the CFL where

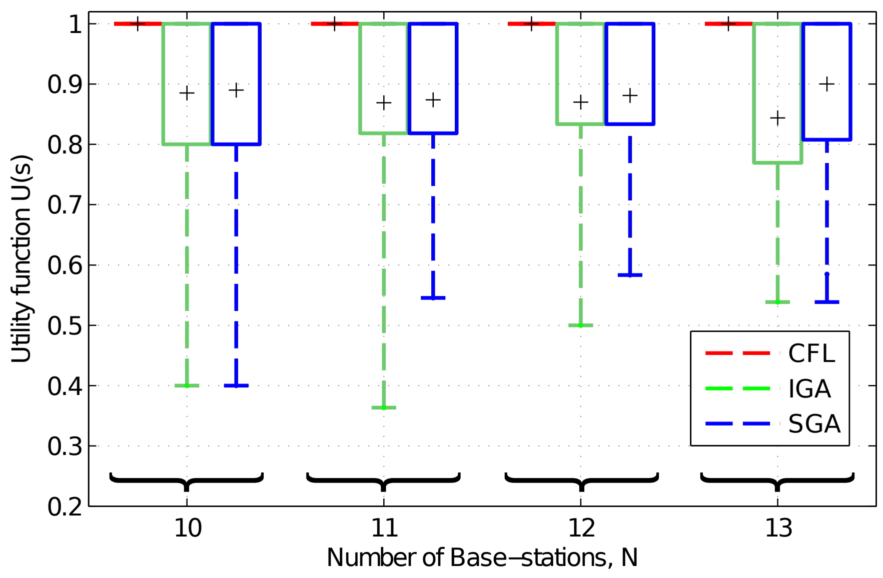


Figure 4: Comparison of SGA, IGA and CFL (with $a = b = 0.1$ algorithms performance when a feasible code allocation exists.

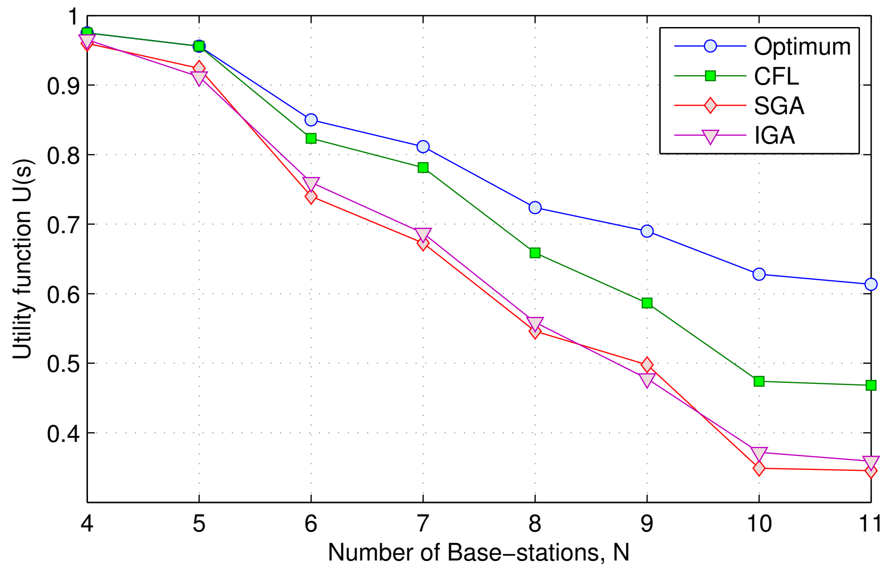


Figure 5: Mean utility function $U(s)$ versus number of base-stations for optimal scrambling code allocation and SGA, IGA and CFL algorithms when $M = 3$.

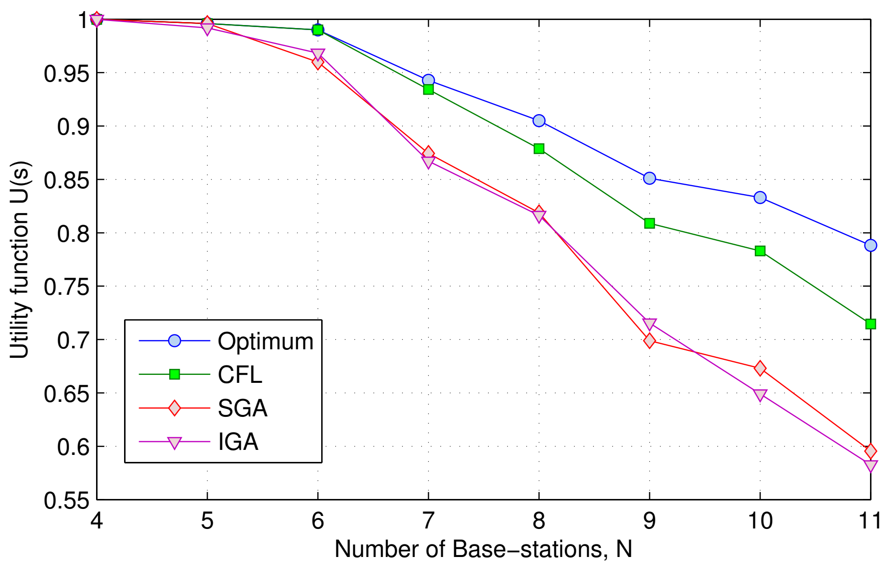


Figure 6: Mean utility function $U(s)$ versus number of base-stations for optimal scrambling code allocation and SGA, IGA and CFL algorithms when $M = 4$.

the CFL is shown to perform relatively close to the optimal especially when considering practical densities. It should be note that the CFL performance is considered to be acceptable given that the code allocation problem is NP-hard and that CFL is a fully distributed algorithm which relies on individual base-station decisions without any message passing (*i.e.* no centralised knowledge of the network topology is available). Similarly, Figure 6 shows the results for the case when the number of scrambling codes is increased to 4. Compared to Figure 5, the results show significant improvement in terms of the average utility function as a result of the added scrambling code. Again, CFL is shown to be superior to both SGA and IGA and the performance is even closer to the global optimum.

3.2.4.2 *Impact of NCL size*

To provide a better understanding of the effect of the number of available scrambling codes, Figure 7 shows the optimal utility function averaged over 100 simulation runs for increasing number of base-stations and varying number of available scrambling codes. As the results show, increasing the number of scrambling codes has a significant effect on the supported density of small cell deployments. The downside would evidently be the increased size of the NCLs. Using the model introduced in [Kourtis, 2000], Figure 8-(a) shows the

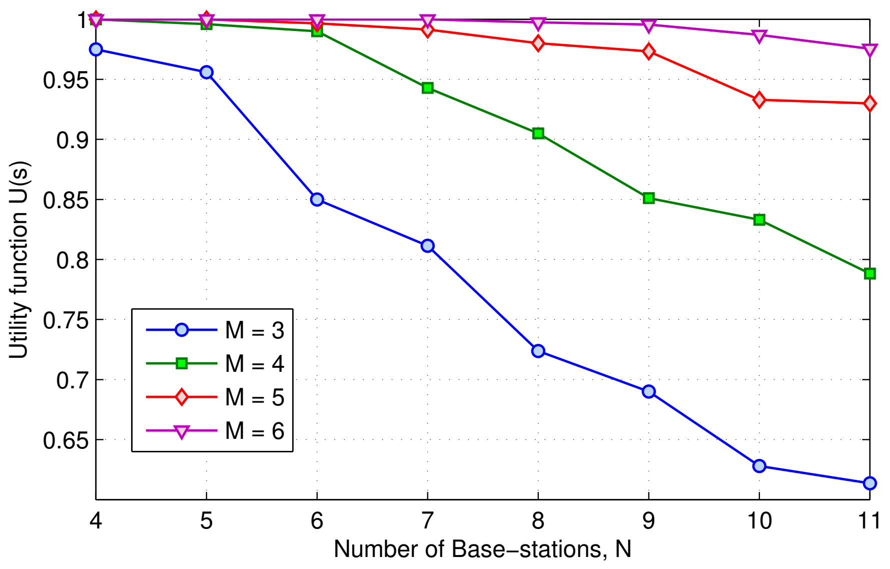
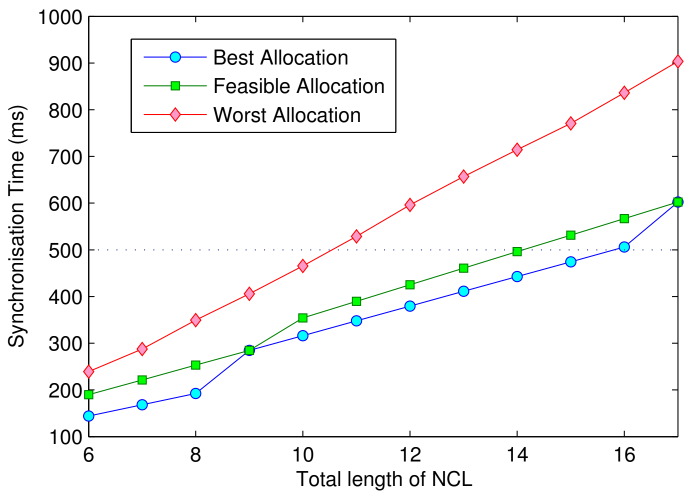


Figure 7: Mean utility function U(s) corresponding to the optimal allocation for different number of base-station and varying number of scrambling codes, M.

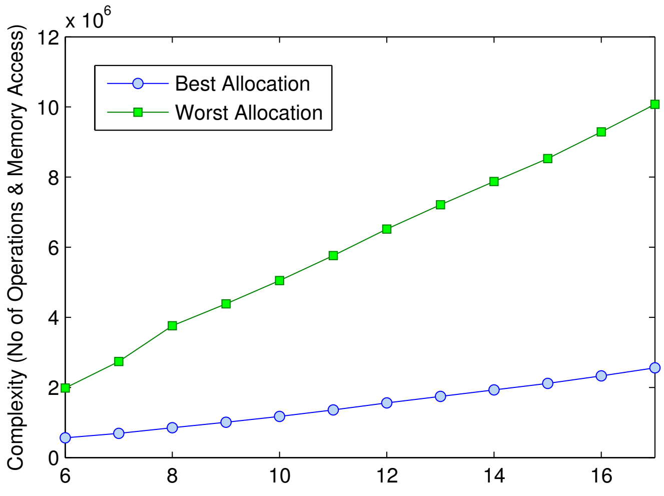
overall synchronisation time required for target cell measurements as function of the NCL size. In fact, the time required for synchronisation depends on how scrambling codes are allocated between the neighbouring cells. In general, there is a trade-off between the synchronisation time and the complexity (in terms of the number of operations and memory access) when scrambling codes are planned. While the synchronisation time strongly depends on the number of code groups within a given NCL, the complexity rises with the number of individual codes [Kourtis, 2000]. The reason for this is because the final code detection is more robust than the code group detection stage albeit at the cost of increased complexity.

Moreover, poor code group detection degrades the overall synchronisation process substantially and results in lengthy code acquisition. As an example, for an NCL of size 12, if 6 identical scrambling codes are selected from 2 groups, the required synchronisation time per cell is 31.9 ms with the associated complexity of 2.59×10^5 operations and memory access. The corresponding values are 40.2 ms and 1.43×10^5 operations if the 2 codes are selected from 6 groups.

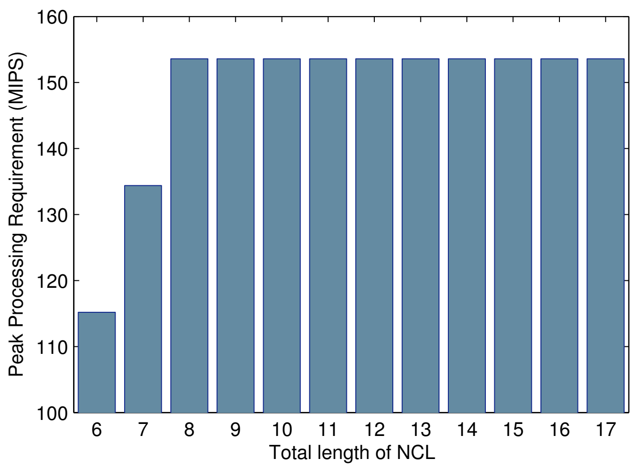
Since the codes that are decided for small cells ought to be fixed, it may not be possible to choose them from the same group as the macrocells. However, it is still possible to select the codes dedicated to small cells from the same group. In Figure 8-(a), considering the



(a)



(b)



(c)

Figure 8: (a) Synchronisation time (b) Associated complexity (total number of operations and memory access) and (c) Peak processing requirement as a function of NCL size.

synchronisation time as the performance metric, the worst case allocation refers to when all codes are selected from different groups and the best allocation corresponds to choosing codes from minimum possible number of code groups.

Given that each code group consists of 8 codes, there is a sharp jump in the synchronisation time for the best allocation when the NCL size is increased from 8 to 9 (or from 16 to 17) which imply the need for a new code group. Furthermore, the feasible allocation refers to the case when the code group of which small cell codes are selected is different to that of the macrocells whilst the number of code groups is kept at minimum.

Given that the measuring period is 500 ms [Kourtis, 2000], Figure 8-(a) confirms that the size of the NCL should not exceeds 14 cells for heterogeneous networks. Therefore, the number of small cells reserved codes depends on the number of codes allocated to macrocells. While with an ideal hexagonal shape macrocells' layout, merely 6 codes are sufficient to represent the neighbouring cells, in reality, a NCL size of 10 for macrocells is still not too conservative. This is especially the case considering dense deployment of the macrocells in urban areas. The exact number of required codes, depends on the network topology including the base-stations location, their antennas orientations and their pilot power settings.

Figure 8-(b) shows the complexity of the target cell search procedure as a function of the number cells within a given NCL. In contrary to Figure 8-(a), the best allocation here refers to when identical codes are selected all from different groups and worst allocation represents the scenario when many individual codes (up to 8) are selected used. While it is desirable to minimise the number of operations and memory access to allow longer battery life of UEs, this will not be a prime performance concern as long as UEs can handle the computation complexity.

For this reason, the worst case peak processing requirement, in Million Instruction Per Second (MIPS), was calculated and shown in Figure 8-(c). Since the peak processing complexity depends on the number of individual codes, it remains the same after the NCL size exceeds 8. Fortunately, considering the processing capabilities of mobile phones [Aguilar, 2008], the worst case peak processing requirement of 153 MIPS is still well bellow almost all mobile phones. The margin is over 92 % when considering more recent phones.

Comparing the results shown in Figure 8, it is evident that for heterogeneous networks consisting of small cells, the best scrambling code allocation strategy is to choose the codes from a minimum possible number of code groups.

3.2.5 *Summary*

This work introduces the problem of scrambling code allocation for Wideband Code Division Multiple Access (WCDMA) small cell networks. The problem differs from code planning in the legacy macrocell networks due to the limited number of codes reserved for small cells, the need for dynamic adaptation and for scalable, distributed allocation algorithms. Additionally, a novel decentralised (with no message passing) algorithm for dynamic scrambling code allocation is proposed and its performance was evaluated against two variants of 3GPP recommended schemes. The results confirm significant performance improvement of the utility function (up to 150 %) when using the proposed scheme. The proposed scheme is fully distributed and is of low computation complexity which makes it suitable for unplanned deployment of small cells base-stations.

Finally, the work discuss the trade-offs involved in increasing the number of codes reserved for small cells. While there is a significant improvement in term of supported small cell deployment density when the number of reserved codes is increased to 5 or 6, results shows that the NCL size may not exceeds a total of 14 cells for a synchronisation time of 500 ms. Additionally, the work confirms the importance of scrambling code planning for cells belonging to a given NCL. Since the synchronisation time is shown to be the prime limiting factor, the best code allocation strategy for heterogeneous networks is the one that results in minimal number of code groups.

3.3 CROWD SOURCING

In this section we consider in more detail the problem of topology discovery, already touched on in Section 3.2.3, and in particular estimating topology by means of user reports in networks which are deployed in an unplanned, decentralised manner. To achieve LTD it is sufficient that each node of the network is aware of its nearest neighbours only. However, a base-station/access-point may not itself be able to detect all conflicts that its users may experience. For example, it may happen that a user is reached by both the serving access point and one of the neighbours, and the conflict is not detected either because of hidden node effects or because of hardware limitations (*e. g.* femtocells are half-duplex).

We tackle this problem via *crowd sourcing*, meaning that users detect and report the existence of conflicting neighbours using, for example, the technology described in [Sapiano, 2010]. Given a "reasonable" user-mobility model, an important problem to address is how much information we expect to obtain from user reports. Closely related to this question is the problem of estimating the minimal density of users that guarantees full knowledge of the local topology. Such an estimate enables in turn to determine the class of network deployments that can effectively benefit from this approach.

Our main contributions are the following: (i) the problem of user-reports-based local topology discovery is stated for the first time through a crisp mathematical formulation; (ii) the topological structure of the serving-AP coverage area is mapped onto states of increasing knowledge, in this way the use of a general user mobility model is naturally allowed for; (iii) the problems mentioned above are given an answer in the case of a simplified mobility model (Model 1 in Section 3.3.2), useful for gaining insight into those situations where user reports function is likely to give the greatest benefit; (iv) this simplified model is shown, under certain conditions, to provide an upper bound on the time of topology discovery, thus it can be used as a design tool (see Section 3.3.2.5).

3.3.1 *Local Topology Discovery Model*

Given a set of wireless APs $\mathcal{A} = \{a_0, \dots, a_N\}$, let $A(a_i) \subset \mathbb{R}^2$ denote the coverage area of access point a_i. We note that in general $A(a_i)$ depends on the transmission power of a_i and on the radio propagation properties of the medium. We focus on serving access point a_0. Let $\mathcal{B}$ denote the neighbouring APs that have non-void intersection with $A(a_0)$, that is,

$$\mathcal{B} = \{a_i \in \mathcal{A}, i > 0 \colon A(a_0) \cap A(a_i) \neq \emptyset\}.$$

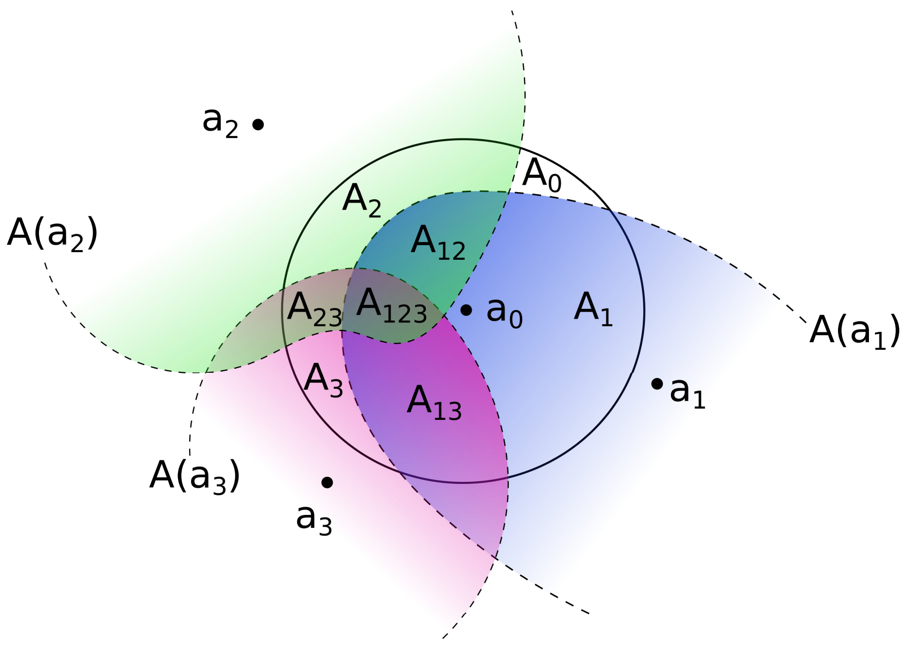


Figure 9: Example of a scenario in which the access point a_0 has three interfering neighbours: a_1, a_2, a_3. The coverage area $A(a_0)$ can be tessellated with the sets $A_1, A_2, A_3, A_{12}, A_{13}, A_{23}, A_{123}$.

We will hereafter use the symbol N to denote the cardinality of $\mathcal{B}$, *i. e.* $N = |\mathcal{B}|$.

Let $\mathcal{P}(\mathcal{B})$ denote the powerset of $\mathcal{B}$. A *tessellation* of the area $A(a_0)$ is the collection of tiles $\{A_{\boldsymbol{i}}\}_{\boldsymbol{i}\in\mathcal{P}(\mathcal{B})}$ such that

$$A(a_0) = \bigcup_{\boldsymbol{i}\in\mathcal{P}(\mathcal{B})} A_{\boldsymbol{i}}\,, \tag{11}$$

where

$$\begin{aligned} A_{\boldsymbol{i}} =& \bigcap_{j\in\boldsymbol{i}} A(a_j) \cap A(a_0) \setminus \bigcup_{j\notin\boldsymbol{i}} A(a_j), \quad \boldsymbol{i} \neq \emptyset, \\ A_{\emptyset} =& A(a_0) \setminus \bigcup_{\boldsymbol{i}\in\mathcal{P}(\mathcal{B})\setminus\emptyset} A_{\boldsymbol{i}}. \end{aligned} \tag{12}$$

In what follows each element $A_{\boldsymbol{j}}$ composing the tessellation is referred to as a *tile*, and we will use the vector notation $\boldsymbol{j}$ to represent a set of neighbouring APs. Let us consider for example $\boldsymbol{i} = \{a_1, a_2\}$; then, the tile $A_{\boldsymbol{j}}$ is the portion of $A(a_0)$ that is covered by a_1 and a_2 only, see Figure 9. For simplicity of notation, we will write $A_0 \coloneqq A_{\emptyset}$.

Whenever a user is in $A_{\boldsymbol{j}}$, it will *report* $\boldsymbol{j}$ to access point a_0. In other words, a_0 will be aware of the existence of those neighbouring

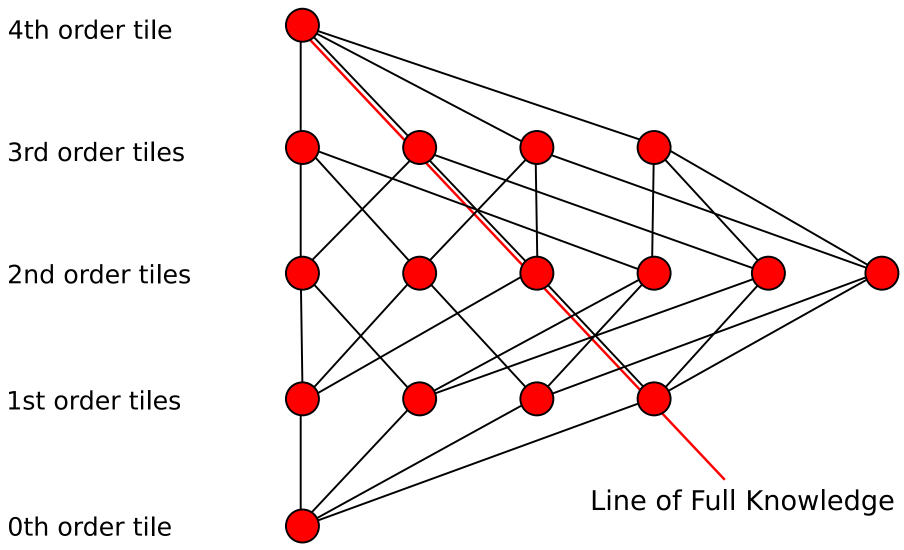


Figure 10: Hypercube representation of the tessellation for $N = 4$. There is one zeroth order tile, namely A^C, four first order tiles, A_1, A_2, A_3 and A_4, six second order tiles, $A_{12}, A_{13}, A_{14}, A_{23}, A_{24}, A_{34}$, four third order tiles, $A_{123}, A_{124}, A_{134}, A_{234}$ and a fourth order tile, *i. e.* A_{1234}.

APs $a_i \in \boldsymbol{j}$. The rate of these reports depends on the mobility model assumed.

To keep the model as conservative as possible, and to encompass the frequent case of half-duplex APs, we assume a_0 can not detect the existence of any neighbour even though a_0 lies in one of the neighbours coverage area.

Let $\mathcal{K}_t$ denote the *knowledge set* of access point a_0 at time t, *i. e.* the set of neighbours that a_0 is aware of. Given a sequence of reports $\{\boldsymbol{j}^1, \ldots, \boldsymbol{j}^t\}$, we have that $\mathcal{K}_t = \bigcup_{i=1}^{t} \boldsymbol{j}^i$. $\mathcal{K}_t$ is a sequence of sets that satisfies

$$\mathcal{K}_t = \bigcup_{s=0}^{t} \mathcal{K}_s; \tag{13}$$

in particular, $|\mathcal{K}_t|$ is non-decreasing in t. Clearly, the knowledge state at time t, $\mathcal{K}_t$, take values in $\mathcal{P}(\mathcal{B})$.

Definition 7 (Full Knowledge)**.** Given an integer T and a finite sequence of reports $\{\boldsymbol{j}^1, \ldots, \boldsymbol{j}^T\}$, the AP a_0 is said to have *Full Knowledge* (FK) of its neighbours at time T, if

$$\mathcal{K}_T = \bigcup_{s=1}^{T} \boldsymbol{j}^s = \mathcal{B}.$$

Remark 1. If a_0 has Full Knowledge (FK) of its neighbours at time T, it does so at all times $T+t$, $t \geqslant 0$. In other words, once a_0 has reached FK, it can not lose it.

Definition 8 (First time of FK)**.** Given a sequence of reports $\{\boldsymbol{j}^1, \boldsymbol{j}^2, \dots\}$, the *first time of FK* τ for the AP a_0 is the first time the latter reaches FK of its neighbours, *i. e.*

$$\tau := \min\{T \geqslant 0 \text{ such that } \mathcal{K}_T = \mathcal{B}\}. \tag{14}$$

Remark 2. The characterisation of the first time of FK will depend in general on the realisation of a sequence of user reports; this means that τ is a random variable. We want to remark that by (14), τ is a *stopping time*, see *e. g.* [Levin et al., 2009].

Remark 3. A generic tessellation of $\mathcal{B}$ can be represented as an hypercube $H = \{0,1\}^N$ by identifying the vertices of H with the tiles $A_{\boldsymbol{j}}$ that the tessellation is composed of. The number of tiles of a generic tessellation of $\mathcal{B}$ is 2^N as well as the vertices of an hypercube, represented as vectors of size N. The tiles of the tessellation and the vertices of the hypercube are mapped one to one as soon as the i-th component of the vertices $x \in H$ is identified with $a_i \in \mathcal{B}$. In other words,

$$A_{\boldsymbol{j}} \leftrightarrow x \quad \Leftrightarrow \quad x_i = \mathbb{1}_{\{a_i \in \boldsymbol{j}\}},\ i = 1, \dots, N,$$

where $\mathbb{1}$ is the indicator function. We define the *order* of a tile as the number of neighbours a report from that tile would give knowledge of. The number of k-th order tiles is $\binom{N}{k}$. A report from a k-th order tile is equivalent to k first-order reports. In particular, FK is attained with a report from the N-th order tile, or at least two reports from two distinct $(N-1)$-th tiles, etc. This property can be graphically represented by what we call the *Line of Full Knowledge*, see Figure 10. The line of FK is clearly not unique[2]; the aim of Figure 10 is only to illustrate that a sequence of T reports $\{\boldsymbol{j}^1, \dots, \boldsymbol{j}^T\}$ is a path on the hypercube H, and that FK is attained whenever a line of FK is reached at a time smaller than T.

Since the knowledge state at time t, $\mathcal{K}_t$, takes values in the same set $\mathcal{P}(\mathcal{B})$, we can map the knowledge states on the hypercube H. In other word, a sequence of reports $\{\boldsymbol{j}^1, \boldsymbol{j}^2, \dots, \boldsymbol{j}^t\}$ is equivalent to a single report from tile $\sum_{s=1}^{t} \boldsymbol{j}^s = \mathcal{K}_t$.

2 For example, there are N tiles of order $N-1$, but only 2 are part of a given line of FK.

We can now define the main problems of this work.

Problem 3.1 (Expected first time of Full Knowledge)**.** *Given an access point a_0, a set $\mathcal{B}$ of neighbours with given position and coverage area and a sequence of user reports, we want to characterise the expectation of the* first time of FK, i. e.

$$\mathbb{E}(\tau) = \sum_{t \geqslant 1} t \, \mathbb{P}(\tau = t) \, .$$

Obviously, the way the user(s) move inside the coverage area $A(a_0)$ heavily affects the difficulty of the problem and its answer. However, the formulation of Problem 3.1 has the great advantage of decoupling the notion of FK from the user mobility model; addressing the mean value of the first time of FK is also an enabler to the estimate of the tail of the distribution of τ – through Markov's inequality, for example. Further, from a numerical point of view, the expected time of FK may be achieved via a Monte Carlo simulation once the set $\mathcal{B}$ and the mobility model in use are fixed.

There may exist situations where we are content to characterise the first time in which only partial knowledge of the local topology is attained. For example, we may be interested in the first moment when the neighbouring APs that have been already discovered, *i. e.* the elements of the knowledge set $\mathcal{K}_t$, are enough to describe a given fraction of the local topology. This idea motivates the following

Problem 3.2 (Expected first time of δ-knowledge)**.** *Given an access point a_0, a set $\mathcal{B}$ of neighbours with given position and coverage area, let ρ be a measure over $\mathcal{P}(\mathcal{B})$. Fixed $\delta \in (0, 1]$, we want to characterise the expectation of the* first time of δ-knowledge $\mathbb{E}(\tau_\delta)$*, where*

$$\tau_\delta = \min \left\{ t \geqslant 0 \text{ such that } \frac{\sum\limits_{\boldsymbol{k} \in \mathcal{P}(\mathcal{K}_t)} \rho(A_{\boldsymbol{k}})}{\sum\limits_{\boldsymbol{j} \in \mathcal{P}(\mathcal{B})} \rho\left(A_{\boldsymbol{j}}\right)} \geqslant \delta \right\} .$$

When $\delta = 1$ and $\rho(A_{\boldsymbol{j}}) > 0$ for each $\boldsymbol{j} \in \mathcal{P}(\mathcal{B})$, Problem 3.2 is equivalent to Problem 3.1. Indeed, $\sum_{\boldsymbol{k} \in \mathcal{P}(\mathcal{K}_t)} \rho(A_{\boldsymbol{k}}) / \sum_{\boldsymbol{j} \in \mathcal{P}(\mathcal{B})} \rho(A_{\boldsymbol{j}}) \geqslant 1$ if and only if $\mathcal{K}_t \equiv \mathcal{B}$.

We will hereafter consider the Lebesgue measure $\rho(A_{\boldsymbol{k}}) = \|A_{\boldsymbol{k}}\|$. This leads to the following interpretation: δ-knowledge is attained when the knowledge set $\mathcal{K}_t$ defines for the first time a tessellation that covers a fraction of $A(a_0)$ larger or equal than δ. Equivalently, τ_δ

is the first time when the fraction of $A(a_0)$ covered by the tiles that would give new information[3] is less than $1 - \delta$.

Remark 4. The concept of δ-knowledge is fundamental in the simulation phase, when we want to know whether user reports can effectively be used to give knowledge of the local topology. Indeed, it is likely that the neighbours a_i whose coverage area do not overlap with $A(a_0)$ save for a nearly negligible portion, will be discovered after a very long time; in other words, the leading contribution to $\mathbb{E}(\tau)$ will be represented by the mean first visit time of the user(s) to $A(a_i)$. Discarding a_i from the picture then, the concept of δ-knowledge let us focus on the quantitative analysis of the LTD, see Section 3.3.3.

3.3.2 *Teleport Mobility*

The characterisation of τ, the first time of FK, depends on the users mobility model that is assumed. This describes how users enter, exit, and move within $A(a_0)$.

The users evolution can then be represented as a pair $U_t = (n_t, X_t)$ where n_t is the number of users that lie in $A(a_0)$ at time t, and $X_t = (x_t^1, x_t^2, \dots, x_t^{n_t})$ is a vector with the position of the n_t users. We assume the evolution of U_t to be driven by a discrete-time Markov chain (MC) throughout the work.

The realisation of $\{U_t\}_{0 \leqslant t \leqslant T}$ completely determines the sequence of user reports $\{\boldsymbol{j}^1, \dots, \boldsymbol{j}^T\}$ to the access point a_0, cf. Remark 2. Since $\mathcal{K}_t$ only depends on $\mathcal{K}_{t-1}$ and U_t, then the bivariate process $(U_t, \mathcal{K}_t)$ is a MC.

It will prove useful to consider a simplified mobility model in which a single user can instantaneously teleport to any tile:

Model 1. (Teleport Mobility) A single user moves within $A(a_0)$ according to a discrete-time MC taking value on $\mathcal{P}(\mathcal{B})$. The user can not exit $A(a_0)$ and no other user can enter it. Assuming that all tiles are Lebesgue-measurable plane sets, the transition probabilities are

$$\mathbb{P}(\boldsymbol{i}, \boldsymbol{j}) = \frac{\|A_{\boldsymbol{j}}\|}{\|A(a_0)\|}, \tag{15}$$

where $\|\cdot\|$ denotes the Lebesgue measure.

3 In the sense that the cardinality of the knowledge set $\mathcal{K}_T$ would increase.

Remark 5. Model 1 greatly simplifies the characterisation of τ, the first time of FK. Indeed, with this mobility model $\mathcal{K}_t$ is independent of U_t, and the sole process $\mathcal{K}_t$ is hence sufficient to describe the process of gathering knowledge from the user reports. We will hereafter refer to $\mathcal{K}_t$ as the *knowledge chain*.

Assuming Model 1, we can easily describe the process of gathering knowledge from user reports as a discrete-time random walk on the hypercube $H = \{0, 1\}^N$ (which we have introduced in Remark 3); having knowledge of n neighbouring APs is in fact equivalent to receiving a report from the n-th order tile that gives information about all of them.

Let $P(\cdot, \cdot)$ the transition kernel of the knowledge chain. If $\boldsymbol{k} \not\subseteq \boldsymbol{l}$, then (13) guarantees that $P(\boldsymbol{k}, \boldsymbol{l}) = 0$ because such transition would mean a loss of knowledge. Conversely, when $\boldsymbol{k} \subseteq \boldsymbol{l}$, a transition from $\boldsymbol{k}$ to $\boldsymbol{l}$ happens if the user moves to a tile that contains the missing information $(\boldsymbol{l} \setminus \boldsymbol{k})$ and do not add more than that information. Therefore,

$$P(\boldsymbol{k}, \boldsymbol{l}) = \begin{cases} \displaystyle\sum_{\boldsymbol{m} \in \mathcal{P}(\boldsymbol{k})} \frac{\left\| A_{\{\boldsymbol{m} \cup (\boldsymbol{l} \setminus \boldsymbol{k})\}} \right\|}{\| A(a_0) \|} & \text{if } \boldsymbol{k} \subseteq \boldsymbol{l}, \\ 0 & \text{otherwise.} \end{cases} \tag{16}$$

The following result holds:

Lemma 3.1. *The matrix* P *is upper triangular.*

Proof. Let us consider the following partial ordering relation among the states:

$$\boldsymbol{k} \preceq \boldsymbol{l} \quad \Leftrightarrow \quad \boldsymbol{k} \subseteq \boldsymbol{l}\,.$$

By (16), $P(\boldsymbol{k}, \boldsymbol{l}) \neq 0$ only if $\boldsymbol{k} \preceq \boldsymbol{l}$. Therefore, any mapping

$$\mathcal{P}(\mathcal{B}) \ni \boldsymbol{l} \quad \leftrightarrow \quad l \in \{1, 2, \ldots, 2^N\}$$

such that

$$\boldsymbol{k} \preceq \boldsymbol{l} \quad \Leftrightarrow \quad k \leqslant l$$

will put the matrix P into an upper triangular form. In particular, we can order the states by increasing cardinality and in lexicographic order[4]. □

4 For N neighbouring APs, *i.e.* with 2^N different tiles, this would mean the sequence $\{1\}, \{2\}, \ldots, \{N\}, \{1, 2\}, \ldots, \{N\text{-}1, N\}, \ldots, \{1, 2, \ldots, N\}$.

The explicit computation of the whole matrix P using (16) is expensive in general – P is a $2^N \times 2^N$ matrix! However, as stated above P is upper triangular, while in Section 3.3.2.3 we show that it is possible to explicitly characterise its spectrum. For the reader's reference, Table 1 shows the matrix P for $N = 3$.

$$\begin{pmatrix}
\|A_0\| & \|A_1\| & \|A_2\| & \|A_3\| & \|A_{12}\| & \|A_{13}\| & \|A_{23}\| & \|A_{123}\| \\
0 & \|A_0\|+\|A_1\| & 0 & 0 & \|A_2\|+\|A_{12}\| & \|A_3\|+\|A_{13}\| & 0 & \|A_{23}\|+\|A_{123}\| \\
0 & 0 & \|A_0\|+\|A_2\| & 0 & \|A_1\|+\|A_{12}\| & 0 & \|A_3\|+\|A_{23}\| & \|A_{13}\|+\|A_{123}\| \\
0 & 0 & 0 & \|A_0\|+\|A_3\| & 0 & \|A_{13}\|+\|A_1\| & \|A_2\|+\|A_{23}\| & \|A_{12}\|+\|A_{123}\| \\
0 & 0 & 0 & 0 & \|A_0\|+\|A_1\|+\|A_2\|+\|A_{12}\| & 0 & 0 & \|A_3\|+\|A_{13}\|+\|A_{23}\|+\|A_{123}\| \\
0 & 0 & 0 & 0 & 0 & \|A_0\|+\|A_1\|+\|A_3\|+\|A_{13}\| & 0 & \|A_2\|+\|A_{12}\|+\|A_{23}\|+\|A_{123}\| \\
0 & 0 & 0 & 0 & 0 & 0 & \|A_0\|+\|A_2\|+\|A_3\|+\|A_{23}\| & \|A_1\|+\|A_{12}\|+\|A_{13}\|+\|A_{123}\| \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{pmatrix}$$

Table 1: Example of transition matrix P (modulo scaling factor $\frac{1}{\|A_{a_0}\|}$) for $N = 3$.

3.3.2.1 *Expected time of Full Knowledge*

Let $\boldsymbol{k}^* = \{1, 2, \dots, N\}$, *i. e.* the state of FK. By formula (16), $P(\boldsymbol{k}^*, \boldsymbol{k}^*) = 1$. This means that the chain has an absorbing state, and the hitting time of this state is τ, the first time of FK. Hence, we can compute the expected time of FK simply by

$$\mathbb{E}[\tau] = (I - Q)^{-1}\mathbf{1}, \tag{17}$$

where Q is obtained from P by removing the row and the column relative to state $\boldsymbol{k}^*$ and $\mathbf{1}$ is the column vector of ones [Gehring and Halmos, 1976]. In a similar way it is possible to compute the other moments of τ.

Even if $I - Q$ is upper triangular and can be block decomposed, the computation of its inverse may not be affordable when the cardinality of $\mathcal{B}$ grows. In Section 3.3.2.4 we will bound the probability of the event $\{\tau > t\}$.

3.3.2.2 *Expected Time of δ-knowledge*

Regarding Problem 3.2, we can easily modify matrix P to obtain the expected time of δ-knowledge. Every state $\boldsymbol{k} \in \mathcal{P}(\mathcal{B})$ such that

$$\sum_{\boldsymbol{l} \in \mathcal{P}(\boldsymbol{k})} \frac{\|A_{\boldsymbol{l}}\|}{\|A_{a_0}\|} \geqslant \delta$$

can be aggregated in the absorbing state, summing the corresponding column of P in the last column, and then eliminating the column and row corresponding to state $\boldsymbol{k}$. In this way it is possible to compute $\mathbb{E}[\tau_\delta]$ using (17).

3.3.2.3 *Eigenvalues*

The following result reveals the spectrum of the matrix P:

Theorem 3.2. *For $\boldsymbol{k} \in \mathcal{P}(\mathcal{B})$, the eigenvalues of P have the form*

$$\lambda_{\boldsymbol{k}} = \frac{1}{\|A(a_0)\|}\left(\|A_0\| + \sum_{\substack{\boldsymbol{l}\subset\boldsymbol{k}\\|\boldsymbol{l}|=1}} \|A_{\boldsymbol{l}}\| + \cdots + \sum_{\substack{\boldsymbol{l}\subset\boldsymbol{k}\\|\boldsymbol{l}|=m}} \|A_{\boldsymbol{l}}\| + \cdots + \sum_{\substack{\boldsymbol{l}\subset\boldsymbol{k}\\|\boldsymbol{l}|=|\boldsymbol{k}|}} \|A_{\boldsymbol{l}}\|\right).$$

Proof. The matrix P being upper triangular by Lemma 3.1, the entries $P(\boldsymbol{k},\boldsymbol{k})$ are the eigenvalues of the matrix. Let us then imagine to have the knowledge chain in state $\boldsymbol{k}$. The only way for the chain to undergo a self-transition ($\boldsymbol{k} \to \boldsymbol{k}$) is that the user reports any combination of neighbouring APs that have already been discovered. In other words, the knowledge chain undergoes a self-transition if and only if the user reports an element of $\mathcal{P}(\boldsymbol{k})$. Therefore,

$$\lambda_{\boldsymbol{k}} = \frac{1}{\|A_{a_0}\|} \sum_{\boldsymbol{l}\in\mathcal{P}(\boldsymbol{k})} |A_{\boldsymbol{l}}|\,.$$

Last formula is equivalent to the thesis. □

Since each eigenvalue is a sum of positive elements, the second-largest eigenvalue $\tilde{\lambda}$ can be obtained by maximising over the tiles of order $N-1$:

$$\tilde{\lambda} = \max_{\boldsymbol{k}:\,|\boldsymbol{k}|=N-1} \lambda_{\boldsymbol{k}}. \tag{18}$$

3.3.2.4 *Convergence Properties, Bounds*

Thank to (18), it is now possible to compute a useful bound on the time of FK with high probability.

Lemma 3.2. *Given $\varepsilon > 0$, let*

$$S(1-\varepsilon) = \frac{\log \varepsilon}{\log \tilde{\lambda}}. \tag{19}$$

Then, $S(1-\varepsilon)$ reports are sufficient to achieve FK with probability greater or equal than $(1-\varepsilon)$.

Proof. Using Lemma 3.1 and Equation (18) on P,

$$\mathbb{P}(\tau > t) \leqslant \tilde{\lambda}^t. \tag{20}$$

For a small target tolerance ε of not achieving FK,

$$\text{if } t \geqslant \frac{\log \varepsilon}{\log \tilde{\lambda}} \quad \Rightarrow \quad \mathbb{P}(\tau > t) \leqslant \varepsilon. \tag{21}$$

□

δ-KNOWLEDGE CONVERGENCE BOUNDS Using the same manipulation of the matrix P described in Section 3.3.2.2, Corollary 3.2 can be applied to the modified matrix to obtain a bound for the number of steps to have δ-knowledge with high probability.

3.3.2.5 *Some Remarks on the Mobility Model*

Model 1 is equivalent to a single user teleporting instantaneously to a random point within the coverage area of the AP; time is discrete. Thus, at each time the AP a_0 receives user reports from point drawn according to the uniform probability distribution over the coverage area $A(a_0)$.

Let us now suppose that the user moves within the AP coverage area according to another discrete-time MC taking values on the coverage area; let us also suppose that such a MC has a unique stationary distribution, uniform over the coverage area; finally, suppose that the user communicates its new position after a number of steps that are sufficient for the MC to forget the past and reach equilibrium. Under these assumptions, when the user sends a report to the AP, its position is distributed according to the stationary (uniform) distribution of the chain. Therefore, the teleport mobility model offers the following nice interpretation: it is equivalent to any mobility model where a single user moves within the AP coverage area according to a discrete-time MC and sends reports to the AP at a rate which is smaller than the inverse mixing time of the chain. Indeed, the evolution of this new MC after a report is sent to the AP, by the *Strong Markov Property* (see *e.g.* [Levin et al., 2009]), is independent of the past trajectory; further, its future evolution is governed by the usual recursion

$$\mu^t(\boldsymbol{k}) = \sum_{\boldsymbol{l} \in \mathcal{P}(\mathcal{B})} \mu^0(\boldsymbol{l})\, q^t(\boldsymbol{k}, \boldsymbol{l}), \tag{22}$$

where μ^t is the probability distribution of $\boldsymbol{k}$ after t steps from last report, q is the chain transition kernel, and μ^0 is a mass concentrated in the point where the last report was sent from. If the reports are sent at a rate $r < 1/\tau_{\text{mix}}$, where τ_{mix} is the mixing time of the chain, then $\mu^{1/r} \approx \pi$, *i. e.* the chain is approximately at equilibrium.

Clearly, the scenario described above does not really depend on the MC having uniform stationary measure. Given a target probability distribution π over the AP coverage area and an ergodic MC having π as its unique stationary distribution, let us imagine that a single user moves within the coverage area according to the chain and sends reports to the AP at a rate smaller than the inverse mixing time. Then, we can generate a sequence of user reports by sampling the measure π. In this case the matrix P describing the knowledge evolution becomes

$$P(\boldsymbol{k},\boldsymbol{l}) = \begin{cases} \sum_{\boldsymbol{m}\in\mathcal{P}(\boldsymbol{k})} \pi\Big(A_{\{\boldsymbol{m}\cup(\boldsymbol{l}\setminus\boldsymbol{k})\}}\Big) & \text{if } \boldsymbol{k}\subseteq\boldsymbol{l}, \\ 0 & \text{otherwise.} \end{cases} \qquad (16')$$

It is questionable whether it is reasonable to assume that user reports are sent at a frequency that is lower than the inverse mixing time of the MC describing the user mobility. As we point out in Section 3.3.3.2, high-frequency reports may not correspond to an achievement of FK after a small number of reports. Let us imagine that a single user moves within $A(a_0)$ according to a Brownian motion and sends reports with a high frequency. It is reasonable to expect that many successive reports will be sent from the same tile, adding then no further information about the local topology. We come back to this point in Section 3.3.3.2.

All in all, we can conclude that Model 1, alongside the low-frequency reports assumption, may represent a viable option to the study of LTD in actual situations.

3.3.3 *Simulations*

3.3.3.1 *Teleport Model on Random Positioned APs*

We developed a simulation framework in MATLAB, where 8 APs have been positioned in the space u.a.r. Each AP has a circular coverage area of the same size. We considered 350 different configurations, with the constraint that a_0 coverage area overlaps with the ones of all other APs, so that FK is attained when all the 7 neighbours are reported.

We can notice that, given the symmetry introduced on the coverage radii and the constraint on the number of neighbours of a_0, the results are invariant under the coverage radius chosen.
For each of these 350 configurations, the expected time of 0.9-knowledge $\mathbb{E}[\tau]$ has been computed using (17), together with the number of steps to guarantee 0.9-knowledge with 90 % confidence, $S(0.9)$.

In Figure 11 we can observe the empirical distribution function of these two measures. $\mathbb{E}[\tau]$ is centred around 10 steps, while $S(0.9)$ is shifted on higher values, as expected being an upper bound. In Figure 12, the empirical cumulative distribution function of $\mathbb{E}[\tau]$ and $S(0.9)$ is shown. We can see that for 95 % of the samples we can expect to have 0.9-knowledge in about 16 steps, while to have 90 % confidence using the bound obtained in (19), we need to wait about 22 steps.

We can notice that the bound obtained with (19) is a conservative estimation, because it uses only the second biggest eigenvalue $\tilde{\lambda}$, *i. e.* it takes in account of only the slowest way to reach the desired knowledge, while the problem has a rich combinatorial structure that can not be completely captured by (20).

3.3.3.2 *Random Walk on a Grid*

In order to investigate and confirm the ideas of Section 3.3.2.5, we simulated a random walk with reflective boundary on a grid as mobility model, and compared it with Model 1 (see Section 3.3.2), for a set of 8 AP positioned as described at the beginning of this section.

LOW FREQUENCY REPORTS We consider the cases in which a report is made for a number of steps that approaches M^2 (the mixing time of the Markov Chain), where M is the number of nodes in the grid. We can observe that, if the report rate is low enough, the empirical mean time of 0.9-knowledge of the random walk model approaches the one of our simplified model.

If we assume typical femtocell parameters, *i. e.* that the coverage radius is 50 m, and that the user make a step in a grid of 2.5 m every 5 s, then Figure 13 suggests that one report every 50 min is enough to reach mixing time ($M^2 \approx 1300$) with the same number of reports

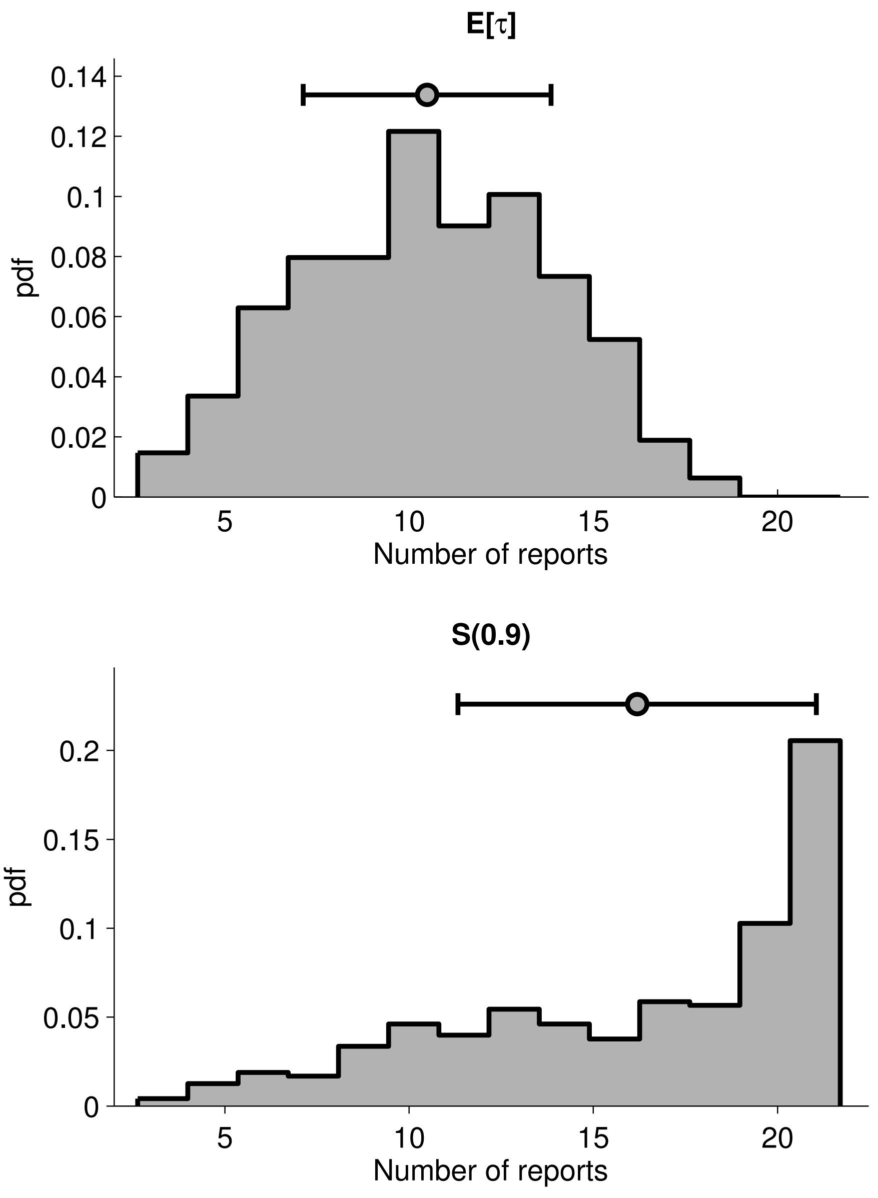


Figure 11: Empirical distribution function of the expected time of 0.9-knowledge, and the number of steps to have 0.9-knowledge with 90 % confidence, for the teleport model on random positioned APs.

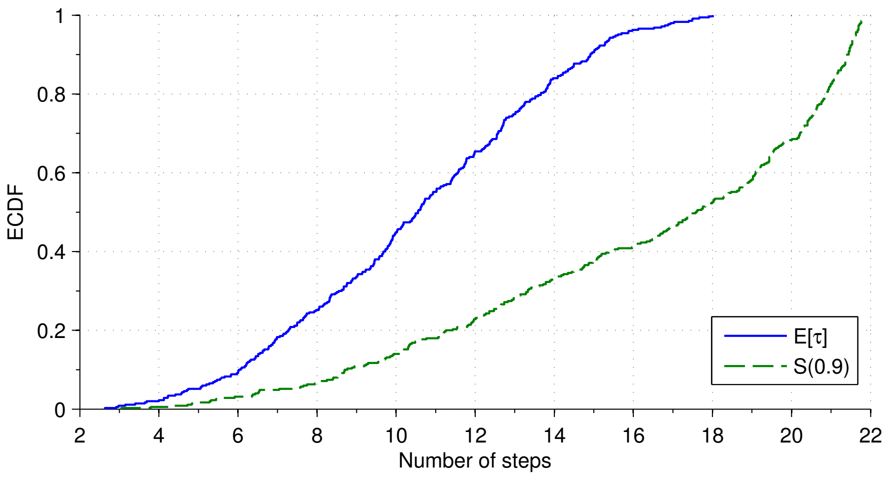


Figure 12: Empirical cumulative distribution function of the expected time of 0.9-knowledge, and the number of steps to have 0.9-knowledge with 90 % confidence, for the teleport model on random positioned APs.

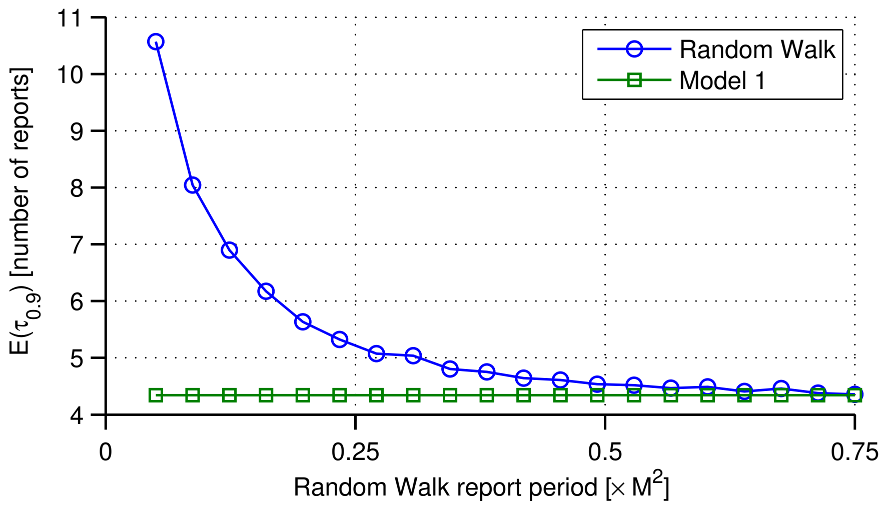


Figure 13: Empirical mean 0.9-knowledge time (in number of reports) of a random walk vs. report period, compared with Model 1.

of the teleport mobility model (Model 1), giving us 0.9-knowledge in less than 5 reports, *i. e.* less than 5 hours.

The reason why more reports are needed in the case of high-frequency reports is the following: since the inter-report time is short, it is likely that many reports will be sent from the same tile, *i. e.* the knowledge chain will undergo many self-transitions. The results thus confirm that Model 1 can effectively be used *at least* to provide an upper bound for τ.

Combining this result with the one from Figure 12, we can conclude we need about 10 steps to have 0.9-knowledge, *i. e.* less than half day. See Section 3.4.1 for an interpretation of these results in terms of implementation.

3.3.3.3 *A Realistic Scenario*

A received power map for 4 APs in the Hynes convention centre has been generated using the Wireless System Engineering (WiSE) [Fortune et al., 1995] software, a comprehensive 3D ray tracing based simulation package developed by Bell Laboratories. APs are assumed

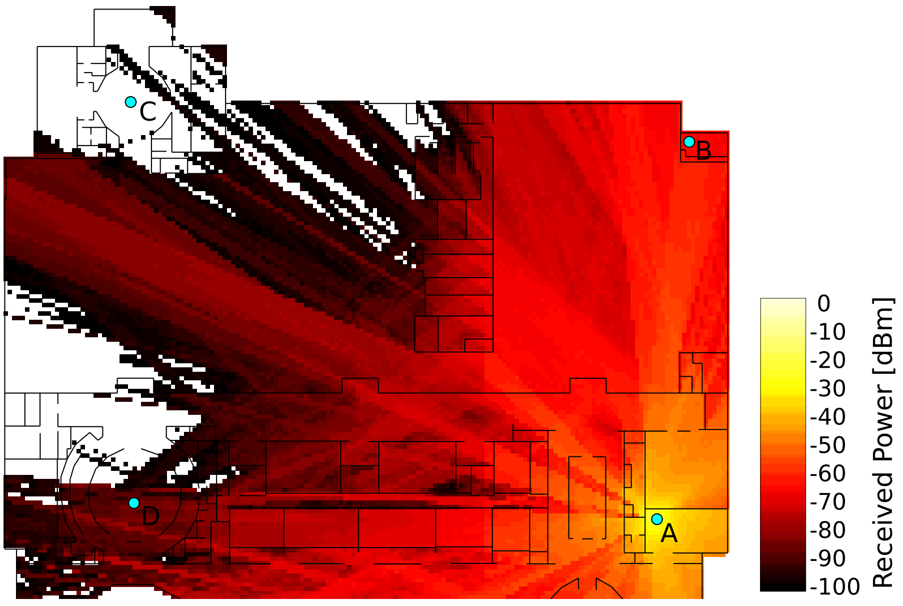


Figure 14: Received power map, in dBm, of Hynes convention centre. AP A is transmitting at 2.1 GHz with a power of 34 mW. Received powers below −100 dBm are not shown. APs B,C and D are not transmitting.

transmitting at a frequency of 2.1 GHz with a power of 34 mW.

In Figure 14 the received power map when only first AP is active is shown. The area shapes are more complex than the simple scenario depicted in Section 3.3.3.1.

In Figure 15, the expected time of δ-knowledge, $\mathbb{E}[\tau_\delta]$, is shown varying δ; we can notice a step shape, where a new step is added every time a new state becomes absorbing, as explained in Section 3.3.2.2.

In Figure 16, the expected time of FK, $\mathbb{E}[\tau_1]$, is shown varying the user detection threshold, from a very conservative value of -60 dBm to a more realistic one of -100 dBm. When the users are more sensitive, the coverage areas, and the higher order tiles in particular, are bigger, leading to better performances. We can see that 14 steps are enough, on average, to get FK in all cases.

These results confirm that the values obtained placing random AP with circular coverage in Section 3.3.3.1 are compatible with real world scenarios.

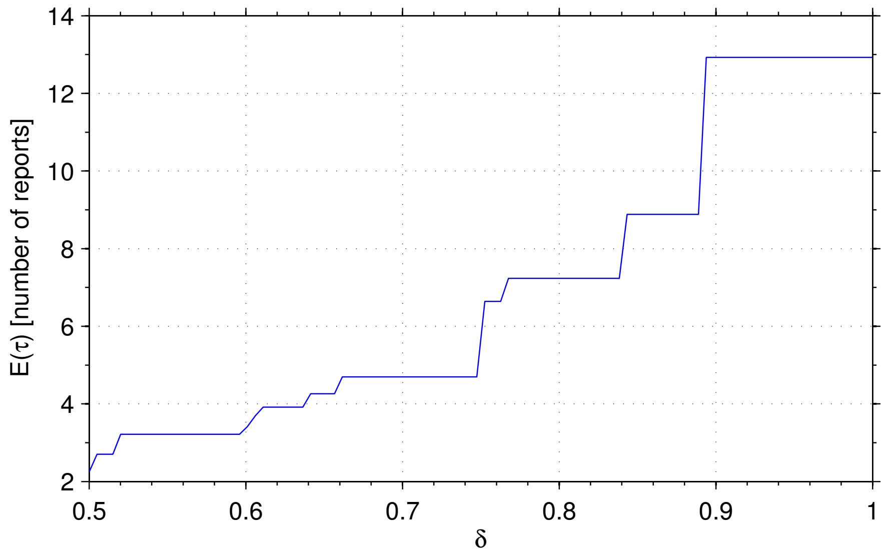


Figure 15: Expected time of δ-knowledge, $\mathbb{E}[\tau_\delta]$ for different values of the parameter δ.

3.4 USE CASES

In this section we present some use cases that are representative of the possible practical applications of the results we have presented so far. Given a particular wireless application, the focus is on whether user reports can effectively be used to efficiently achieve knowledge of the network local topology.

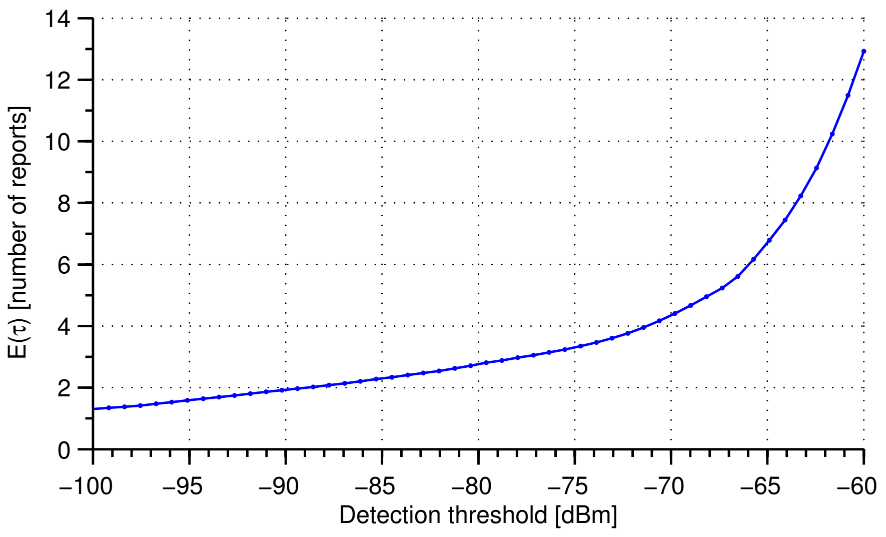


Figure 16: Expected time of FK, $\mathbb{E}[\tau]$, for different values of the user-detection threshold.

In the case of femtocell deployment for residential use, each base station typically serves a very small number of devices. Using data of typical residential densities and coverage areas, a statistic of the tessellation can be devised. If it is possible to establish a time $\tilde{T}$ after which the user position can be considered as drawn from a uniform distribution, then $S(\delta)$ is an upper bound of the time of δ-knowledge for all the inter-report times smaller than or equal to $\tilde{T}$.

Opposite to the previous example, femtocells deployed in congested places like a mall have an extremely large basin of potential users. However, in situations where users main interest is other than connecting to the internet, it is reasonable expecting the single-user reporting-activity to be rather sporadic. Therefore, the Poissonian approximation that we have mentioned at the end of Section 3.3.2.5 may be applicable. In these case, characterising the time to achieve δ-knowledge is possible through a statistic of the typical tessellations.

3.4.1 *Summary*

We introduce the problem of user-reports-based Local Topology Discovery, providing a crisp mathematical formulation of it in the case of Model 1. We show that Model 1 can effectively be used as an upper bound for a wide range of mobility models, when the user reports frequency is lower than the mixing time of the MC of the actual mobility model. In Section 3.3.2.4 we provide an useful method to estimate the time of δ-knowledge when the problem is too big to solve exactly using Equation (17).

Simulations on random scenarios show that the expected number of reports in order to have a high degree of knowledge of the local topology is very small. Roughly speaking, a user moving at $0.5\,\mathrm{m/s}$ according to a random walk model, and providing a report every hour, will guarantee the AP will have 0.9-knowledge with high probability in less than half day. Since the local topology is not typically expected to change every day, this is an acceptable time, especially considering that it is actually possible to send reports on a shorter timescale. The simulations on more realistic scenarios (Section 3.3.3.3) give very similar results in term of time of 0.9-knowledge.

In summary, our results are encouraging, and corroborate the heuristic recommendations in [Edwards, 2008] and [Checco et al., 2012]. In the case of femtocells, implementation of topology discovery should be relatively straightforward because the hardware and the firmware are already capable of generating and managing user reports. Looking ahead, a more extensive study on more realistic scenarios is still required, where the typical topological properties of a urban area are taken in account. Similarly, an analysis of more realistic mobility models is desirable. While the simple case of Model 1 can be used to estimate any other Markovian model (*i. e.* any mobility model that can be described with a Markov process) with unique stationary measure, it is likely to be an accurate estimate only when the report frequency is sufficiently low, namely slower than the mixing time of the Markov Chain.

4

DECENTRALISED COLOURING WITH PARTIAL SENSING

Convergence of the CFL algorithm to a satisfying assignment is only guaranteed by existing results when *all* devices participating in a constraint are able to sense the satisfaction/dissatisfaction of the constraint. This sensing requirement is violated in a number of important practical problems, for example in wireless networks with hidden terminals. In this chapter we show that in fact the CFL algorithm continues to find a proper graph colouring even in the presence of strong sensing restrictions, in particular sensing asymmetry of the type encountered when hidden terminals are present.

Our main analytic contribution is to establish sufficient conditions on the sensing behaviour to ensure that the CFL algorithm finds satisfying assignments with probability one. These conditions take the form of connectivity requirements on the induced sensing graph. These requirements are mild, and we demonstrate that they are commonly satisfied in wireless allocation tasks.

CONTENTS

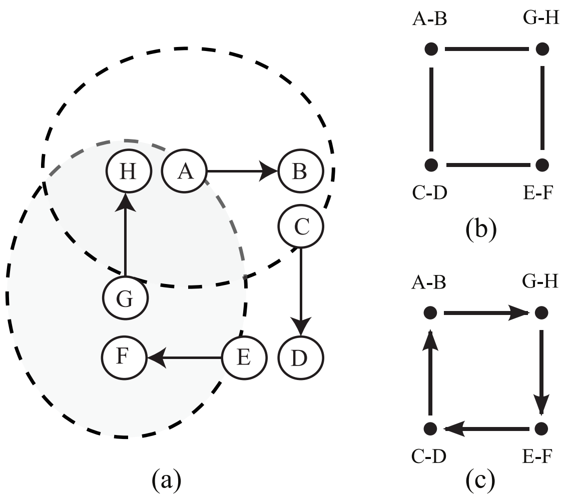


Figure 17: (a) Illustrating a wireless network with asymmetric sensing due to hidden terminals. The shaded areas indicate the interference created by transmitters A and G. Transmissions by A prevent H receiving transmissions by G. However, the converse is not true *i. e.* transmissions by G do not prevent B from successfully receiving transmissions by A. Link $A - B$ can therefore be satisfied while $G - H$ is dissatisfied. Similarly for the other links shown. Associating each edge with a vertex yields graph (b) corresponding to (a) for which a proper colouring is sought. Sensing restrictions then yield the induced oriented graph (c), as explained in Section 4.1.1.

We argue that our results are of considerable practical importance in view of the prevalence of both communication and sensing restrictions in wireless resource allocation problems.

The analysis of stochastic learning algorithms is challenging, and part of the technical contribution is the development of novel analysis tools. We present a number of examples demonstrating the efficacy of CFL-like algorithms when subject to strong sensing as well as communication constraints, and explore the impact of sensing constraints on the rate of convergence.

4.1 COLOURING PROBLEMS WITH SENSING RESTRICTIONS

4.1.1 *Decentralised Solvers*

Definition 9 (Decentralised CP Solver With Sensing Restrictions)**.** A Decentralised CP solver where (10) is replaced with the restriction that for each variable x_i, must select its next value based only on an evaluation of

(D6) $\min_{m \in \mathcal{C}_i} \Phi_m(\vec{x})$, where *information set* $\mathcal{C}_i \subseteq \mathcal{M}_i$ is a subset of edges incoming to node i and we adopt the convention that $\min_{m \in \emptyset} \Phi_m(\vec{x}) = 1$.

Note that despite the sensing restrictions we still require the solver to satisfy (D1) and find a satisfying assignment, *i. e.* for all t sufficiently large

$$\vec{x}(t) = \vec{x}, \text{with} \min_{i \in V} \min_{m \in \mathcal{M}_i} \Phi_m(\vec{x}) = 1.$$

This definition captures sensing restrictions where, for example, a hidden terminal is unable to sense whether or not its transmissions are causing excessive interference to the set of receivers for which it is hidden. In such cases, the variable x_i associated with the hidden terminal can only evaluate $\min_{m \in \mathcal{C}_i} \Phi_m(\vec{x})$ rather than $\min_{m \in \mathcal{M}_i} \Phi_m(\vec{x})$, where $\mathcal{C}_i \subseteq \mathcal{M}_i$ (where equality holds only if sensing restrictions are absent).

It is important to note that an assignment $\vec{x}$ may ensure $\min_{m \in \mathcal{C}_i} \Phi_m(\vec{x}) = 1$, $i \in V$ but might have $\min_{m \in \mathcal{M}_i} \Phi_m(\vec{x}) = 0$ for one or more variables and so need not be satisfying in the absence of sensing restrictions *i. e.* it may not be a proper colouring. We therefore require the following sensing condition in order to satisfy (D1):

Lemma 4.1. *Let $\mathcal{C} := \cup_{i \in V} \mathcal{C}_i$. Suppose that for each pair of edges $i \leftrightarrow j$ in $\mathcal{M}$, at least one directed edge appears in at least one information set C_i for some vertex i i.e. $(i,j) \in \mathcal{M} \implies (i,j)$ or $(j,i) \in \mathcal{C}$. Then an assignment $\vec{x}$ is satisfying with sensing restrictions iff it is satisfying in the absence of sensing restrictions. That is,* $\min_{i \in V} \min_{m \in \mathcal{C}_i} \Phi_m(\vec{x}) = 1 \iff \min_{i \in V} \min_{m \in \mathcal{M}_i} \Phi_m(\vec{x}) = 1$.

Proof. Suppose $\min_{i \in V} \min_{m \in \mathcal{C}_i} \Phi_m(\vec{x}) = 1$. That is, $\Phi_m(\vec{x}) = 1\ \forall m \in \mathcal{C}$. By definition, $\Phi_{(i,j)}(\vec{x}) = \Phi_{(j,i)}(\vec{x})$ and since $(i,j) \in \mathcal{M} \implies (i,j)$ or $(j,i) \in \mathcal{C}$ the result follows.
Conversely, suppose $\min_{i \in V} \min_{m \in \mathcal{M}_i} \Phi_m(\vec{x}) = 1$. Since $\mathcal{C}_i \subseteq \mathcal{M}_i$, the result immediately follows. $\square$

It will be useful to consider oriented partial graph $G' = (V, \mathcal{C})$ induced by the information set $\{\mathcal{C}_1, \dots, \mathcal{C}_N\}$. This graph has the same set V of vertices as graph G for which a proper colouring is sought, but the edges are now defined by the set of *ordered* pairs $(i,j) \in \mathcal{C}$ if $(i,j) \in \mathcal{C}_j$. We say $i \to j$ if there is a directed edge from

i to j, and $i \not\to j$ if there is no edge from i to j. For example, Figure 17(c) gives the graph G' corresponding to Figure 17(b). Here, the directed edge from $A - B$ to $G - H$ indicates that while $G - H$ can sense whether the edge between $A - B$ and $G - H$ is satisfied or not, $A - B$ cannot.

4.1.2 *Examples*

Before proceeding, we briefly demonstrate that several important resource allocation tasks in wireless networks fall within our framework of graph colouring with sensing restrictions.

4.1.2.1 *Channel Allocation With Hidden Terminals*

Consider a network of N wireless links $i = 1, \ldots, N$, each consisting of a transmitter T_i and a receiver R_i. Let P_i denote the transmit power of T_i and γ_{ij} denote the path loss between the transmitter T_i of link i and the receiver R_j of link j. The received power at R_i from T_j is therefore $\gamma_{ji}P_j$. Each link can select one from a set $\mathcal{D} = \{1, \ldots, D\}$ of available channels to use. Link i would like to select a channel in such a way that the signal power impinging on the receiver R_i from other links sharing the same channel is less than a specified threshold Q_i – Q_i may, for example, be selected to ensure that the SINR at R_i is above a target threshold. Each link i can sense that another link j is sharing the same channel when the received power $\gamma_{ji}P_j \geqslant Q_j$ (this might correspond to the minimum interference power that causes decoding errors on the link or to the carrier-sense threshold in 802.11).

To formulate this as a colouring problem, let $\mathcal{D} = \{1, \ldots, D\}$ be the palette of available colours. Associate variable x_i with wireless link i, $i \in \{1, \ldots, N\}$, with the value of $x_i \in \mathcal{D}$ corresponding to the channel selected by link i. Define graph $G = (V, \mathcal{M})$ with $V := \{1, \ldots, N\}$ and set of edges $\mathcal{M}$. Add edge (j, i) to $\mathcal{M}$ whenever the received power $\gamma_{ji}P_j$ from link j at link i is above threshold Q_i when both links select the same channel, $i \neq j$, $i, j \in \{1, \ldots, N\}$. Importantly, whenever an edge (j, i) is in $\mathcal{M}$ we also add edge (i, j) to $\mathcal{M}$, so that G is an undirected graph. A proper colouring of graph G corresponds to a satisfactory channel allocation *i.e.* $\gamma_{ji}P_j > Q_i$, for all $i \in \{1, \ldots, N\}$ and all j such that $x_j = x_i$ and $j \in \{1, \ldots, N\}$.

Now define graph $G' = (V, \mathcal{C})$ with edge $(j, i) \in \mathcal{C}$ when the received power $\gamma_{ji}P_j$ from link j at link i is above threshold Q_i when both links select the same channel. Note that, unlike for graph G, we do *not* also add edge (i,j) to $\mathcal{C}$ unless $\gamma_{ij}P_i > Q_j$ when both links select the same channel. Observe that the edges in graph G' embody the sensing abilities of links, and in general $\mathcal{C} \neq \mathcal{M}$ and so $G' \neq G$.

Note that we can readily generalise this formulation to include, for example, the selection of multiple channels/sub-carriers by each link and to allow multiple transmitters and receivers in a link (which might then correspond to a WLAN).

4.1.2.2 *Decentralised TDMA Scheduling with Hidden Terminals*

When using a time division access scheme, wireless networks need to have a schedule for accessing the channel. This schedule can be decided in a centralised manner, but it is possible to require a decentralised way of solving the problem. The classical CSMA/CA approach to decentralised scheduling does not yield convergence to a single schedule and leads to continual collisions. Recently, there has been interest in decentralised approaches for finding collision-free schedules [Fang et al., 2010]. Consider a wireless network with N links, $i = 1, \ldots, N$. Time is slotted and partitioned into periodic schedules on length $T \geqslant N$ slots. The transmitter on each link would like to select a slot that is different from the choice made by other transmitters if their collisions would collide (transmissions in the same slot need not collide when, for example, the two transmitters are located sufficiently far apart). A link is able to sense whether its transmission in a slot was successful or not.

To formulate this as a colouring problem, let $\mathcal{D} = \{1, \ldots, D\}$ be the set of available time slots in the periodic schedule. Associate variable x_i with link i, $i \in \{1, \ldots, N\}$, with the value of $x_i \in \mathcal{D}$ corresponding to the slot selected by the transmitter of link i. Define graph $G = (V, \mathcal{M})$ with $V := \{1, \ldots, N\}$ and set of edges $\mathcal{M}$. Add edge (j,i) to $\mathcal{M}$ whenever simultaneous transmissions by the transmitters of links i and j would lead to failure of the transmission by i. Whenever an edge (j,i) is in $\mathcal{M}$, also add edge (i,j) to $\mathcal{M}$. A proper colouring of graph G corresponds to a non-colliding schedule.

Define graph $G' = (V, \mathcal{C})$ with edge $(j,i) \in \mathcal{C}$ when simultaneous transmissions by the transmitters of links i and j would lead to failure of the transmission by j. Unlike for graph G, we do not also add edge (i,j) to $\mathcal{C}$ unless simultaneous transmissions by transmitters i and j would lead to failure of the transmission by j. Once again, the edges in graph G' embody the sensing abilities of links and in general $\mathcal{C} \neq \mathcal{M}$.

4.2 SOLVING COLOURING PROBLEMS WITH SENSING RESTRICTIONS

We now investigate in which conditions the CFL algorithm is still a decentralised CP solver, adding the constraint (D6) of partial sensing.

4.2.1 *Algorithm*

We rewrite the CFL algorithm, with the only difference here of envisaging sensing restrictions. An instance of this algorithm is run in parallel for every variable.

Algorithm 2 Communication-Free Learning with sensing restrictions.

```
1: Initialise $p_{i,j} = 1/D, j \in \{1, \dots, D\}$.
2: Realise a random variable, selecting $x_i = j$ with probability $p_{i,j}$.
3: loop
4:    Evaluate $\min_{m \in \mathcal{C}_i} \Phi_m(\vec{x})$, returning *satisfied* if its value is $1$, and
      *unsatisfied* otherwise.
5:    Update: If *satisfied*,
          $$p_{i,j} = \begin{cases} 1 & \text{if } j = x_i \\ 0 & \text{otherwise.} \end{cases}$$
      If *unsatisfied*,
          $$p_{i,j} = \begin{cases} (1-b)p_{i,j} + a/(D-1+a/b) & \text{if } j = x_i \\ (1-b)p_{i,j} + b/(D-1+a/b) & \text{otherwise,} \end{cases}$$
      where $a, b \in [0, 1]$ are design parameters.
6:    Realise a random variable, selecting $x_i = j$ with probability $p_{i,j}$.
7: end loop
```

The only difference from Algorithm 2, is that each variable has an information set that can be smaller than $\mathcal{M}_i$.

In the examples in this work we select $a = b = 0.1$, and do not optimise these values to particular settings, because the choice of optimal parameter depends heavily on the topological properties of the graph, and should be done when a specific class of problem is tackled, or an adaptive mechanism should be devised.

In order to be a decentralised CP solver with sensing restrictions, Algorithm 2 must satisfy conditions $(D1) - (D6)$. We can see immediately that Algorithm 2 satisfies $(D2) - (D6)$.

(D3)-(D6) By construction, the only information used by the algorithm is $\min_{m \in \mathcal{C}_i} \Phi_m(\vec{x})$ in Step 4 and thus it satisfies the criteria (D3), (D4), (D5) and (D6).

(D2) Algorithm 2 also satisfies the (D2) criterion that it sticks with a solution from the first time one is found. To see this, note that the effect of Step 5 is that if a variable experiences success in all clauses Φ that it participates in it continues to select the same value with probability 1. Thus if all variables are simultaneously satisfied in all clauses, *i.e.* if $\min_{m \in \mathcal{C}_i} \Phi_m(\vec{x})$, then the same assignment will be reselected indefinitely with probability 1.

It remains to verify satisfaction of $(D1)$, *i.e.* convergence of the algorithm to a satisfying assignment, which is the subject of the next section.

4.2.2 *Convergence Analysis*

Recall the following definition:

Definition 10 (Strongly Connected Graph)**.** A path of length q in oriented graph $G' = (V, \mathcal{C})$ is a sequence $\mu = (u_1, u_2, \dots, u_q)$ of edges in $\mathcal{C}$ such that the terminal endpoint of edge u_i is the initial endpoint of edge u_{i+1} for all $i < q$. Oriented graph $G' = (V, \mathcal{C})$ is strongly connected if it contains a path starting in x and ending in y, for each pair of distinct vertices $x \neq y \in V$.

We now state our main analytic result:

Theorem 4.1. *Consider any satisfiable colouring problem with graph* $G = (V, \mathcal{M})$ *and information sets* $\{\mathcal{C}_1, \dots, \mathcal{C}_N\}$*. Suppose:*

(A) *At least one half of each undirected edge* $i \leftrightarrow j$ *in* $\mathcal{M}$ *appears in at least one information set* C_i *for some vertex* i, i.e. $(i, j) \in \mathcal{M} \implies (i, j)$ *or* $(j, i) \in \mathcal{C}$;

(B) *The induced graph* $G' = (V, \mathcal{C})$ *is strongly connected.*

Then with probability greater than $1-\varepsilon \in (0,1)$, *the number of iterations for Algorithm* 2 *to find a satisfying assignment is less than*

$$(N^3)\exp(N^4\log(\gamma^{-1}))\log(\varepsilon^{-1}) \text{ where } \gamma = \frac{\min(a,b)}{D-1+a/b}.$$

Proof. See Appendix. □

As Theorem 4.1 covers any arbitrary CP that admits a solution, for any given instance these bounds are likely to be loose. They do, however, allow us to conclude the following corollary proving that if a solution exists, Algorithm 2 will almost surely find it:

Corollary 4.1. *For any colouring problem that admits a proper colouring and that fulfills conditions (A) and (B), Algorithm* 2 *will find a proper colouring in almost surely finite time.*

Intuitively, we expect that sensing restrictions may increase the time it takes to find a satisfying assignment. When $\mathcal{C}_i = \mathcal{M}_i$, $i \in V$ (perfect sensing) then $\mathcal{C} = \mathcal{M}$ and our analysis yields the following bound on the convergence rate:

Corollary 4.2. *When* $\mathcal{C}_i = \mathcal{M}_i \forall i \in V$, *with probability greater than* $1-\varepsilon \in (0,1)$, *the number of iterations for Algorithm* 2 *to find a satisfying assignment is less than*

$$(N)\exp(\frac{N(N+1)}{2}\log(\gamma^{-1}))\log(\varepsilon^{-1}).$$

Proof. See Appendix. □

That is, our upper bound on convergence rate is improved from N^4 to N^2 with perfect sensing. This corresponds to the bound found in [Duffy et al., 2013] (see Section 3.1.3.1) for generic Decentralised Constraint Satisfaction (DCS) problems, but it is looser than the refined bound found there for graph colouring problems (Theorem 3.1). However, it is important to stress that this observation comes with the caveat that, as already noted, we believe both of these bounds are extremely loose. Hence, we revisit this question below using numerical simulations, which yield tight measurements of convergence rate.

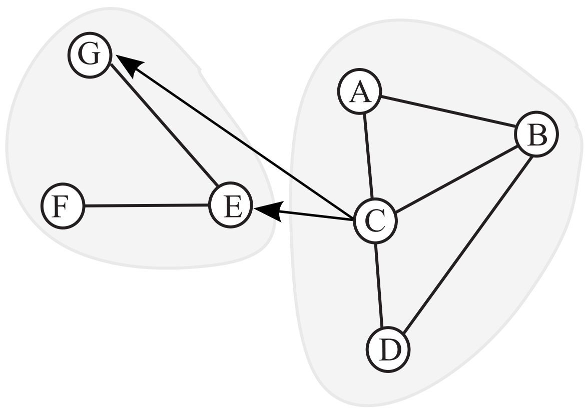


Figure 18: Example of a graph G with two strongly connected components ($\{A, B, C, D\}$ and $\{E, F, G\}$) which are sparsely interconnected. The chromatic number $\chi(G)$ of graph G is 4, the chromatic numbers of the connected components are 3 and 2 respectively.

4.2.3 *Relaxing Strong Connectivity Requirement*

The requirement in Theorem 4.1 for the sensing graph G' to be strongly connected can be relaxed in a number of ways. If graph G is not connected, we only have to ask for strong connectivity separately for the induced graph corresponding to each connected component. More generally, we can extend our analysis to situations where graph G consists of a number of strongly connected components with sufficiently sparse interconnections between these components.

To help gain insight, consider the example graph G shown in Figure 18. Graph G consists of two strongly connected components, $\{A, B, C, D\}$ and $\{E, F, G\}$, with two directed edges between them. Subgraph $\{A, B, C, D\}$ has no incoming edges and can be coloured on its own (*i. e.* without reference to the rest of graph G). Component $\{E, F, G\}$ has two incoming edges. Observe that these can be thought of as, in the worst case, reducing by two the number of colours available in our palette $\mathcal{D}$ when colouring $\{E, F, G\}$. Now, $\mathcal{D}$ contains at least $\chi(G) = 4$ colours (since we assume colouring of graph G is feasible) while subgraph $\{E, F, G\}$ is colourable using only two colours. Hence, regardless of the colours of vertices C and D on the two incoming edges, sufficient colours are always available to colour subgraph $\{E, F, G\}$. We formalize these observations in Theorem 4.2.

Definition 11 (Subgraph of G' generated by V_k)**.** The subgraph of graph $G' = (V, \mathcal{C})$ generated by V_k is the graph $(V_k, \{(i, j) : i, j \in V_k, (i, j) \in \mathcal{C}\})$. That is, the graph with V_k as its vertex set and with all the arcs in G' that have both their endpoints in V_k. With a slight abuse of notation, we will identify the subgraph with the vertex set V_k that generates it.

Definition 12 (In-degree of a subgraph)**.** The in-degree of the subgraph graph $G' = (V, \mathcal{C})$ generated by V_k, denoted by $\mathrm{deg}(V_k)$, is the number of vertices $j \in V \setminus V_k$ that have at least one edge $(i, j) \in \mathcal{C}$, $j \in V_k$ ending in V_k.

Theorem 4.2. *Let $V = \bigcup_{k=1}^{p} V_k$, $V_i \bigcap V_j = \emptyset$ be a partition of the vertex set V of oriented graph $G' = (V, \mathcal{C})$ such that (i) the subgraph generated by V_k, $k \in \{1, \dots, p\}$ is strongly connected and (ii) the subgraph generated by the union $\cup_{k \in S} V_k$ of any subset $S \subset \{1, \dots, p\}$ is not strongly connected. That is, directed edges may exist between strongly connected components, but their union is not strongly connected. Let D be the number of colours available in our palette $\mathcal{D}$ and let $\chi(V_k)$ be the chromatic number of the (undirected) subgraph of $G = (V, \mathcal{M})$ generated by V_k. Suppose that*

$$\chi(V_k) \leqslant D - \mathrm{deg}(V_k),\ k = 1, \dots, p \tag{23}$$

Then for any colouring problem that admits a proper colouring and that fulfills condition (B) of Theorem 4.1, Algorithm 2 will find a proper colouring in almost surely finite time.

Proof. The main idea is that if a strongly connected component V_k requires less colours than D to be coloured, and if the number of edges entering in V_k is small enough, as shown in Equation (23) and in Figure 18, then V_k can be coloured by Algorithm 2 even if some vertices $j \notin V_k$ are not reachable by any $i \in V_k$, with $i \leftarrow j$. The original colouring problem is satisfiable by hypothesis, so we have at least $\chi(G)$ available colours D in our palette. We need to consider two cases. Case 1: $\mathrm{deg}(V_k) = 0$. Since $\chi(V_k) \leqslant \chi(G)$ (since V_k is a subgraph of G), at least one proper colouring of subgraph V_k exists and we can use Theorem 4.1 to establish that Algorithm 2 will almost surely find a proper colouring. Case 2: $\mathrm{deg}(V_k) > 0$. The incoming edges reduce by at most $\mathrm{deg}(V_k)$ the choice of the colours available for subgraph V_k. Hence, provided $\chi(V_k) \leqslant D - \mathrm{deg}(V_k)$ then we can apply Theorem 4.1 to subgraph V_k in isolation from the rest of graph G to establish that Algorithm 2 will almost surely find a proper colouring. □

4.3 PERFORMANCE ON RANDOM GRAPHS

The upper bound in Theorem 4.1 is a worst case bound, and in addition we believe that it may not be tight. Hence, it is important to also evaluate the performance of Algorithm 2 using numerical measurements. In this section we present measurements for a class of random graphs that are based on an idealised model of wireless network interference. These graphs have been widely studied [Dousse, 2012] and provide a method for technology-neutral evaluation. In Section 4.4 we evaluate performance in a technology specific manner using graphs derived from the Wireless Geographic Logging Engine (WiGLE) database of 802.11 hot spot locations.

4.3.1 *Random Graph Model*

We use realizations drawn from the Directed Boolean Model (DBM) described in [Dousse, 2012]. The vertices of the graph are drawn from a Poisson point process in $[0,1]^2$ with intensity λ (with appropriate re-scaling to a required area – in the examples here we re-scale to an area of $100\,\mathrm{m}^2$). In the original undirected Boolean model (also known as the blob model [see Grimmett, 1999, Section 10.5]), each vertex is the center of a closed ball of random radius. The radii of the balls are independently and identically distributed. The (undirected) connectivity graph is obtained by adding an edge between all pairs of points whose balls overlap, *i. e.* $B(y) \cap B(z) \neq \emptyset$, where $B(y), B(z)$ denote the balls centered on vertices y, z respectively. To obtain a directed graph, following [Dousse, 2012] we slightly change the above rule and put a directed edge between y and z if $z \in B(y)$ and an edge between z and y if $y \in B(z)$. This modified model is referred to as the Directed Boolean Model (DBM).

In our measurements the radii are chosen uniformly at random from the finite set of the coverage areas corresponding to transmitting powers in the range 12 dBm to 20 dBm, with steps of 2 dBm, and a specified detection threshold R. We use the 3GPP path loss model for indoor environments [3GPP TS25.101, 2004], based on the Okumura-Hata log-distance model

$$\mathrm{PL}_{\mathrm{dB}}(d) = 43.3 \cdot \log_{10} d + 11.5 + 20 \cdot \log_{10} f$$

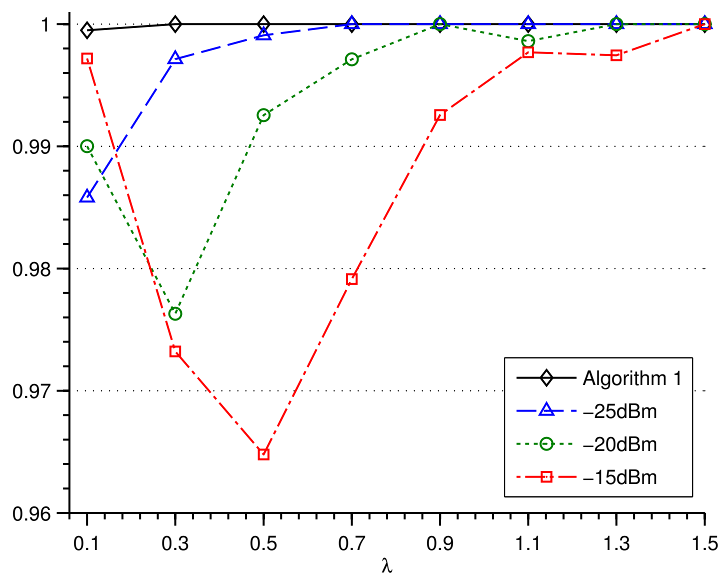


Figure 19: Fraction of DBM graphs nodes satisfying connectivity requirements of Theorem 4.2 versus the detection threshold. Additionally, the fraction of nodes correctly coloured by Algorithm 2 for detection threshold of -25 dBm is shown.

where d is the distance in meters and f is the frequency in GHz. In the examples here we select fixed frequency $f = 2.412$ GHz. For detection threshold R in dB and transmit power P in dB, the coverage radius is then given by

$$d : PL_{dB}(d) + P \geqslant R$$

Figures 18 and 20 show examples of graph generated using this model.

We focus in the most challenging cases by selecting the number D of available colours equal to the minimum feasible value $\chi(G)$.

4.3.2 *Meeting Connectivity Requirements*

Theorems 4.1 and 4.2 place connectivity requirements on the induced sensing graph G' in order to ensure that Algorithm 2 converges to a satisfying assignment. We begin by evaluating the fraction of random graphs in the Directed Boolean Model (DBM) that meet these requirements. Figure 19 plots this fraction for a range of detection thresholds R and vertex densities λ. It can be seen that for detection thresholds below -15 dBm greater than 96 % of graphs satisfy the connectivity requirements. Figure 20 shows some examples of some DBM graphs corresponding to a -25 dBm threshold. Observe that they consist of a number of connected components and so the relaxed connectivity

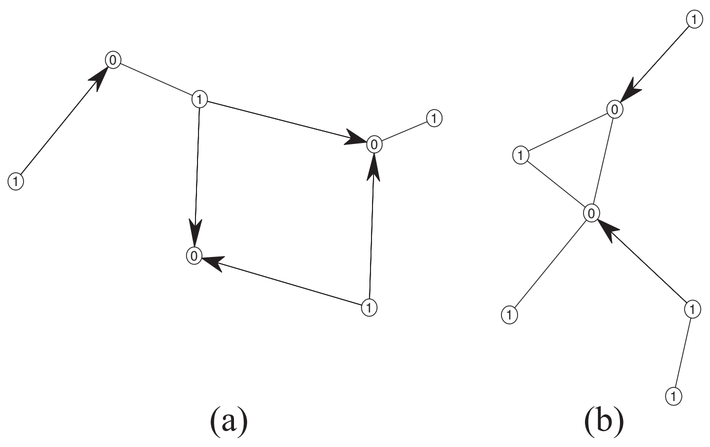


Figure 20: Example DBM graphs. The nodes labeled with 1 are the one that satisfy the connectivity conditions of Theorem 4.2.

conditions provided by Theorems 4.2 are of considerable importance here. Note also that modern wireless devices typically have a noise floor of less than −70 dBm and so −25 dBm is conservative.

Moreover, Figure 19 shows the measured fraction of vertices for which Algorithm 2 successfully found a satisfying assignment for a detection threshold of −25 dBm. It can be seen that greater than 99.9% of vertices are successfully coloured by the algorithm. For $\lambda = 0.5$ and detection threshold of −15 dBm, 4% of the vertices that do not fulfill the conditions of Theorem 4.2 are still correctly coloured by Algorithm 2. This small gap can be explained with the fact that the conditions of Theorem 4.2 are sufficient, but not necessary for convergence: some topologies can lead to convergence for their particular structure or because of a fortunate initial condition (see Figure 20 for some examples).

4.3.3 *Convergence Rate*

Figure 21 shows the empirical PDF of convergence time for Algorithm 2 versus the detection threshold used for sensing. For a threshold of −25 dBm, the mean convergence time is less than 2000 iterations. When the required threshold is increased to −15 dBm, the mean convergence time decreases to less than 1000 iterations. These measurements are for a link density of $\lambda = 0.5$, corresponding to on average 50 wireless links in an area of $100\,\text{m}^2$. Recall that we selected the number D of available colours equal to the minimum feasible $\chi(G)$, thereby focussing on the most challenging situations. For

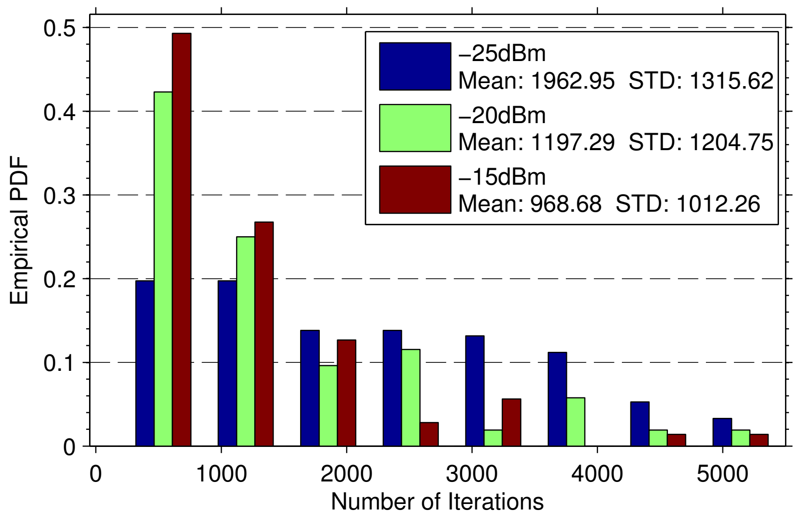


Figure 21: Measured convergence rate of Algorithm 2 for DBM graphs using a number of available colours equal to the chromatic number χ of the graph for three different detection thresholds. The density is $\lambda = 0.5$.

larger numbers of colours it can be verified experimentally that the convergence time decreases exponentially in the number of colours above $\chi(G)$.

The comparison of the bounds given by Theorem 4.1 with the case without sensing restrictions given by Corollary 4.2 suggests that sensing restrictions lead to an increase in the convergence time. This is indeed the case, as shown in Figure 22, where the convergence rate of Algorithm 2 is shown with and without sensing restrictions for DBM graphs with $\lambda = 0.5$ and detection threshold of $-15\,\text{dBm}$. However for DBM graphs it can be seen that this increase is small.

We also analyzed in Section 4.4.1 the impact of the number of available colours on the convergence time.

4.4 CASE STUDY: MANHATTAN WIFI HOT SPOTS

From the online database WiGLE [wig, 2010] we obtained the locations of WiFi wireless Access Points (APs) in an approximately $150\,\text{m}^2$ area at the junction of 5th Avenue and 59th Street in Manhattan[1]. This space contains 81 APs utilizing the IEEE 802.11 wireless standard. We model radio path loss with distance as d^α, where d is the distance in meters and $\alpha = 4.3$ is the path loss exponent (consistent with

1 The extracted (x,y,z) coordinate data used is available online at www.hamilton.ie/net/xyz.txt

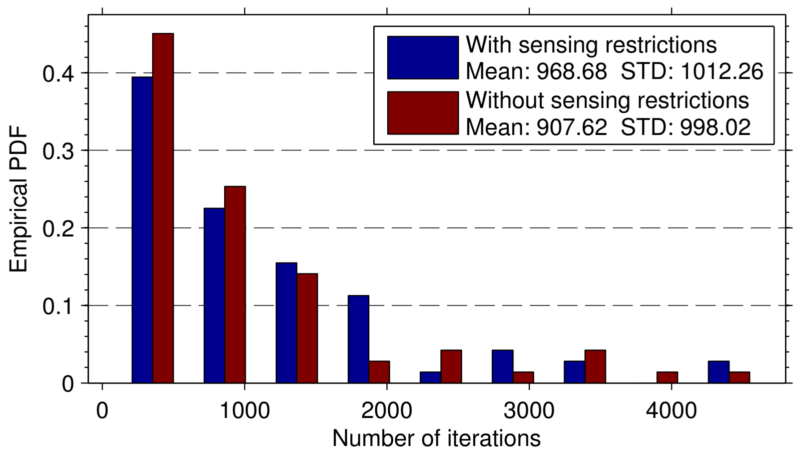


Figure 22: Measured convergence rate of Algorithm 2 for DBM graphs with detection threshold of −15 dBm and density of $\lambda = 0.5$, with and without sensing restrictions.

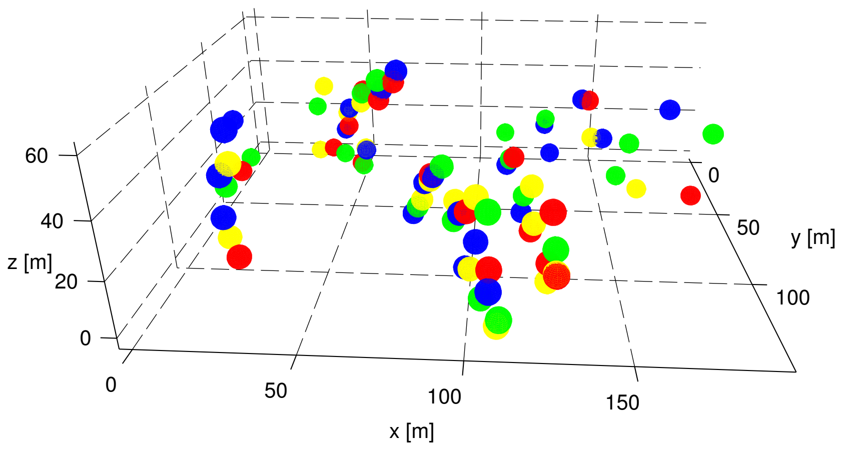


Figure 23: Example assignment for Manhattan WiFi hot spots. Wireless access points are indicated by points and the colour indicates the radio channel selected by the AP.

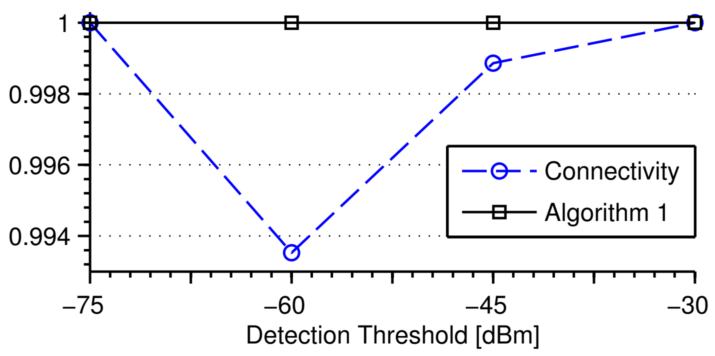


Figure 24: Connectivity for Manhattan WiFi hot spots. Each measurement represents the fraction of nodes that satisfy connectivity requirements of Theorem 4.2. The fraction of nodes correctly coloured by Algorithm 2 is also shown.

the 3GPP indoor propagation model [3GPP TS25.101, 2004]), and the AP transmit powers are selected uniformly at random in the range 12–20 dBm, with steps of 2 dBm. The aim of each AP is to select its radio channel in such a way as to ensure that it is different from nearby WLANs. This can be written as a colouring problem with $N = 81$ APs and N variables x_i corresponding to the channel of AP i, $i = 1, \ldots, N$. As per the 802.11 standard [802, 1997] and FCC regulations, each AP can select from one of 11 radio channels in the 2.4 GHz band and so the x_i, $i = 1, 2, \ldots, N$ take values in $\mathcal{D} = \{1, 2, \ldots, 11\}$. To avoid excessive interference each AP requires that the received signal strength from other APs sharing the same channel is attenuated by at least -60 dB. When all APs use the maximum transmit power of 18 dBm allowed by the 802.11 standard, this requirement is met when the received power is less than -45 dBm and ensures that the SINR is greater than 20 dB (sufficient to sustain a data rate of 54 Mbps when the connection is line of sight and channel noise is Gaussian [Goldsmith, 2005]).

The APs do not belong to a single administrative domain and so a decentralised solver is required. The presence of hidden terminals means that the solver must find a satisfying solution while subject to sensing asymmetry.

The connectivity requirement of Theorem 4.2 was observed to be satisfied $> 99\%$ of examples, see Figure 24.

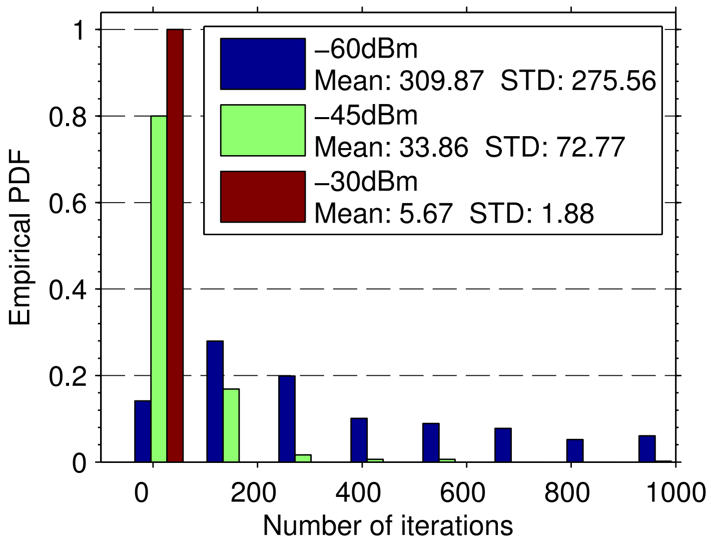


Figure 25: Measured convergence rate of Algorithm 2 for Manhattan WiFi hot spots using a number of available colours equal to the chromatic number χ of the graph.

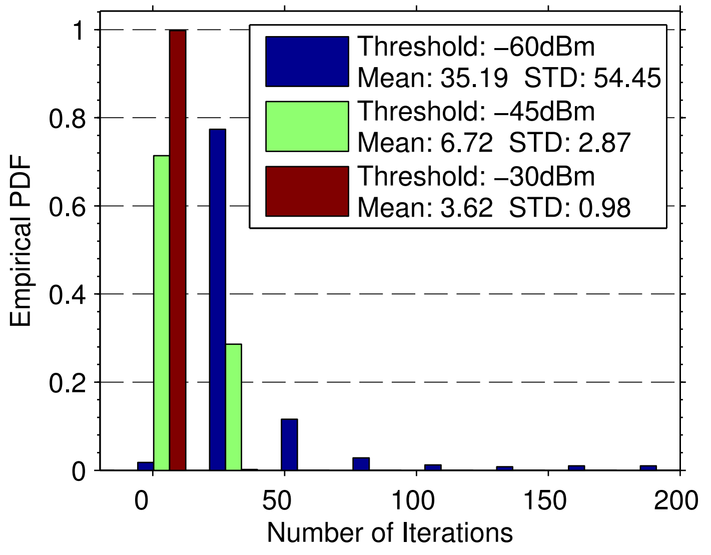


Figure 26: Measured convergence rate of Algorithm 2 for Manhattan WiFi hot spots using a number of available colours equal to $\chi + 2$, where χ is the chromatic number of the graph.

4.4.1 *Convergence Time*

Algorithm 2 was observed to converge in less than 1000 iterations in all examples. Figure 25 shows the measured distribution of convergence time for Algorithm 2 versus the detection threshold used for sensing. For a threshold of −45 dBm, corresponding to the target requirement noted above, the mean convergence time is less than 34 iterations. In a prototype lab set-up we have shown that an update interval of less than 10 seconds is feasible on current 802.11 hardware. Thus the mean time to convergence is under 6 minutes, which is a reasonable time-frame for practical purposes. When the required threshold is increased to −30 dBm, the mean convergence time decreases to less than 6 iterations *i. e.* under 1 minute when each iteration takes 10 seconds.

To examine the impact of the number of available colours we compare, in Figures 25 and 26, the convergence time when the number of colours is equal to χ and $\chi + 2$ respectively. For a detection threshold of −60 dBm, adding two colours reduces the mean convergence time of almost 10 times.

4.5 SUMMARY

We constructively establish the existence of decentralised learning-based solvers that are able to find satisfying assignments even in the presence of sensing restrictions, in particular sensing asymmetry of the type encountered when hidden terminals are present. Our main analytic contribution is to establish sufficient conditions on the sensing behaviour to ensure that the solvers find satisfying assignments with probability one. These conditions take the form of connectivity requirements on the induced sensing graph. These requirements are mild, and we demonstrate that they are commonly satisfied in wireless allocation tasks. We explore the impact of sensing constraints on the speed which a satisfying assignment is found, showing the increase in convergence time is not significant in common scenarios.

Our results are of considerable practical importance in view of the prevalence of both communication and sensing restrictions in wireless resource allocation problems. The class of algorithms analysed here requires no message-passing whatsoever between wireless devices, and we show that they continue to perform well even when devices are only able to carry out constrained sensing of the surrounding radio environment.

4.A APPENDIX – PROOFS

We will exhibit a lower bound for the probability of a sequence of events that ultimately lead to an increase in the number of properly coloured vertices. Such a sequence can be quite complicated in cases where a vertex i is unsatisfied by a vertex j such that $i \not\to j$ (asymmetric sensing), because in this case it is necessary to propagate the dissatisfaction to j via another path.

We can divide the colouring process in two distinct phases: during the first, all unsatisfied vertices select their target colouring. This phase is repeated as long as new vertices become unsatisfied, see Lemma 4.3.

The second phase is needed when a vertex i is unsatisfied by a vertex j such that $i \not\to j$. In this case vertex i will propagate the dissatisfaction to j via another path, and do so in a way that allows the rest of the vertices in the path to restore their colour before this second phase started (see Lemma 4.7). Ultimately, the second phase will only change the colour of one vertex, namely j. After the second phase is executed, it may be necessary to run first phase again, and repeat the two phases until convergence. Both phases are constructed to have a strictly increasing number of vertices with target colour after the phase is executed. We explain the whole sequence after proving some intermediate lemmas and after some clarification on the notation.

Consider graph $G' = (V, \mathcal{C})$. Let

$$A = \bigcup \left\{ \vec{x} \colon \Phi_m(\vec{x}) = 1 \text{ for all } m \in \mathcal{C} \right\},$$

denote the set of assignments which are absorbing for Algorithm 2 and

$$B = \bigcup \{ \vec{x} \colon x_i \neq x_j \text{ for all } i \leftrightarrow j \},$$

the set of proper colourings, with $A \supseteq B$. Under condition (A) of Theorem 4.1, $A = B$ and all absorbing assignments are also satisfying. When the colouring problem is feasible 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 $F_{\vec{x}(t)}$ denote the set of vertices that have their target colour, *i.e.* $F_{\vec{x}(t)} = \{i \colon i \in V, x_i(t) = a_i\}$. Furthermore, let $U_{\vec{x}(t)}$ denote the set of unsatisfied vertices, *i.e.* $U_{\vec{x}(t)} = \{i \colon i, j \in V, x_j(t) = x_i(t), j \to i\}$, where $i \to j$ and $j \leftarrow i$ denote the existence of an oriented edge $(i, j) \in \mathcal{C}$. Define $\gamma = \min(a, b)/(D - 1 + a/b)$.

Lemma 4.2. *If a vertex is unsatisfied, when using Algorithm 2 the probability that the vertex chooses any colour j at the next step is greater than or equal to γ.*

Proof. This follows from step 5 of Algorithm 2. □

Lemma 4.3. *Given any satisfiable CP and an information set $\{\mathcal{C}_1, \dots, \mathcal{C}_N\}$ with starting unsatisfied assignment $\vec{x}(0) \in \mathcal{D}^N, \vec{x}(0) \notin A$ such that $F_{\vec{x}(0)} \not\supseteq U_{\vec{x}(0)}$, Algorithm 2 will reach an assignment $\vec{x}(\tilde{t})$ such that $F_{\vec{x}(\tilde{t})} \supsetneq F_{\vec{x}(0)}$ and $F_{\vec{x}(\tilde{t})} \supseteq U_{\vec{x}(\tilde{t})}$ in $\tilde{t} \leqslant |F_{\vec{x}(\tilde{t})}| - |F_{\vec{x}(1)}| \leqslant N$ steps with probability greater than*

$$\gamma^{\sum_{k=|F_{\vec{x}(0)}|}^{|F_{\vec{x}(\tilde{t})}|} k} > \gamma^{N(N+1)/2}$$

In other words, all vertices that had their target colour in $\vec{x}(0)$ will still have it in $\vec{x}(\tilde{t})$, and all unsatisfied vertices in $\vec{x}(\tilde{t})$ will have their target colour.

Proof. At the first step we consider the event that changes the assignment to

$$x_i(1) = \begin{cases} a_i & \text{if } i \in U_{\vec{x}(0)}, \\ x_i(0) & \text{otherwise.} \end{cases} \tag{24}$$

This event is feasible since Algorithm 2 ensures that all satisfied vertices will remain unchanged and each unsatisfied vertex may change its colour. The probability that this event happens is greater than $\gamma^{|U_{\vec{x}(0)}|}$. After this step we have $F_{\vec{x}(1)} = F_{\vec{x}(0)} \cup U_{\vec{x}(0)}$. Now, the set of unsatisfied variables $U_{\vec{x}(1)}$ could have changed. If $U_{\vec{x}(1)} \subseteq F_{\vec{x}(1)}$, we have finished, otherwise we consider again the event that changes the assignment similarly to equation (24), *i.e.* at generic step t we have

$$x_i(t) = \begin{cases} a_i & \text{if } i \in U_{\vec{x}(t-1)}, \\ x_i(t-1) & \text{otherwise.} \end{cases}$$

The probability of this happening is greater than $\gamma^{|U_{\vec{x}(t-1)}|}$, and it can be lower bounded by $\gamma^{|F_{\vec{x}(t)}|}$ because $F_{\vec{x}(t)} = F_{\vec{x}(t-1)} \cup U_{\vec{x}(t-1)}$. Since while $U_{\vec{x}(t-1)} \nsubseteq F_{\vec{x}(t-1)}$ we have $F_{\vec{x}(t)}$ is a strictly growing set, and we have a finite number of vertices N, a finite time $\tilde{t} \leqslant N$ exists after which we will necessarily have $U_{\vec{x}(\tilde{t})} \subseteq F_{\vec{x}(\tilde{t})}$. The worst case in regards to the number of steps is when at each step, only one new vertex is added to $F_{x(t)}$, giving us the bound for the number of steps of $|F_{x(\tilde{t})}| - |F_{x(1)}|$. □

Lemma 4.4. *Consider any satisfiable CP and an information set $\{\mathcal{C}_1, \dots, \mathcal{C}_N\}$ with induced graph $G' = (V, \mathcal{C})$ and colour $x_i(t) \in \mathcal{D}$ associated with each vertex $i \in V$ at time t. Let $A \subset \mathcal{D}^{|V|}$ denote the set of satisfying assignments. Suppose $|V| > 1$, $\vec{x}(0) \notin A$ (the initial choice of colours is not a satisfying assignment) and graph G' is strongly connected. Let $a \in A$ be an arbitrary satisfying assignment. If $F_{\vec{x}(0)} \supseteq U_{\vec{x}(0)}$, there exists a pair of vertices k, j with same colour such that $j \to k$ and $k \not\to j$, with $k \in U_{\vec{x}(0)}$ and $j \notin U_{\vec{x}(0)}, j \notin F_{\vec{x}(0)}$; in other words, a vertex k exists that is unsatisfied by a satisfied vertex j that doesn't have final colour, and the minimum path length between k and j is greater than 1.*

Proof. Consider any unsatisfied vertex $k \in U_{\vec{x}(0)}$. At least one such vertex exists because $\vec{x}(0) \notin A$. The hypothesis $F_{\vec{x}(0)} \supseteq U_{\vec{x}(0)}$ ensures $x_k(0) = a_k$. Since k is unsatisfied, there exists a vertex j such that $j \to k$ and $x_j(0) = x_k(0)$, and $x_j(0) \neq a_j$, because $x_k(0) = a_k$, $x_j(0) = x_k(0) = a_k$ and since $k \leftrightarrow j$ in G, we must have $a_j \neq a_k$. The hypothesis $F_{\vec{x}(0)} \supseteq U_{\vec{x}(0)}$ also ensures that j is satisfied, because if it was unsatisfied it should have its final colour because $F_{\vec{x}(0)} \supseteq U_{\vec{x}(0)}$ and this would contradict the property just proved that $x_j(0) = a_j$. Since j is satisfied and it has same colour than k, we have $k \not\to j$. □

Definition 13 (1-rotation). A *1-rotation* is an operator P acting on vector $\vec{s} = (s_1, s_2, \dots, s_m)$, $m > 1$, such that $P(\vec{s})_i = s_{i+1}$, $i = \{1, 2, \dots, m-1\}$ and $P(\vec{s})_m = s_1$. Repeating a 1-rotation m times yields the identity operation, *i.e.* $P^m(\vec{s}) = \vec{s}$.

Lemma 4.5. *Consider any satisfiable CP and an information set $\{\mathcal{C}_1, \dots, \mathcal{C}_N\}$ and induced graph $G' = (V, \mathcal{C})$ and colour $x_i(t) \in \mathcal{D}$ associated with each vertex $i \in V$ at time t. Suppose there exists a cycle $p_1 \to p_2 \to \cdots \to p_m \to p_{m+1} \to p_1 \subseteq G'$, $m > 1$, with $x_{p_{m+1}}(0) = x_{p_1}(0)$ at time $t = 0$. Let $\vec{s}(0) = (x_{p_1}(0), x_{p_2}(0), \dots, x_{p_m}(0))$. With probability greater than γ^{Nm}, after m time steps Algorithm 2 will realize a 1-rotation of the vector $\vec{s}(0)$, i.e. $\vec{s}(m) = P(\vec{s}(0)) = (x_{p_2}(0), x_{p_3}(0), \dots, x_{p_m}(0), x_{p_1}(0))$, while leaving the colours of all other vertices unchanged.*

Proof. Observe that at time $t = 0$ vertex p_1 is unsatisfied since $x_{p_{m+1}}(0) = x_{p_1}(0)$ and $p_{m+1} \to p_1$. Consider the event that at time $t = 1$

$$\begin{aligned} x_{p_1}(1) &= x_{p_2}(0) \\ x_{p_2}(1) &= x_{p_2}(0) \\ &\vdots \\ x_{p_m}(1) &= x_{p_m}(0). \end{aligned}$$

So vertex p_1 takes the colour of p2 and the colours of all other vertices remain unchanged. This event is feasible since Algorithm 2 ensures that all satisfied vertices will remain unchanged and each unsatisfied vertex may choose any colour from set D with probability at least γ. From the latter, the event described occurs with probability greater than γ^N. Observing that vertex p_2 is now unsatisfied since $x_{p_1}(1) = x_{p_2}(0) = x_{p_2}(1)$ and $p_1 \to p_2$, suppose that at time $t = 2$

$$\begin{aligned} x_{p_1}(2) &= x_{p_1}(1) = x_{p_2}(0) \\ x_{p_2}(2) &= x_{p_3}(1) = x_{p_3}(0) \\ x_{p_3}(2) &= x_{p_3}(1) \\ &\vdots \\ x_{p_m}(2) &= x_{p_m}(1). \end{aligned}$$

Again this event is feasible and occurs with probability greater than γ^N. After m such steps we have $\vec{s}(m) = P(\vec{s}(0))$ as claimed, and this sequence of events will occur with probability greater than γ^{Nm}. $\square$

Lemma 4.6. *Consider any satisfiable CP and an information set $\{\mathcal{C}_1, \dots, \mathcal{C}_N\}$ with induced graph $G' = (V, \mathcal{C})$ and colour $x_i(t) \in \mathcal{D}$ associated with each vertex $i \in V$ at time t. Let $A \subset \mathcal{D}^{|V|}$ denote the set of satisfying assignment. Suppose $\vec{x}(0) \notin A$ (the initial choice of colours is not a satisfying assignment) and graph G' is strongly connected. Let $d \in \mathcal{D}$ be an arbitrary colour. Let $k \in V$ be an unsatisfied vertex and let j be a vertex such that $j \to k$, $x_j(0) = x_k(0)$ (at least one such vertex exists since k is unsatisfied). With probability greater than γ^{N^3}, in $\tilde{t} < N^2$ steps Algorithm 2 will choose $\vec{x}(\tilde{t})$, such that $x_i(\tilde{t}) = x_i(0)\ \forall i \in \{i : i \in V, i \neq j\}$ and $x_j(\tilde{t}) = d$.*

Proof. Since G' is strongly connected, there exists a cycle $k \to \cdots \to j \to k \subseteq G'$. Let us relabel the $m + 1 > 1$ vertices in the cycle using

the ordering induced by the cycle, *i.e.* $p_1 = k, p_{m+1} = j$ and so $p_1 \to p_2 \to \cdots \to p_{m+1} \to p_1$.
Define vector $\vec{s}(t) = (x_{p_1}(t), \ldots, x_{p_{m-1}}(t), x_{p_m}(t))$. We need to consider two cases. $m = 1$. In this case the cycle is $p_1 \to p_2 \to p_1$. By assumption, $x_{p_2}(0) = x_{p_1}(0)$ and so vertex p_2 is unsatisfied since $p_1 \to p_2$. It follows that, with probability at least γ^N, after 1 time step Algorithm 2 will realize the event that vertex p_2 selects colour d and the colour of all other vertices remains unchanged. $m > 1$. Using Lemma 4.5, with probability greater than γ^{Nm} in m steps Algorithm 2 will realize a 1-rotation of the vector $\vec{s}(0)$ *i.e.* $\vec{s}(m) = (x_{p_2}(0), \ldots, x_{p_m}(0), x_{p_1}(0))$ leaving the colours of all other vertices unchanged. Observe that vertex $j = p_{m+1}$ must now be unsatisfied because $x_{p_m}(m) = x_{p_1}(0)$, $x_{p_{m+1}}(m) = x_{p_{m+1}}(0) = x_{p_1}(0)$ and $p_m \to p_{m+1}$. Now consider the event at time $m+1$ where vertex p_{m+1} takes the colour of vertex p_1 (and the colour of all other vertices remains unchanged). This event occurs with probability greater than γ^N. After $m+1$ steps we have $\vec{s}(m+1) = (x_{p_2}(0), \ldots, x_{p_m}(0), x_{p_1}(0))$ and $x_{p_{m+1}}(m+1) = x_{p_2}(0)$, and this event occurs with probability greater than $\gamma^{N(m+1)}$. Applying again Lemma 4.5, after a 1-rotation and changing the colour of unsatisfied vertex p_{m+1} we have $\vec{s}(2m+2) = (x_{p_3}(0), \ldots, x_{p_1}(0), x_{p_2}(0))$ and $x_{p_{m+1}}(2m+2) = x_{p_3}(0)$. This state is reached after $2(m+1)$ steps with probability greater than $\gamma^{2N(m+1)}$. Repeating, after $m(m+1)$ steps $\vec{s}(m^2+m) = (x_{p_1}(0), \ldots, x_{p_{m-1}}(0), x_{p_m}(0))$ and $x_{p_{m+1}}(m^2+m) = d$ (where at the very last step we select the colour of unsatisfied vertex p_{m+1} to equal d rather than the colour of p_1). This state is reached after $m(m+1)$ steps with probability greater than $\gamma^{Nm(m+1)}$. Since $m \leqslant N$, $m(m-1) < N^2$ steps and $\gamma^{Nm(m-1)} > \gamma^{N^3}$. $\square$

Lemma 4.7. *Consider any satisfiable CP and an information set* $\{\mathcal{C}_1, \ldots, \mathcal{C}_N\}$ *with induced graph* $G' = (V, \mathcal{C})$ *and colour* $x_i(t) \in \mathcal{D}$ *associated with each vertex* $i \in V$ *at time* t. *Let* $A \subset \mathcal{D}^{|V|}$ *denote the set of satisfying assignments. Suppose* $|V| > 1$, $\vec{x}(0) \notin A$ *(the initial choice of colours is not a satisfying assignment) and graph* G' *is strongly connected. Let* $a \in A$ *be an arbitrary satisfying assignment. If* $F_{\vec{x}(0)} \supseteq U_{\vec{x}(0)}$ *with probability greater than* γ^{N^3}, *in* $\tilde{t} \leqslant N^2$ *steps Algorithm 2 will reach an assignment* $\vec{x}(\tilde{t})$ *such that* $F_{\vec{x}(\tilde{t})} \supsetneq F_{\vec{x}(0)}$ *and* $|F_{\vec{x}(\tilde{t})}| = |F_{\vec{x}(0)}| + 1$;

Proof. Lemma 4.4 ensures a pair i, j exists such that $x_k(0) = a_k, x_k(0) \in U_{\vec{x}(0)}$ and $x_j(0) = x_k(0)$. Lemma 4.6 ensures that, in less than N^2

steps, with probability greater than γ^{N^3}, Algorithm 2 will reach an assignment in which vertex j assumes colour a_j and the colours of all other vertices are unchanged. □

Proof of Theorem 4.1. Consider Algorithm 2 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. We consider the following sequence, divided in two phases:

```
1:  $t \leftarrow 0$
2:  repeat
3:    if $F_{\vec{x}(t)} \not\supseteq U_{\vec{x}(t)}$ then
4:      Phase 1 Applying Lemma 4.3, after $\tilde{t} \leqslant N$ steps $F_{\vec{x}(t+\tilde{t})} \supseteq U_{\vec{x}(t+\tilde{t})}$ and $F_{\vec{x}(t+\tilde{t})} \supsetneq F_{\vec{x}(t)}$ (so $|F_{\vec{x}(t+\tilde{t})}| \geqslant |F_{\vec{x}(t)}| + 1$). This event happens with probability greater than $\gamma^{N^2}$.
5:      $t \leftarrow t + \tilde{t}$
6:    end if
7:    if $U_{\vec{x}(t)} \neq \emptyset$ then
8:      Phase 2 We have $F_{\vec{x}(t)} \supseteq U_{\vec{x}(t)}$. Applying Lemma 4.7, after $\tilde{t} < N^2$ steps $|F_{\vec{x}(t+\tilde{t})}| = |F_{\vec{x}(t)}| + 1$. This event happens with probability greater than $\gamma^{N^3}$.
9:      $t \leftarrow t + \tilde{t}$
10:   end if
11: until $U_{\vec{x}(t)} = \emptyset$ .
```

This sequence is terminating, because the set $F_{\vec{x}(t)}$ is strictly increasing, and when $|F_{\vec{x}(t)}| = N$ we necessarily have $U_{\vec{x}(t)} = \emptyset$. Each vertex will be added to $F_{\vec{x}(t)}$ only once, either by Phase 1 or Phase 2.

When a vertex is added by Phase 1, it will require at most N steps and occur with probability at least $\gamma^{N(N+1)/2}$. When added by Phase 2, it will require at most N^2 steps and occur with probability at least γ^{N^3}. Since $N \leqslant N^2$ and $\gamma^{N(N+1)/2} \geqslant \gamma^{N^3}$ for $N > 1$, we can therefore upper bound the total number of steps by $N \cdot N^2 = N^3$ and lower bound the probability of the sequence by $(\gamma^{N^3})^N = \gamma^{N^4}$.

Due to the Markovian nature of Algorithm 2 and the independence of the probability of the above sequence on its initial conditions, if this sequence does not occur in N^3 iterations, it has the same probability of occurring in the next N^3 iterations. The probability of convergence in $k \cdot N^3$ steps is greater than $1-(1-\gamma^{N^4})^k$. For $1-(1-\gamma^{N^4})^k \geqslant 1-\varepsilon$ we require $k \leqslant \frac{\log \varepsilon}{\log(1-\gamma^{N^4})} \leqslant -\frac{\log \varepsilon}{\gamma^{N^4}} = e^{N^4 \log(\gamma^{-1})} \log(\varepsilon^{-1})$ □

Proof of Corollary 4.2. After running Phase 1 in the proof of Theorem 4.1 for the first time, we have $F_{\vec{x}(t+\tilde{t})} \supsetneq F_{\vec{x}(t)}$. If $U_{\vec{x}(t)} = \emptyset$ we have finished without running Phase 2. Otherwise we must run Phase 2. But in this case we have from Lemma 4.4 that $\mathcal{C} \neq \mathcal{M}$ (because there exists a pair of vertices i, j such that $j \to k$ and $k \not\to j$), leading to a contradiction. So after Phase 1 $U_{\vec{x}(t)} = \emptyset$ and Phase 2 is never executed. The running time of Phase 1 is no greater than N and occurs with probability at least $\gamma^{\sum_{k=1}^{N} k} = \gamma^{(N+1)N/2}$.

□

5 SIMPLIFIED CFL AND FAST CONVERGENCE

In this chapter we introduce a simplified CFL algorithm and prove 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 $\Delta = o(N)$) when the number of available colours is greater than Δ, the maximum degree of the graph.

The simplified algorithm, in addition to being easier to analyse, also gives some insight into which parts of such algorithms are essential to provide fast convergence. It is easier to implement than CFL and we show how the proposed algorithm can be efficiently implemented to realise collision-free scheduling in two applications, a warehouse served by RFID robots and a smart electronic bookshelf, without the need to modify the RFID protocol or the readers, and keeping backward compatibility with standard RFID tags.

CONTENTS

5.1 SIMPLIFIED CFL ALGORITHM

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

Algorithm 3 Simplified Communication-Free Learning

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

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*[1].

5.1.1 *Role of Parameter S*

When $S = 0$, SCFL algorithm becomes that used in [Barcelo et al., 2011], while it is equivalent to that in [Motskin et al., 2009] when $S \to \infty$.

1 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.

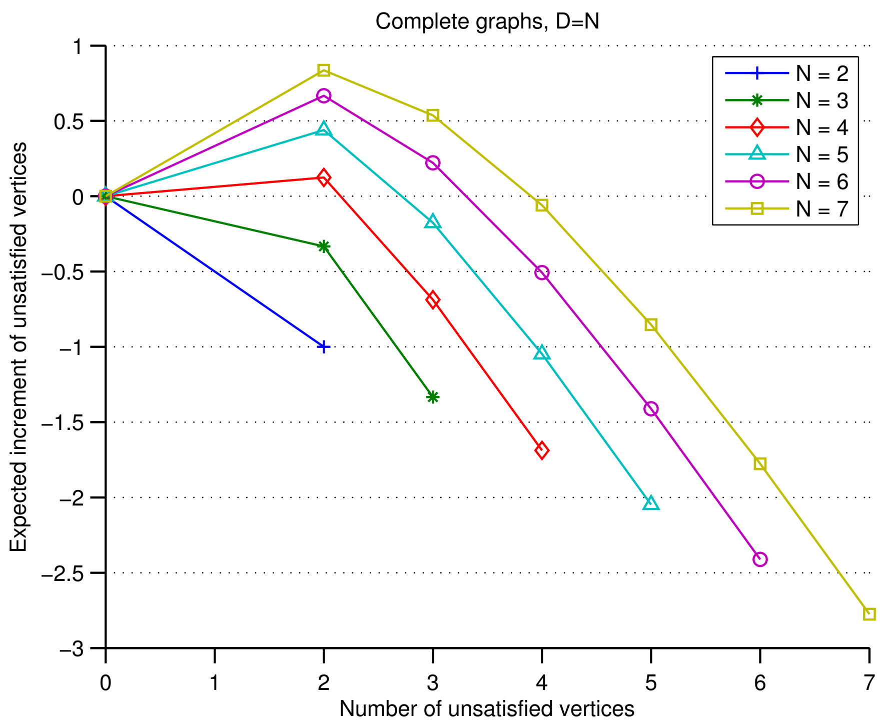


Figure 27: Expected drift for unsatisfied vertices for SCFL algorithm when $S = 0$ on a complete graph.

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 to 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 27, 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 [Motskin et al., 2009] would mean that the algorithm cannot react at all to topology changes.

In Section 5.2.1, we prove that it is enough to choose $S = \Delta + 1$ (where Δ is the maximum degree of the graph) to guarantee fast convergence.

5.1.2 *Loose Bound for any Number of Colours*

We can obtain similar bounds on the convergence time obtained in [Duffy et al., 2013] by the CFL algorithm when the number of colours used is greater than or equal to the chromatic number χ. Namely we have:

Theorem 5.1. *Consider a feasible CP on a graph $G = \{\mathcal{N}, \mathcal{E}\}$, with palette $\mathcal{D}$. Given any unsatisfied assignment $\vec{x}(0) \in \mathcal{D}^N$, then with probability greater than $1 - \varepsilon \in (0, 1)$, the number of iterations for SCFL algorithm to find a satisfying assignment is less than*

$$((S+1)N)\exp(\tfrac{(S+1)N(N+1)}{2}\log(D))\log(\varepsilon^{-1}).$$

Proof. See Appendix □

5.2 FAST COLOURING – PERFORMANCE ANALYSIS

5.2.1 *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 5.1.1).

Theorem 5.2. *Consider a CP on a graph $G = \{\mathcal{N}, \mathcal{E}\}$ with maximum degree Δ and $D > \Delta + 1$ available colours. Let $|\mathcal{N}| = N \geqslant 2$ and $\mathcal{Z}_t$ be the set of vertices in the non-permanent state at time $t \in \{0, 1, 2, \dots\}$, with*

$|\mathcal{Z}_t| = Z_t \in \{0, 1, 2, \dots, N\}$. *Let* R *be the first time in which all vertices reach the permanent state. For* SCFL *algorithm with* $S = \Delta + 1$ *we have*

$$\mathbb{P}\left(R \geqslant \frac{\log N + \log(\varepsilon^{-1}) + K}{\log \frac{\Delta+1}{\Delta} + \frac{K}{\Delta+1}} \right) \leqslant \varepsilon,$$

for $-1 < K \leqslant \log \frac{1}{1+\log 4}$. *Given* ε*, we denote the bound* $B(N, \Delta, 1-\varepsilon) = \frac{\log N + \log(\varepsilon^{-1}) + K}{\log \frac{\Delta+1}{\Delta} + \frac{K}{\Delta+1}}$ *when* $K = \log \frac{1}{1+\log 4}$.

Corollary 5.1. *If* $\Delta = o(N)$*, then for* $N \to \infty$*, the order of steps before convergence is* $R = \mathcal{O}(\log N)$*, or more precisely*

$$\mathbb{P}\big(R \geqslant \log N + o(1)\big) \leqslant \varepsilon, \quad \textit{for } N \to \infty.$$

Corollary 5.2 (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*

$$\mathbb{P}\big(R \geqslant N \log N + o(1)\big) \leqslant \varepsilon, \quad \textit{for } N \to \infty.$$

Proof. See Appendix. □

5.2.2 *Discussion*

Theorem 5.2 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 5.1, 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 $D \geqslant \Delta + 1$ cannot be easily extended to the general case of an arbitrarily small number of colours. Szegedy and Vishwanathan [1993] 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 SCFL algorithm is order optimal in the case of complete graphs (when $\Delta = N$).

5.2.3 *Differences with the State-of-the-art*

As we will see by means of simulations in Section 5.2.4, SCFL algorithm has very similar performance to CFL algorithm.

The main advantage of 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 more appealing for real world applications, see for example Section 5.3.2.

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 colour 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 5.2 considerably more involved, hence it is left for future work.

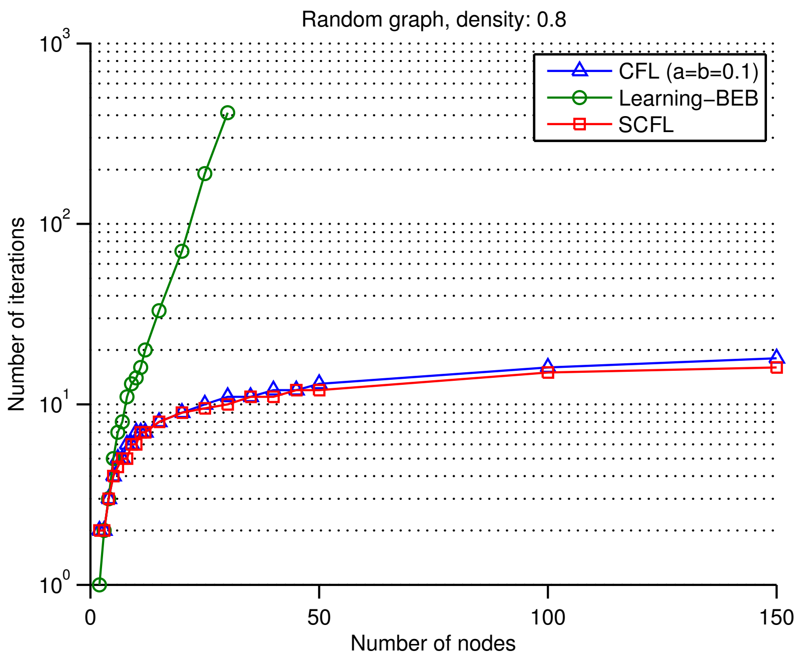
5.2.4 *Simulations*

In Figure 28, a comparison of the convergence time over 10 000 random graphs of the CFL algorithm, the Learning-BEB [Barcelo et al., 2011] algorithm and SCFL is shown, when $\Delta + 1$ colours are available. Two classes of random graphs are created by selecting each 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 5.1.1 and Figure 27 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 make them uninteresting (and hard to compute) in the present context.

In Figure 29, 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 5.2 with confidence $1 - \varepsilon = 0.9$ is 2167 iterations.

In Figure 30, the empirical CDF of convergence time for the CFL and SCFL algorithms is shown. Also the upper bound obtained using



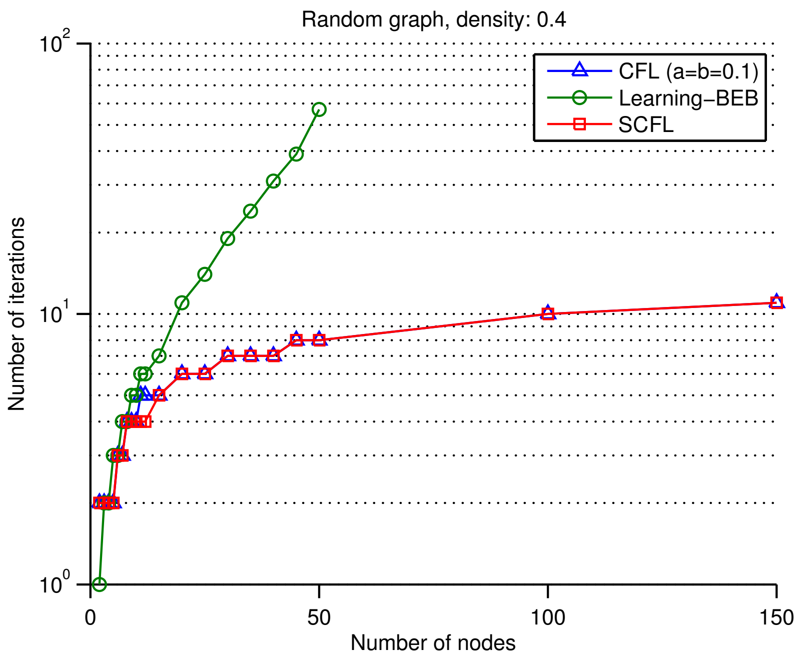


Figure 28: Comparison of the convergence time on random graphs for CFL algorithm, Learning-BEB algorithm and Algorithm 1 for densities of 0.8 and 0.4 respectively, using $\Delta + 1$ colours.

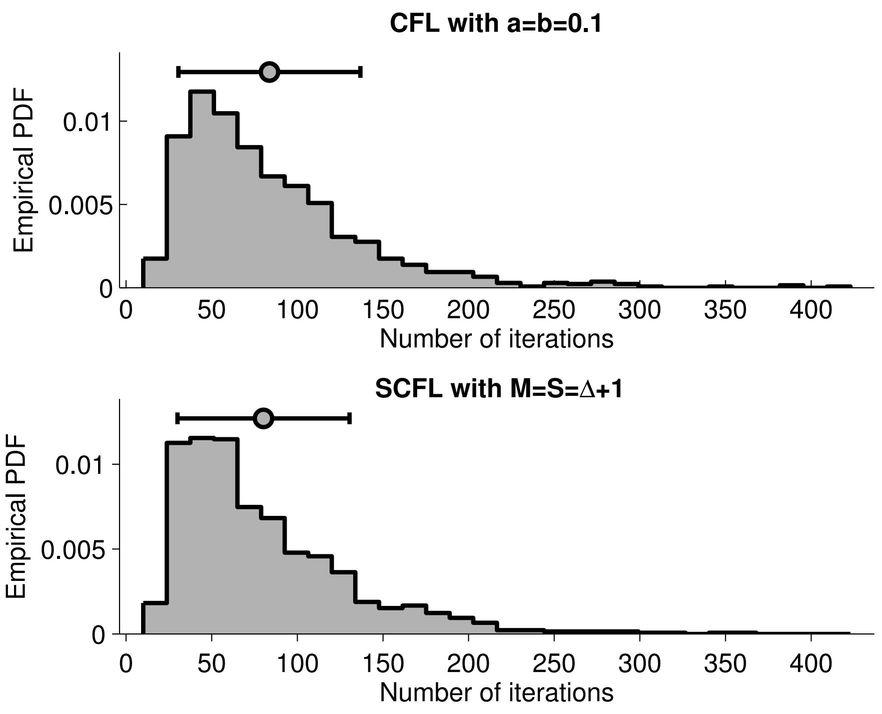


Figure 29: Empirical PDF of the number of iterations to colour a complete graph with 48 vertices over 10 000 samples, for the CFL algorithm and the SCFL algorithm.

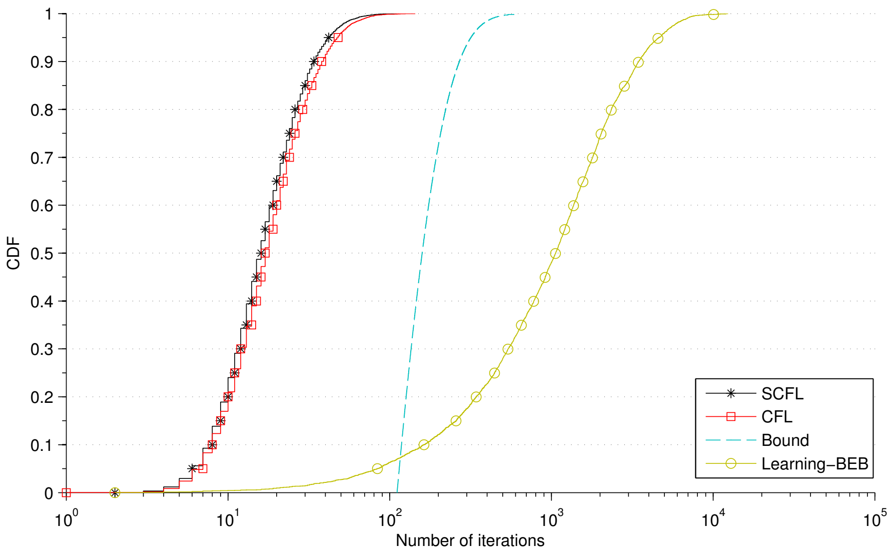


Figure 30: Empirical CDF of convergence time for CFL and SCFL. Also the upper bound obtained using Theorem 5.2 is shown.

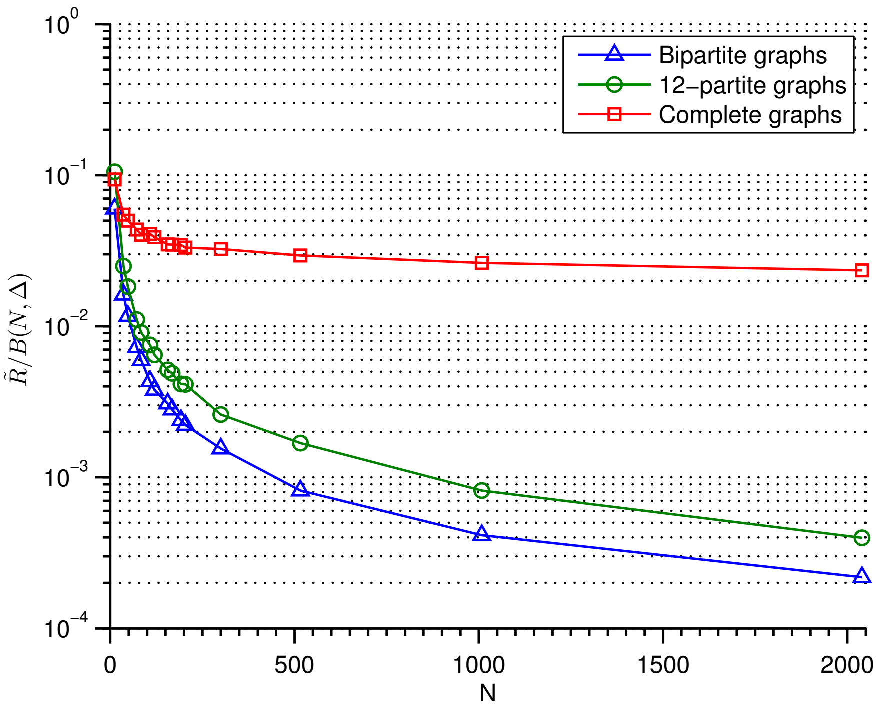


Figure 31: Ratio between the empirical estimation of the number of iterations and the bound $B(N, \Delta, 1/2)$.

Theorem 5.2 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 [1993] 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 31 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 very similar qualitative results for bipartite and random graphs, thus they have not been included.

5.3 USE CASE – RFID ROBOT/SMART BOOKSHELF

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 [Yan et al., 2008, Chow et al., 2006, Carreras et al., 2013]. The robot will move in 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 in the items using one antenna per time, facilitating administrative tasks like renting or collection [Melià-Seguí et al., 2013, Lang and Han, 2014, Lau et al., 2010].

We will refer to both the robot and the antenna array as *reader*. We are interested only in the RFID identification phase of the problem. In Figure 32 a schematic of both examples is shown, in which the reader (either a RFID robot or one antenna in the array) is in the range of detection of 12 tags per time slot(tags with 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 [Finkelzeller, 2003, Want, 2006, Shih et al., 2006]. 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 other 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.

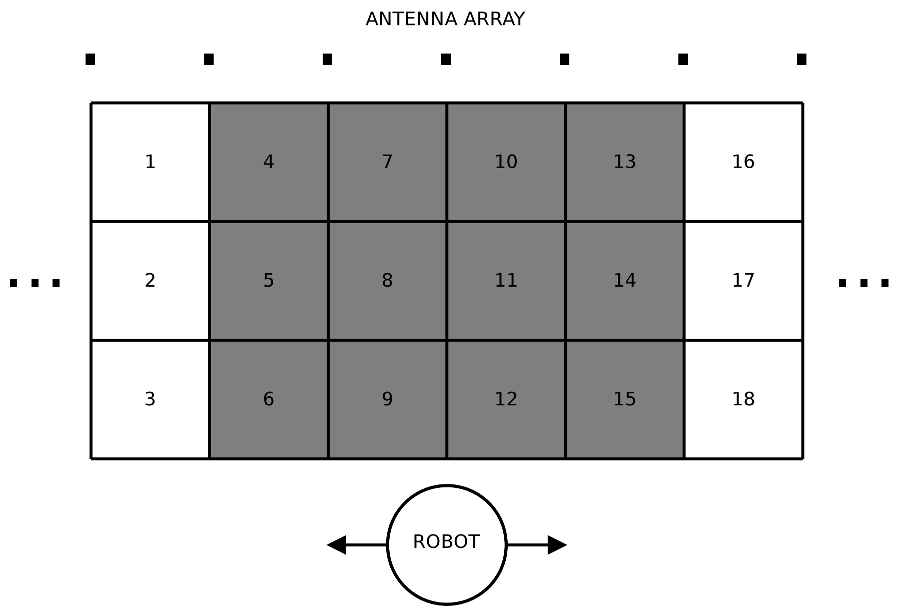


Figure 32: A schematic example of a grid of tags, in which the reader (either a RFID robot or an antenna in an array) is in the range of detection of 12 tags per time slot (tags with gray background when the reader is in central position).

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.

5.3.1 *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. [2012] 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 $G_{s_1,\dots,s_k}$, *i.e.* the graph composed of k independent sets of (possibly different) size $s_i, i = 1, \dots, k$, 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 32 can be modeled as a CP on a 12-partite graph.

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 5.3.3.

5.3.2 *Implementation*

We show how to implement SCFL algorithm in a 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 *QueryAdjust* command. The 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 *QueryAdjust* command every S period, *i. e.* every S · D queries.

Modified tags will thus have the following additional capabilities

A. 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.

B. If the tag receives a *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.

5.3.3 *Comparison with Slotted Aloha*

The average time needed by SCFL algorithm to identify correctly all tags in a complete graph has been computed and compared with the one of the following algorithms [Lee et al., 2005], 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 [Lee et al., 2005] for detail on 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, in the sense that for each superframe their behaviour is a realisation of the same stochastic process. On the other hand, SCFL algorithm has a transient period in which a collision-free schedule is being searched, while after convergence the tags will deterministically select same slot at every subsequent superframe. As shown in Table 2, 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 as well better than the state-of-the-art dynamically adjusted slotted Aloha (66 % reduction). Using ISO15693 high tag data rate [Want, 2006], 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

	200 tags		1000 tags	
Algorithm	First inventory	At steady state	First inventory	At steady state
BFSA	1280	1280	5850	5850
DFSA	662	662	5425	5425
EDFSA	628	628	2916	2916
SCFL	816	200	5040	1000

Table 2: 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.

case of a tag grid in which 12 tags can interfere for each antenna, we simulated time of convergence, time of first inventory and time of identification at steady state of SCFL algorithm, compared to standard slotted Aloha with flagging enabled and superframe size equal to $\Delta + 1$ (as SCFL) varying the number of tags, for complete graphs and 12-partite complete graphs. In Figure 33, we can see that time of first inventory is comparable for the two algorithms, and after conver-

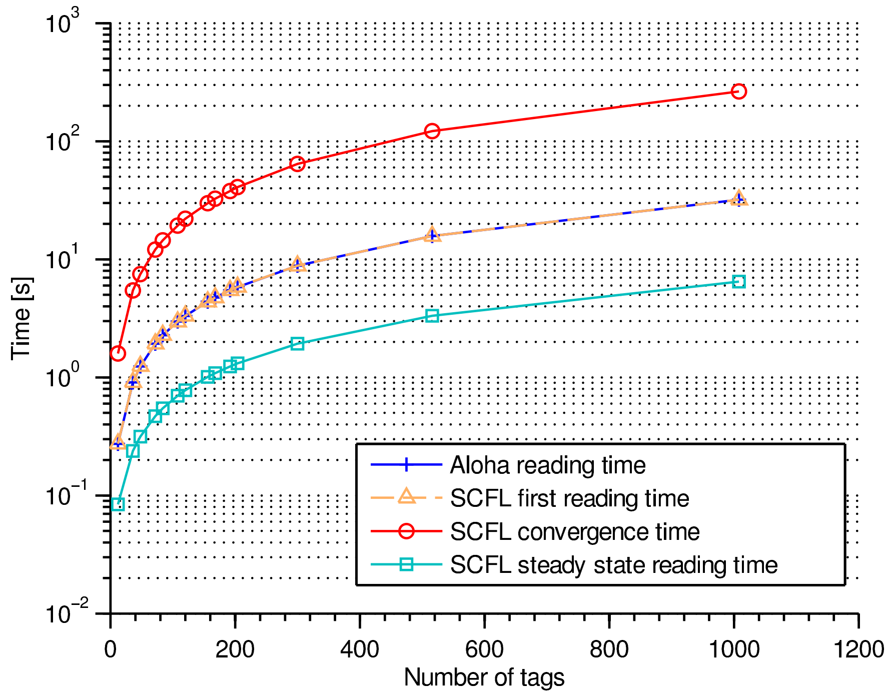


Figure 33: Reading time of SCFL algorithm and Aloha for a 12-partite complete graph. For SCFL algorithm time of convergence and time of reading at steady state are also shown.

gence (less than 5 minutes for a shelf of 1000 items) 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 know the status of the items (taken or not), while if new books are introduced or the topology of the grid change, SCFL algorithm will have same performance of Aloha for at most 5 minutes, converging again to a collision-free schedule.

5.3.4 *Memory and Computation Footprint vs. CFL*

The implementation of 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 34, where the empirical PDF of the number of iterations 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 5.2 with confidence $1 - \varepsilon = 0.9$ is 784 iterations.

5.3.5 *Summary*

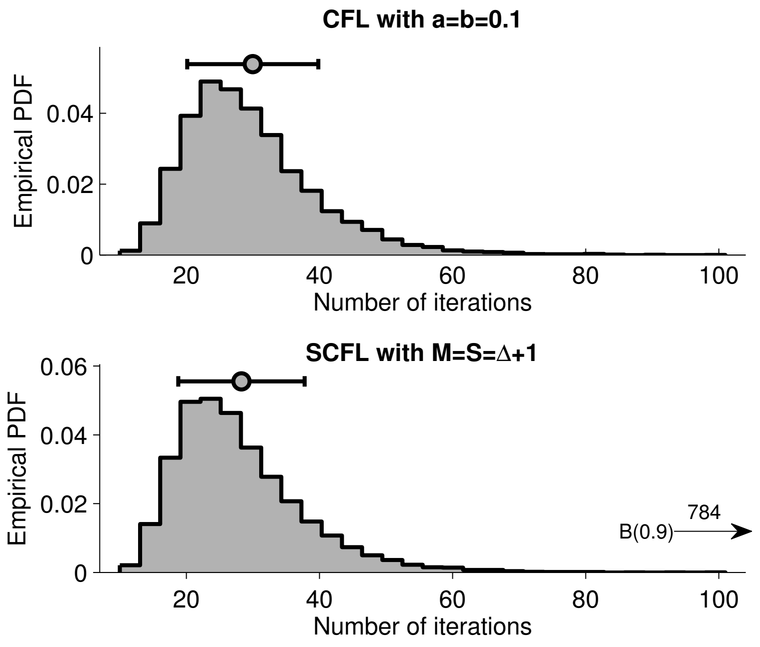


Figure 34: Empirical PDF of the number of iterations to colour a complete 12-partite graph with 10 vertices for each independent set over 10 000 samples, for the CFL algorithm and the SCFL algorithm.

The SCFL algorithm is responsive to topology changes, and 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 $\Delta = o(N)$) when the number of available colours is greater than Δ.

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.

5.A 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 χ) 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 5.1. *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 γ.*

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

Proof of Theorem 5.1. 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

$$x_i((S+1)) = \begin{cases} a_i & \text{if } i \in U_{\vec{x}(0)}, \\ x_i(0) & \text{otherwise.} \end{cases} \tag{25}$$

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 (25), *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 $\gamma^{(S+1)|U_{\vec{x}(\tau(S+1)-1)}|}$. 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 $\gamma^{(S+1)\cdot 1} \cdot \gamma^{(S+1)\cdot 2} \dots \gamma^{(S+1)\cdot N} = \gamma^{(S+1)N(N+1)/2}$.

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 $1 - \left(1 - \gamma^{\frac{(S+1)N(N+1)}{2}}\right)^k$.

For $1 - \left(1 - \gamma^{\frac{(S+1)N(N+1)}{2}}\right)^k \geqslant 1 - \varepsilon$, we require

$$\begin{aligned} k \leqslant \frac{\log \varepsilon}{\log\left(1 - \gamma^{\frac{(S+1)N(N+1)}{2}}\right)} &\leqslant - \frac{\log \varepsilon}{\gamma^{\frac{(S+1)N(N+1)}{2}}} \\ &= e^{\frac{(S+1)N(N+1)}{2}\log(\gamma^{-1})} \log(\varepsilon^{-1}). \end{aligned}$$

□

Lemma 5.2. *If vertex i is in non-permanent state at the end of iteration t, then*

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

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 $\frac{\Delta}{M} \leqslant \frac{\Delta}{\Delta+1}$, because $M \geqslant \Delta+1$; so we have $\frac{M-\Delta}{M} = 1 - \frac{\Delta}{M} \geqslant 1 - \frac{\Delta}{\Delta+1} = \frac{1}{\Delta+1}$. □

Lemma 5.3. *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 5.3. *A vertex in permanent state can be unsatisfied only by non-permanent neighbours.*

Lemma 5.4. *If a vertex i is in permanent state and the counter k is equal to zero, then*

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

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 5.3, 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 $M \geqslant \Delta + 1$, we have $\frac{1}{M} \leqslant \frac{1}{\Delta+1}$ and so $1 - \frac{1}{M} \geqslant 1 - \frac{1}{\Delta+1} = \frac{\Delta}{\Delta+1}$. □

Lemma 5.5. *Let* Z, N, Δ *positive integer numbers, with* $N - Z \geqslant 1$*, and* $\Delta \geqslant 2$*. 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:

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

□

Lemma 5.6. *For any choice of the integers* $N > 2$, $1 \leqslant \Delta \leqslant N - 1$, $\tau \geqslant 1$, *and the real* $1 + \log 4 < k < e$ *we have*

$$(\tfrac{\Delta}{\Delta+1})^{\tau(\Delta+1)+1} k^{\tau-1} N + \left(1 - \left(\frac{\Delta}{\Delta+1}\right)^{\frac{\Delta\left(\frac{\Delta}{\Delta+1}\right)^{\tau(\Delta+1)} k^{\tau-1} N}{N - \left(\frac{\Delta}{\Delta+1}\right)^{\tau(\Delta+1)} k^{\tau-1} N}}\right)$$
$$(N - (\tfrac{\Delta}{\Delta+1})^{(\Delta+1)} k^{\tau-1} N) \leqslant k^{\tau} N (\tfrac{\Delta}{\Delta+1})^{\tau(\Delta+1)+1}. \tag{26}$$

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 Δ and decreasing with τ, 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 (26) is equivalent to

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

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 Δ, so we just obtain the bound choosing $\Delta = 1$, that brings to $F(\Delta, Y) \leqslant F^*(1) = 1 + \log 4$. This guarantees that (26) is satisfied when $1 + \log 4 < k < e$. □

Lemma 5.7. *Let $\mathcal{N} \supseteq \mathcal{Z}$ two integer sets of cardinality N and Z respectively, and $N > 1$ and $Z \leqslant N$. Let $\Delta > 1$ be an integer and n a integer vector of length N. If*

$$n(i) = 0, \quad \textit{when } \mathcal{Z} = \mathcal{N},$$

and

$$\sum_{i\in\mathcal{N}\setminus\mathcal{Z}} n(i) \leqslant \Delta Z, \tag{27}$$

then the following holds

$$f(Z, n) := \sum_{i\in\mathcal{N}\setminus\mathcal{Z}} \left(1 - \left(\frac{\Delta}{\Delta+1}\right)^{n(i)}\right) \leqslant \left(1 - \left(\frac{\Delta}{\Delta+1}\right)^{\frac{\Delta Z}{N-Z}}\right)(N - Z). \tag{28}$$

Proof. We maximise over n the concave function $f(Z, n)$ subject to constraint (27). 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 $Z \geqslant 1$ 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 $\nabla_{n(i)} f(Z, n) = 0 \Leftrightarrow \mu = -\log\frac{\Delta}{\Delta+1}\left(\frac{\Delta}{\Delta+1}\right)^{n(i)}$, where μ is the (unique) multiplier, because $1 - 1/\Delta \neq 0$), and from complementary slackness we obtain that constraint (27) is tight (because we get that $\mu = 0$ if constraint (27) 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

$$n(i) = \frac{\Delta Z}{N - Z}, \quad i \in \mathcal{N} \setminus \mathcal{Z}, \quad Z < N, \tag{29}$$

and $n(i) = 0$ when $Z = N$ (by definition of $n(i)$). □

Proof of Theorem 5.2. Let Z_t the number of permanent vertices at time t. Observe that, by Lemma 5.3, $Z_t = 0$ is an absorbing state *i. e.* $Z_\tau = 0$ $\forall\, \tau \geqslant t$, and since Z_t is non-negative we have that $\mathbb{E}[Z_t] = 0$ implies $Z_t = 0$. It follows that $\mathbb{P}(R \geqslant \tau \cdot S) = \mathbb{P}(Z_{\tau S} \geqslant 1)$ and by Markov's inequality,

$$\mathbb{P}(Z_{\tau S} \geqslant 1) \leqslant \mathbb{E}[Z_{\tau S}].$$

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

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

So now we have

$$\mathbb{E}\big[Z_{t+1} | Z_t\big] = Z_t - \mathbb{E}\big[M(t+1)\big] = Z_t - \sum_{i \in \mathcal{Z}_t} \mathbb{E}\big[X_i(t+1)\big] =$$
$$Z_t - \sum_{i \in \mathcal{Z}_t} 1 \cdot \mathbb{P}\big(i \text{ is permanent at time } t+1\big), \tag{30}$$

and we can use Lemma 5.2 to obtain that in expectation, at least a fraction $\frac{\Delta}{\Delta+1}$ of non-permanent vertices will become permanent at each step

$$\mathbb{E}\left[Z_{t+1}|Z_t\right] \leqslant \left(1 - \frac{1}{\Delta+1}\right) Z_t, \; t \neq S \mod S. \tag{31}$$

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

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

Second Part: $\mathbf{t > S}$

At time $\tau \cdot S + 1$, $\tau \geqslant 1$, we have to consider the number of vertices that exit from permanent state. By Lemma 5.4, and using the same construction used in (30) and (31), the probability a vertex i remains permanent after a reset is:

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

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:

$$e(Z_t) \leqslant \sum_{i \in \mathcal{N} \setminus \mathcal{Z}_t} \left(1 - \left(\frac{\Delta}{\Delta+1}\right)^{n(i,t)}\right) =: f(Z_t, n(\cdot, t)). \tag{32}$$

Each non-permanent vertex can affect at most Δ permanent vertices, and because of Corollary 5.3 a permanent vertex cannot affect any other permanent vertex, so the set $\mathcal{N} \setminus \mathcal{Z}_t$ can be affected by at most a number of edges equal to

$$\sum_{i \in \mathcal{N} \setminus \mathcal{Z}_t} n(i,t) \leqslant \Delta Z_t. \tag{33}$$

To bound $e(Z_t)$, we maximise over $n(\cdot,t)$ the concave function $f(Z_t, n(\cdot,t))$ subject to constraint (33). Using Lemma 5.7 we get for any $\tau > 1$, combining (31), (28) and (29)

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

Applying iterated expectation rule,

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

We can now use Lemma 5.5 to apply Jensen's inequality:

$$\mathbb{E}\left[\varphi(Z_t)\right] \leqslant \varphi(\mathbb{E}[Z_t]),$$

with $\varphi(Z_t) = (N - Z_t)\left(1 - \left(\frac{\Delta}{\Delta+1}\right)^{\frac{\Delta Z_t}{N - Z_t}}\right)$. So we obtain

$$\mathbb{E}\left[Z_{\tau S+1}\right] = \mathbb{E}\left[\mathbb{E}\left[Z_{\tau S+1} | Z_{\tau S}\right]\right] \leqslant (\tfrac{\Delta}{\Delta+1})\, \mathbb{E}[Z_{\tau S}]$$
$$+ \left(1 - \left(\tfrac{\Delta}{\Delta+1}\right)^{\frac{\Delta \mathbb{E}\left[Z_{\tau S}\right]}{N - \mathbb{E}\left[Z_{\tau S}\right]}}\right)(N - \mathbb{E}[Z_{\tau S}]). \tag{34}$$

We set now, $S = \Delta + 1$, and we start considering the case $\tau = 1$. From (31), we have that $\mathbb{E}\left[Z_{\Delta+1}\right] \leqslant \left(\frac{\Delta}{\Delta+1}\right)^{\Delta+1} N < \frac{N}{e}$. We want to bound the RHS of (34) substituting $\mathbb{E}\left[Z_{\Delta+1}\right]$ with $\left(\frac{\Delta}{\Delta+1}\right)^{\Delta+1} N$, but to do that we have to prove that the RHS of (34) is increasing with $\mathbb{E}\left[Z_{\Delta+1}\right]$. The first addend of the RHS of (34) is affine and increasing with $\mathbb{E}\left[Z_{\Delta+1}\right]$. Let us call second addend f. For Lemma 5.5, f is strictly concave, so the subgradient property is $f(x) \leqslant f(y) + (x - y)\partial f(y)$ and so $f(x) \leqslant f(N/e) + (x - N/e)\partial f(N/e)$, and for $x \in [0, N/e]$ and $\partial f(N/e) \geqslant 0$ we have $f(x) \leqslant f(N/e)$. Thus it is enough to show that $f' > 0$ when $\mathbb{E}\left[Z_{\Delta+1}\right] = \frac{N}{e}$. To simplify the analysis, we do the following strictly monotonic change of variable (that thus preserve the stationary points): $x = \frac{\mathbb{E}\left[Z_{\Delta+1}\right]}{N - \mathbb{E}\left[Z_{\Delta+1}\right]}$.

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

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

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

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

$$g_1(\Delta) = +e\left(\frac{\Delta}{\Delta+1}\right)^{\frac{\Delta}{e-1}} - \left(\frac{\Delta}{\Delta+1}\right)^{\frac{\Delta}{e-1}} - e + 1 > 0$$
$$g_2(\Delta) = -e\,\Delta\left(\frac{\Delta}{\Delta+1}\right)^{\frac{\Delta}{e-1}}\log\left(\frac{\Delta}{\Delta+1}\right).$$

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

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:

$$\mathbb{E}\left[Z_{\Delta+2}\right] \leqslant (\tfrac{\Delta}{\Delta+1})^{\Delta+2}N + \left(1-\left(\frac{\Delta}{\Delta+1}\right)^{\frac{\Delta\left(\frac{\Delta}{\Delta+1}\right)^{\Delta+1}N}{N-\left(\frac{\Delta}{\Delta+1}\right)^{\Delta+1}N}}\right)(N-(\tfrac{\Delta}{\Delta+1})^{\Delta+1}N). \quad (35)$$

We can now use Lemma 5.6, with $\tau = 1$ to bound the RHS of (35) with $kN(\frac{\Delta}{\Delta+1})^{(\Delta+1)+1}$ (we note that this quantity is again smaller than kN/e). Applying again (31) for the next $N-1$ steps, we get that $\mathbb{E}\left[Z_{2(\Delta+1)}\right] \leqslant kN(\frac{\Delta}{\Delta+1})^{2(\Delta+1)}$, that is thus smaller than N/e (because $k < e$). So we can use the same reasoning that brought from (34) to (35) and then apply Lemma 5.6 again with $\tau = 2$. Iterating this procedure we get, at any step $\tau(\Delta+1)$,

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

Now, setting a target small probability ε, we need to choose

$$\tau \geqslant \frac{\log N + \log(\varepsilon^{-1}) + \log(k^{-1})}{(\Delta+1)\log(\frac{\Delta+1}{\Delta}) + \log(k^{-1})},$$

to obtain the thesis. $\square$

Proof of Corollaries 5.1 and 5.2. Corollary 5.1 is obtained taking the limit for $N \to \infty$. Corollary 5.2 is obtained observing that $\frac{1}{N}$ is the first term of Taylor series at ∞ of $\log\left(\frac{N}{N-1}\right)$. $\square$

6 CONCLUSIONS

Proportional Fairness: Chapter 2

In this thesis, we provide the first rigorous analysis of proportional fairness in 802.11 WLANs. We show that a unique proportional fair rate allocation exists and, correcting previous studies, that this allocation assigns equal *total air-time* to flows. Total air-time is the time spent on both colliding and successful transmissions and differs from other air-time quantities proposed heuristically in the literature.

Scrambling Code Allocation: Section 3.2

We introduce the problem of scrambling code allocation for WCDMA small cell networks. The problem differs from code planning in macrocell networks due to the limited number of codes reserved for small cells and the need for dynamic adaptation and for scalable, distributed planning. We adapt the CFL algorithm to this task, allowing asynchronous updates, and we evaluate its performance against two variants of 3GPP recommended schemes. The results confirm significant performance improvement. The proposed scheme is fully distributed, which makes it suitable for unplanned deployment of small cells base-stations.

Local Topology Discovery: Section 3.3

We introduce the problem of user-reports-based Local Topology Discovery, providing a crisp mathematical formulation of it in the case of a simple mobility model (Model 1). We show that Model 1 can effectively be used as an upper bound for a wide range of mobility models, when the user reports frequency is lower than the mixing time of the Markov chain of the underlying mobility model.

Simulations on random scenarios show that the expected number of reports to have 0.9-knowledge of the local topology is modest. Roughly speaking, a user moving at 0.5 m/s according to a random walk model, and providing a report every hour, will guarantee the Access Point will have 0.9-knowledge with high probability in less than half day. Since the local topology is not typically expected to change every day, this is an acceptable time. The simulations on more realistic scenarios (Section 3.3.3.3) give similar results in term of time to 0.9-knowledge. These results encourage the implementation of the user reports function, corroborating the heuristic recommendations in Edwards [2008] and Checco et al.. In the case of femtocells, such implementation would be easy, because the hardware and the firmware are already capable of managing user reports.

Sensing Restrictions: Chapter 4

We then investigate the impact of sensing restrictions on decentralised learning-based Colouring Problem solvers. We constructively establish the existence of solvers that are able to find satisfying assignments even in the presence of sensing restrictions, in particular sensing asymmetry of the type encountered when hidden terminals are present. Our main analytic contribution is to establish sufficient conditions on the sensing behaviour to ensure that the solvers find satisfying assignments with probability one. These conditions take the form of connectivity requirements on the induced sensing graph. These requirements are mild, and we demonstrate that they are commonly satisfied in wireless allocation tasks. We explore the impact of sensing constraints on the speed with which a satisfying assignment is found, showing the increase in convergence time is not significant in common scenarios. Our results are of considerable practical importance in view of the prevalence of both communication and sensing restrictions in wireless resource allocation problems. The class of algorithms analysed here requires no message-passing whatsoever between wireless devices, and we show that they continue to perform well even when devices are only able to carry out constrained sensing of the surrounding radio environment.

Fast colouring with $\Delta + 1$ colours: Chapter 5

Finally, we introduce the SCFL algorithm. It is responsive to topology changes, and 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 $\Delta = o(N)$) when the number of available colours is greater than Δ. The SCFL algorithm can be efficiently implemented in realistic industrial tasks, such as a warehouse served by RFID robots, or a smart electronic bookshelf, without the need to modify the RFID protocol or the readers, and keeping 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 higher. We can conclude that 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.

6.1 FUTURE WORK

Regarding proportional fairness, the main direction of research now is to study proportional fairness in multi-hop wireless networks.

Proportional Fairness: Chapter 2

Regarding Local Topology Discovery more extensive study on more realistic scenarios is required, where the typical topological properties of a urban area are taken in account. Similarly, an analysis of more realistic mobility models is desirable: our work encompasses the simple case of Model 1, that can be used to estimate any other Markovian model (*i. e.* any mobility model that can be described with a Markov process) with unique stationary measure only when the report frequency is sufficiently low, namely slower than the mixing time of the Markov chain. An analysis of the behaviour of the mobility models during their transient behaviour is left for future work.

Local Topology Discovery: Section 3.3

With regards to sensing restriction study, future work includes the research of more refined results for specific classes of graphs and, more importantly, the extension of our analysis to more general DCS problems.

Sensing Restrictions: Chapter 4

The main limitation of SCFL algorithm is the requirement of a (although weak) form of central synchronisation, that we believe can be relaxed in a future work. This would make the algorithm suitable for more complex applications, like 802.11 networks. Also in this case, the extension to more general DCS problems would contribute to the evolution of this field.

Fast colouring with $\Delta + 1$ colours: Chapter 5

PUBLICATIONS

I. Alessandro Checco and Douglas J. Leith. Learning-Based Constraint Satisfaction With Sensing Restrictions. *Selected Topics in Signal Processing, IEEE Journal of*, 7(5):811–820, 2013.

II. Alessandro Checco and Douglas J. Leith. Proportional Fairness in 802.11 Wireless LANs. *Communications Letters, IEEE*, 15(8):807–809, 2011.

III. Alessandro Checco, Rouzbeh Razavi, Douglas J. Leith, and Holger Claussen. Self-configuration of Scrambling Codes for WCDMA Small Cell Networks. In *Personal Indoor and Mobile Radio Communications (PIMRC), 2012 IEEE 23rd International Symposium on*, pages 149–154. IEEE, 2012.

IV. Alessandro Checco and Douglas J. Leith. Fast, responsive decentralised graph colouring. *CoRR*, abs/1405.6987, 2014.

V. Alessandro Checco, Carlo Lancia, and Douglas J. Leith. Using crowd sourcing for local topology discovery in wireless networks. *CoRR*, abs/1401.1551, 2014.

BIBLIOGRAPHY FOR CHAPTER 1

M. Heusse, F. Rousseau, G. Berger-Sabbatel, and a. Duda. Performance anomaly of 802.11b. *IEEE INFOCOM 2003. Twenty-second Annual Joint Conference of the IEEE Computer and Communications Societies (IEEE Cat. No.03CH37428)*, pages 836–843, 2003. doi: 10.1109/INFCOM.2003.1208921. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1208921.

Julien Herzen, Adel Aziz, Ruben Merz, Seva Shneer, and Patrick Thiran. A measurement-based algorithm to maximize the utility of wireless networks. In *Proceedings of the 3rd ACM workshop on Wireless of the students, by the students, for the students*, pages 13–16. ACM, 2011.

Li Bin Jiang and Soung Chang Liew. Proportional fairness in wireless LANs and ad hoc networks. *IEEE Wireless Communications and Networking Conference, 2005*, pages 1551–1556, 2005. doi: 10.1109/WCNC.2005.1424745. URL `http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1424745`.

G. Tan and J. Guttag. Time-based fairness improves performance in multi-rate WLANs. In *Proceedings of the annual conference on USENIX Annual Technical Conference*, page 23. USENIX Association, 2004. URL `http://portal.acm.org/citation.cfm?id=1247438`.

Jiaping Liu, Alexander L. Stolyar, Mung Chiang, and H. Vincent Poor. Queue Back-Pressure Random Access in Multihop Wireless Networks: Optimality and Stability. *IEEE Transactions on Information Theory*, 55(9):4087–4098, September 2009. ISSN 0018-9448. doi: 10.1109/TIT.2009.2025563. URL `http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5208499`.

Koushik Kar, S Sarkar, and L Tassiulas. Achieving proportional fairness using local information in Aloha networks. *IEEE Transactions on*, 2004. URL `http://repository.upenn.edu/cgi/viewcontent.cgi?article=1069&amp;context=ese_papers`.

Xin Wang and Koushik Kar. Cross-layer rate control for end-to-end proportional fairness in wireless networks with random access. *Proceedings of the 6th ACM international symposium on Mobile ad hoc networking and computing - MobiHoc '05*, page 157, 2005. doi: 10.1145/1062689.1062710. URL `http://portal.acm.org/citation.cfm?doid=1062689.1062710`.

VA Siris and George Stamatakis. Optimal CWmin selection for achieving proportional fairness in multi-rate 802.11 e wlans: Test-bed implementation and evaluation. *network testbeds, experimental evaluation*, pages 41–48, 2006. URL `http://portal.acm.org/citation.cfm?id=1160996`.

Albert Banchs, Pablo Serrano, and Huw Oliver. Proportional fair throughput allocation in multirate IEEE 802.11e wireless LANs. *Wireless Networks*, 13(5):649–662, June 2006. ISSN 1022-0038. doi: 10.1007/s11276-006-6972-9. URL `http://www.springerlink.com/index/10.1007/s11276-006-6972-9`.

Joseph Dunn, Michael Neufeld, Anmol Sheth, Dirk Grunwald, and John Bennett. A Practical Cross-Layer Mechanism For Fairness in 802.11 Networks. *Mobile Networks and Applications*, 11 (1):37–45, December 2005. ISSN 1383-469X. doi: 10.1007/s11036-005-4459-z. URL http://www.springerlink.com/index/10.1007/s11036-005-4459-z.

Thyagarajan Nandagopal, Tae-Eun Kim, Xia Gao, and Vaduvur Bharghavan. Achieving MAC layer fairness in wireless packet networks. *Proceedings of the 6th annual international conference on Mobile computing and networking - MobiCom '00*, pages 87–98, 2000. doi: 10.1145/345910.345925. URL http://portal.acm.org/citation.cfm?doid=345910.345925.

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

W. Webb. *Wireless Communications: The Future*. New York, 2007.

3GPP TR 25.967. Home Node B (HNB) Radio Frequency (RF) requirements (FDD). Ver 10(version 10), 2011.

R. Choudhury, S. Bandyopadhyay, and K. Paul. A mobile agent based mechanism to discover geographical positions of nodes in ad hoc wireless networks. In *6th Asia-Pacific Conference on Communications (APCC2000), Seoul, Korea*, volume 30, 2000.

S. Marwaha, C.K. Tham, and D. Srinivasan. A novel routing protocol using mobile agents and reactive route discovery for ad hoc wireless networks. In *Networks, 2002. ICON 2002. 10th IEEE International Conference on*, pages 311–316. IEEE, 2002.

R. Choudhury, S. Bandyopadhyay, and K. Paul. A distributed mechanism for topology discovery in ad hoc wireless networks using mobile agents. In *Proceedings of the 1st ACM international symposium on Mobile ad hoc networking & computing*, pages 145–146. IEEE Press, 2000.

A. Raniwala and T. Chiueh. Architecture and algorithms for an IEEE 802.11-based multi-channel wireless mesh network. In *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, volume 3, pages 2223–2234. IEEE, 2005.

A. Mishra, V. Shrivastava, D. Agrawal, S. Banerjee, and S. Ganguly. Distributed channel management in uncoordinated wireless environments. In *Proceedings of the 12th annual international conference on Mobile computing and networking*, pages 170–181. ACM, 2006a.

A. Mishra, V. Brik, S. Banerjee, A. Srinivasan, and W. Arbaugh. A client-driven approach for channel management in wireless LANs. In *IEEE Infocom*, volume 6, 2006b.

K. K. Leung and B. J. Kim. Frequency assignment for IEEE 802.11 wireless networks. In *sVehicular Technology Conference, 2003. VTC 2003-Fall. 2003 IEEE 58th*, volume 3, pages 1422–1426. IEEE, 2003.

L. Narayanan. *Handbook of Wireless Network and Mobile Computing*, chapter Channel assignment and graph multicoloring. Wiley Series on Parallel and Distributed Computing, 2002.

O. Dousse. Percolation in directed random geometric graphs. In *Information Theory Proceedings (ISIT), 2012 IEEE International Symposium on*, pages 601–605. IEEE, 2012.

K. Kothapalli, M. Onus, C. Scheideler, and C. Schindelhauer. Distributed coloring in $\mathcal{O}(\sqrt{\log N})$-bits. In *Proc. of IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, 2006.

S. T. Hedetniemi, D. P. Jacobs, and P. K. Srimani. Fault tolerant distributed coloring algorithms that stabilize in linear time. In *Proceedings of the IPDPS-2002 Workshop on Advances in Parallel and Distributed Computational Models*, pages 1–5, 2002.

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

B. Kauffmann, F. Baccelli, A. Chaintreau, V. Mhatre, K. Papagiannaki, and C. Diot. Measurement-based self organization of interfering 802.11 wireless access networks. In *INFOCOM 2007. 26th IEEE International Conference on Computer Communications. IEEE*, pages 1451–1459. IEEE, 2007.

Julien Herzen, Ruben Merz, and Patrick Thiran. Distributed spectrum assignment for home WLANs. In *INFOCOM, 2013 Proceedings IEEE*, pages 1573–1581. Ieee, 2013a.

Julien Herzen, Ruben Merz, and Patrick Thiran. SAW: Spectrum assignment for WLANs. In *ACM S3 2013*, number EPFL-CONF-189777, 2013b.

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.

Márió 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.

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.

B. Kauffmann, F. Baccelli, A. Chaintreau, K. Papagiannaki, C. Diot, et al. Self organization of interfering 802.11 wireless access networks. 2005.

Jaume Barcelo, Boris Bellalta, Cristina Cano, and Miquel Oliver. Learning-BEB: Avoiding collisions in WLAN. *Other IFIP Publications*, (1), 2011.

Leonid Barenboim and Michael Elkin. Distributed Δ+ 1-coloring in linear in Δ time. In *Proceedings of the 41st annual ACM symposium on Theory of computing*, pages 111–120. ACM, 2009.

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

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.

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.

Arik Motskin, Tim Roughgarden, Primoz Skraba, and Leonidas J. Guibas. Lightweight coloring and desynchronization for networks. In *INFOCOM*, pages 2383–2391, 2009.

M. Fang, D. Malone, K. R. Duffy, and Douglas J. Leith. Decentralised learning macs for collision-free access in wlans. *Wireless Networks*, pages 1–16, 2010.

P. Clifford and Douglas J. Leith. Channel dependent interference and decentralized colouring. *Network Control and Optimization*, pages 95–104, 2007.

D. J. Leith, P. Clifford, V. Badarla, and D. Malone. WLAN channel selection without communication. *Computer Networks*, 2012.

BIBLIOGRAPHY FOR CHAPTER 2

K. Duffy, David Malone, and Douglas J. Leith. Modeling the 802.11 distributed coordination function in non-saturated conditions. *IEEE Communications Letters*, 9(8):715–717, August 2005. ISSN 1089-7798. doi: 10.1109/LCOMM.2005.1496592. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1496592.

Douglas J. Leith, Vijay Subramanian, and Ken Duffy. Log-convexity of rate region in 802.11e WLANs. *IEEE Communications Letters*, 14(1): 57–59, January 2010. ISSN 1089-7798. doi: 10.1109/LCOMM.2010.01.091154. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5370273.

Joseph Dunn, Michael Neufeld, Anmol Sheth, Dirk Grunwald, and John Bennett. A Practical Cross-Layer Mechanism For Fairness in 802.11 Networks. *Mobile Networks and Applications*, 11 (1):37–45, December 2005. ISSN 1383-469X. doi: 10.1007/s11036-005-4459-z. URL http://www.springerlink.com/index/10.1007/s11036-005-4459-z.

Albert Banchs, Pablo Serrano, and Huw Oliver. Proportional fair throughput allocation in multirate IEEE 802.11e wireless LANs. *Wireless Networks*, 13(5):649–662, June 2006. ISSN 1022-0038. doi: 10.1007/s11276-006-6972-9. URL http://www.springerlink.com/index/10.1007/s11276-006-6972-9.

R. T. Rockafellar. *Convex analysis*. 1997. ISBN 0691080690. URL http://books.google.com/books?hl=en&amp;lr=&amp;id=1TiOka9bx3sC&amp;oi=fnd&amp;pg=PR7&amp;dq=Convex+Analysis&amp;ots=HpVNXCKW7b&amp;sig=MJxPD3Ik6lUS6lgzRs-ltNimnT0.

V Subramanian. Convexity conditions for 802.11 wlans. 2012.

BIBLIOGRAPHY FOR CHAPTER 3

D. J. Leith, P. Clifford, V. Badarla, and D. Malone. WLAN channel selection without communication. *Computer Networks*, 2012.

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

H. Holma and A. Toskala. *WCDMA for UMTS - radio access for third generation mobile communications*. Chichester, 2000.

C. R. Chang, J. Z. Wan, and M. F. Yee. PN offset planning strategies for non-uniform CDMA networks. In *Vehicular Technology Conference, 1997 IEEE 47th*, volume 3, pages 1543–1547.

Y. H. Jung and Y. H. Lee. Scrambling code planning for 3GPP W-CDMA systems. In *Proceedings of IEEE vehicular technology conference*, pages 2431–2434, Rhodes, Greece, Spring 2001.

David Soldani and Ivan Ore. Self-optimizing neighbor cell list for utra fdd networks using detected set reporting. In *Vehicular Technology Conference, 2007. VTC2007-Spring. IEEE 65th*, pages 694–698. IEEE, 2007.

J. Guey, Y. Wang, and J. Cheng. Improving the robustness of Target Cell Search in WCDMA Using Interference Cancellation. 2005.

3GPP TR 25.967. Home Node B (HNB) Radio Frequency (RF) requirements (FDD). Ver 10(version 10), 2011.

S.J. Fortune, D.M. Gay, B.W. Kernighan, O. Landron, R.A. Valenzuela, and M.H. Wright. WiSE design of indoor wireless systems: practical computation and optimization. *Computational Science & Engineering, IEEE*, 2(1):58–68, 1995.

3GPP TS25.101. UE Radio transmission and Reception (FDD). 2004.

3GPP TR 36.814. Evolved Universal Terrestrial Radio Access (E-UTRA); Further advancements for E-UTRA physical layer aspects. Rel 9, 2010.

S. Kourtis. Code planning strategy for UMTS-FDD networks. In *Vehicular Technology Conference Proceedings, 2000. VTC 2000-Spring Tokyo. 2000 IEEE 51st*, volume 2, pages 815–819. IEEE, 2000.

D.P. Aguilar. *A framework for evaluating the computational aspects of mobile phones*. PhD thesis, University of South Florida, 2008.

P. Sapiano. Discovering neighbouring femto cells, August 2010. URL http://www.freepatentsonline.com/EP2214434A1.html.

David Asher Levin, Yuval Peres, and Elizabeth Lee Wilmer. *Markov chains and mixing times*. AMS, 2009.

F.W. Gehring and P.R. Halmos. *Finite Markov Chains*, chapter 3: Absorbing Markov Chains, page 224. Springer-Verlag, 1976.

J. Edwards. Implementation of network listen modem for WCDMA femtocell. In *Cognitive Radio and Software Defined Radios: Technologies and Techniques, 2008 IET Seminar on*, pages 1–4. IET, 2008.

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.

BIBLIOGRAPHY FOR CHAPTER 4

M. Fang, D. Malone, K. R. Duffy, and Douglas J. Leith. Decentralised learning macs for collision-free access in wlans. *Wireless Networks*, pages 1–16, 2010.

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

O. Dousse. Percolation in directed random geometric graphs. In *Information Theory Proceedings (ISIT), 2012 IEEE International Symposium on*, pages 601–605. IEEE, 2012.

G. R. Grimmett. *Percolation*, volume 321. Springer, 1999.

3GPP TS25.101. UE Radio transmission and Reception (FDD). 2004.

wigle.net, 2010. URL http://www.wigle.net/.

IEEE Standard for Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications , Nov. 1997. P802.11, 1997.

A. Goldsmith. *Wireless Communications*. Cambridge university press, 2005.

BIBLIOGRAPHY FOR CHAPTER 5

Jaume Barcelo, Boris Bellalta, Cristina Cano, and Miquel Oliver. Learning-BEB: Avoiding collisions in WLAN. *Other IFIP Publications*, (1), 2011.

Arik Motskin, Tim Roughgarden, Primoz Skraba, and Leonidas J. Guibas. Lightweight coloring and desynchronization for networks. In *INFOCOM*, pages 2383–2391, 2009.

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

Márió 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.

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.

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.

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.

Joan Melià-Seguí, Rafael Pous, Anna Carreras, Marc Morenza-Cinos, Raúl Parada, Zeinab Liaghat, and Ramir 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.

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

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.

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

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

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.

Joan Melià-Seguí, Joaquin Garcia-Alfaro, and Jordi Herrera-Joancomartí. On the similarity of commercial EPC gen2 pseudo-random number generators. *Transactions on Emerging Telecommunications Technologies*, 2012.

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.

BIBLIOGRAPHY FOR CHAPTER 6

J. Edwards. Implementation of network listen modem for WCDMA femtocell. In *Cognitive Radio and Software Defined Radios: Technologies and Techniques, 2008 IET Seminar on*, pages 1–4. IET, 2008.

A. Checco, R. Razavi, Douglas J. Leith, and H. Claussen. Self-configuration of scrambling codes for WCDMA small cell networks.

INDEX

COLOPHON

This document was typeset using the typographical look-and-feel `classicthesis` developed by André Miede. The style was inspired by Robert Bringhurst's seminal book on typography "*The Elements of Typographic Style*".

Hermann Zapf's *Palatino* and *Euler* type faces (Type 1 PostScript fonts *URW Palladio L* and *FPL*) are used. The "typewriter" text is typeset in *Bera Mono*, originally developed by Bitstream, Inc. as "Bitstream Vera". (Type 1 PostScript fonts were made available by Malte Rosenau and Ulrich Dirr.)

Final Version as of December 28, 2014.