Combinatorics, Probability and Computing (2009) 00, 000–000. © 2009 Cambridge University Press
DOI: 10.1017/S0000000000000000 Printed in the United Kingdom

Online conflict-free colouring for hypergraphs[†]

A. BAR-NOY[1], P. CHEILARIS[2‡],
S. OLONETSKY[3] and S. SMORODINSKY[4]

[1] Computer and Information Science Department, Brooklyn College, USA

[2] Center for Advanced Studies in Mathematics, Ben-Gurion University, Israel

[3] School of Computer Science, Tel-Aviv University, Israel

[4] Department of Mathematics, Ben-Gurion University, Israel

amotz@sci.brooklyn.cuny.edu, panagiot@math.bgu.ac.il, olonetsk@post.tau.ac.il, shakhar@math.bgu.ac.il

We provide a framework for online conflict-free colouring any hypergraph. We introduce the notion of a degenerate hypergraph, which characterises hypergraphs that arise in geometry. We use our framework to obtain an efficient randomised online algorithm for conflict-free colouring any k-degenerate hypergraph with n vertices. Our algorithm uses $O(k \log n)$ colours with high probability and this bound is asymptotically optimal. Moreover, our algorithm uses $O(k \log k \log n)$ random bits with high probability. We introduce algorithms that are allowed to perform a few recolorings of already coloured points. We provide deterministic online conflict-free colouring algorithms for points on the line with respect to intervals and for points on the plane with respect to halfplanes (or unit disks) that use $O(\log n)$ colours and perform a total of at most $O(n)$ recolorings.

1. Introduction

A *hypergraph* is a pair (V, E), where V is a finite set and $E \subseteq \mathcal{P}(V)$. The set V is called the *ground set* or the *vertex set* and the elements of E are called *hyperedges*. A *proper k*-colouring of a hypergraph $H = (V, E)$, for some positive integer k, is a function $C\colon V \to \{1, 2, \ldots, k\}$ such that no $S \in E$ with $|S| \geq 2$ is monochromatic. Let $\chi(H)$ denote the minimum integer k for which H has a k-colouring; $\chi(H)$ is called the *chromatic number* of H. A *conflict-free* colouring of H is a colouring of V with the further restriction that for any hyperedge $S \in E$ there exists a vertex $v \in S$ with a unique colour (i.e., no other vertex of S has the same colour as v). Both proper colouring and conflict-free colouring of hypergraphs are generalisations of vertex colouring of graphs (the definition coincides when the underlying hypergraph is a simple graph). Therefore, computing such colourings is at least as hard as computing vertex colourings for simple graphs.

The study of conflict-free colourings originated in the work of Even et al. [11] and Smorodinsky [20] who were motivated by the problem of frequency assignment in cellular networks.

[†] A preliminary version of this work appeared in the 34th International Colloquium on Automata, Languages and Programming (ICALP 2007).

[‡] The second author was partially supported, during his stay at the Alfréd Rényi Institute of Mathematics, by the Discrete and Convex Geometry project (MTKD-CT-2005-014333), in the framework of the European Community's "Structuring the European Research Area" programme.

Cellular networks are heterogeneous networks with two different types of nodes: *base stations* (that act as servers) and *clients*. Base stations are interconnected by an external fixed backbone network whereas clients are connected only to base stations. Connections between clients and base stations are implemented by radio links. Fixed frequencies are assigned to base stations to enable links to clients. Clients continuously scan frequencies in search of a base station with good reception. The fundamental problem of frequency assignment in such cellular networks is to assign frequencies to base stations so that every client, located within the receiving range of at least one station, can be served by some base station, in the sense that the client is located within the range of the station and no other station within its reception range has the same frequency (such a station would be in *conflict* with the given station due to mutual interference). The goal is to minimise the number of assigned frequencies ("colours") since the frequency spectrum is limited and costly.

Suppose we are given a set of n base stations, also referred to as *antennas*. Assume, for simplicity, that the area covered by a single antenna is given as a disk in the plane. Namely, the location of each antenna and its radius of transmission is fixed and is given (the transmission radii of the antennas are not necessarily equal). Even et al. [11] showed that one can find an assignment of frequencies to the antennas with a total of at most $O(\log n)$ frequencies such that each antenna is assigned one of the frequencies and the resulting assignment is free of conflicts, in the preceding sense. Furthermore, it was shown that this bound is worst-case optimal. Let $\mathcal{R}$ be a set of regions in the plane. For a point $p \in \cup_{r\in\mathcal{R}} r$, let $r(p) = \{r \in \mathcal{R} \mid p \in r\}$. Let $H(\mathcal{R})$ denote the hypergraph $(\mathcal{R}, \{r(p) \mid p \in \cup_{r\in\mathcal{R}}\})$. We say that $H(\mathcal{R})$ is the hypergraph *induced* by $\mathcal{R}$. Thus, Even et al. [11] showed that any hypergraph induced by a family $\mathcal{R}$ of n discs in the plane admits a conflict-free colouring with only $O(\log n)$ colours and that this bound is tight in the worst case. Furthermore, such a colouring can be found in deterministic polynomial time. However, in [11] it was also shown that finding the minimum number of colours needed to conflict-free colour a given collection of discs is NP-hard even when all discs are congruent, and an $O(\log n)$ approximation-ratio algorithm is provided. The results of [11] were further extended in [14] by combining more involved probabilistic and geometric ideas. The main result of [14] is a general randomised algorithm which conflict-free colours any set of n "simple" regions (not necessarily convex) whose union has "low" complexity, using a "small" number of colours. Many more variants and extensions were studied in several recent works (see, e.g., [8, 10, 11, 14, 18, 20, 21, 1, 16, 5, 6, 9]).

To capture a dynamic scenario where antennas can be added to the network, Chen et al. [8] initiated the study of online conflict-free colouring of hypergraphs. They considered a very simple hypergraph H which has its vertex set represented as a set P of n points on the line and its hyperedge set consists of the intersections of P with every possible interval. The set $P \subset \mathbb{R}$ is revealed by an adversary online: Initially, P is empty, and the adversary inserts points into P, one point at a time. Let $P(t)$ denote the set of points revealed after the t-th point has been inserted. Each time a point is inserted, the algorithm needs to assign a colour $C(p)$ to it, which is a positive integer. Once the colour has been assigned to p, it cannot be changed in the future. The colouring should remain conflict-free at all times. That is, for any interval I and for any time t there is a colour that appears exactly once in $I \cap P(t)$. Among other results, the authors of [12] provided a randomised algorithm for online conflict-free colouring n points on the line with $O(\log n \log\log n)$ colours with high probability. Their algorithm assumes that the adversary is oblivious in the sense that it does not have access to the random bits used by the probabilistic algorithm. They also provided a deterministic algorithm for online conflict-free colouring n points on the line with $\Theta(\log^2 n)$ colours in the worst case.

1.1. An online conflict-free colouring framework

In this work, we investigate the most general form of online conflict-free colouring applied to arbitrary hypergraphs. Suppose that the vertices of an underlying hypergraph $H = (V, E)$ are given online by an adversary. At every given time step t, a new vertex $v_t \in V$ is given and the algorithm must assign v_t a colour such that the colouring is a valid conflict-free colouring of the hypergraph that is induced by the vertices $V_t = \{v_1, \dots, v_t\}$ (see the exact definition in section 2). Once v_t is assigned a colour, that colour cannot be changed in the future. The goal is to find an algorithm that minimises the maximum total number of colours used (where the maximum is taken over all permutations of the set V).

We present a general framework for online conflict-free colouring any hypergraph. Interestingly, this framework is a generalisation of some known colouring algorithms. For example the unique maximum greedy algorithm of [8] can be described as a special case of our framework. Also, when the underlying hypergraph is a simple graph then the first-fit greedy online algorithm is another special case of our framework. Based on this framework, we introduce a *randomised algorithm* and show that the maximum number of colours used is a function of the some natural parameter of the hypergraph which we call the *degeneracy* of the hypergraph. We define the notion of a k-degenerate hypergraph as a generalisation of the same notion for simple graphs. Specifically we show that if the hypergraph is k-degenerate, then our algorithm uses $O(k \log n)$ colours with high probability against an *oblivious adversary* (see [4]). An oblivious adversary has to commit to a specific input sequence before revealing the first vertex to the algorithm without knowing the random bits that the algorithm is about to use.

As demonstrated in [8], the problem of online conflict-free colouring the very special hypergraph induced by points on the real line with respect to intervals is highly non-trivial. The best randomised online conflict-free colouring algorithm of [12] uses $O(\log n \log\log n)$ colours. Chen, Kaplan and Sharir [7, 9] studied the hypergraph induced by points in the plane with respect to intervals, halfplanes, and unit disks and obtained randomised online conflict-free colouring algorithms that use $O(\log n)$ colours with high probability. Our algorithm is more general, also uses only $O(\log n)$ colours, and it has better constant of proportionality in front of the logarithmic factor than the algorithms of [7, 8, 9]. An interesting evidence to our algorithm being fundamentally different from the ones in [7, 8, 9], when used for the hypergraphs mentioned above, is that our algorithm uses exponentially fewer random bits; the algorithms of [7, 8, 9] use $\Omega(n)$ random bits and our algorithm uses $O(\log n)$ random bits with high probability.

Another interesting corollary of our result is that we obtain a randomised online colouring for k-inductive graphs with $O(k \log n)$ colours with high probability. This case was studied by Irani [15] who showed that the first-fit greedy algorithm achieves the same bound deterministically (a very short proof of this fact can be found in [22]).

1.2. Deterministic online conflict-free colouring with recolouring

We initiate the study of online conflict-free colouring where at each step, in addition to the assignment of a colour to the newly inserted point, we allow some recolouring of other points. The bi-criteria goal is to minimise the total number of recolorings done by the algorithm and the total number of colours used by the algorithm. We introduce an online algorithm for conflict-free colouring points on the line with respect to intervals, where we recolour at most one point at each step. Our algorithm uses $\Theta(\log n)$ colours. This is in contrast with the $O(\log^2 n)$ colours used by the best known deterministic algorithm by [8] that does not recolour points. We also provide an online algorithm for conflict-free colouring points on the plane with respect to half-planes that uses $\Theta(\log n)$ colours and the total number of recolorings is $O(n)$. For this problem no deterministic algorithm that uses a polylogarithmic number of colours was known before.

From the application point of view, there is motivation to study this recolouring model. The frequency spectrum is quite expensive, so a solution which uses a logarithmic number of colours is desirable. On the other hand excessive recolouring is not desirable, because if a base station is given another colour there is a disruption of service for all agents connected to it.

Organisation. In section 2 we define the notion of a k-degenerate hypergraph. In section 3 we present the general framework for online conflict-free colouring of hypergraphs. In section 4 we introduce the randomised algorithm derived from the framework. In section 5 we show deterministic online algorithms for intervals and halfplanes with recolouring. In section 6 we describe the results for several hypergraphs that arise in geometric contexts. Finally, in section 7 we conclude with a discussion and some open problems.

2. Preliminaries

We start with some basic definitions.

Definition. Let $H = (V, E)$ be a hypergraph. For a subset $V' \subseteq V$ let $H(V')$ be the hypergraph (V', E') where $E' = \{e \cap V' | e \in E\}$. We say that $H(V')$ is the hypergraph *induced* by V'.

Definition. For a hypergraph $H = (V, E)$, the *Delaunay graph* $G(H)$ is the simple graph $G = (V, F)$ where the edge set F is defined as $F = \{\{x, y\} \mid \{x, y\} \in E\}$ (i.e., G is the graph on the vertex set V whose edges consist of all hyperedges in H of cardinality two).

A common graph theoretic definition follows.

Definition. A graph $G = (V, E)$ is called k-inductive (or k-degenerate) for some positive integer k, if every (vertex-induced) subgraph of G has a vertex of degree at most k.

We sensibly extend to a similar definition for hypergraphs.

Definition (degenerate hypergraph). Let $k > 0$ be a fixed integer and let $H = (V, E)$ be a hypergraph on the n vertices $v_1, \ldots, v_n$. For a permutation $\pi \colon \{1, \ldots, n\} \to \{1, \ldots, n\}$ define the n partial sums, indexed by $t = 1, \ldots, n$,

$$S_t^\pi = \sum_{j=1}^{t} d(v_{\pi(j)}),$$

where

$$d(v_{\pi(j)}) = \left|\{i < j \mid \{v_{\pi(i)}, v_{\pi(j)}\} \in G(H(\{v_{\pi(1)}, ..., v_{\pi(j)}\}))\}\right|,$$

that is, $d(v_{\pi(j)})$ is the number of neighbours of $v_{\pi(j)}$ in the Delaunay graph of the hypergraph induced by $\{v_{\pi(1)}, ..., v_{\pi(j)}\}$. Assume that for all permutations π and for every $t \in \{1, \ldots, n\}$ we have

$$S_t^{\pi} \leq kt. \tag{2.1}$$

Then, we say that H is *k-degenerate*.

3. A framework for online conflict-free colouring

Let $H = (V, E)$ be any hypergraph. Our goal is to define a framework that colours the vertices of V in an online fashion, i.e., when the vertices of V are revealed by an adversary one at a time. At each time step t, the algorithm must assign a colour to the newly revealed vertex v_t. This colour cannot be changed in future times $t' > t$. The colouring has to be conflict-free for all the induced hypergraphs $H(V_t)$ with $t = 1, \ldots, n$, where $V_t \subseteq V$ is the set of vertices revealed by time t.

For a fixed positive integer h, let $A = \{a_1, \ldots, a_h\}$ be a set of h *auxiliary* colours. This auxiliary colours set should not be confused with the set of *main* colours used for the conflict-free colouring: $\{1, 2, \ldots\}$. Let $f\colon \mathbb{N}^+ \to A$ be some fixed function. In the following, we define the framework that depends on the choice of the function f and the parameter h.

A table (to be updated online) is maintained with row entries indexed by the variable i with range in $\mathbb{N}^+$. Each row entry i at time t is associated with a subset $V_t^i \subseteq V_t$ in addition to an auxiliary proper non-monochromatic colouring of $H(V_t^i)$ with at most h colours. We say that $f(i)$ is the auxiliary colour that *represents* entry i in the table. At the beginning all entries of the table are empty. Suppose all entries of the table are updated until time $t - 1$ and let v_t be the vertex revealed by the adversary at time t. The framework first checks if an auxiliary colour can be assigned to v_t such that the auxiliary colouring of V_{t-1}^1 together with the colour of v_t is a proper non-monochromatic colouring of $H(V_{t-1}^1 \cup \{v_t\})$. Any (proper non-monochromatic) colouring procedure can be used by the framework. For example a first-fit greedy method in which all colours in the order $a_1, \ldots, a_h$ are checked until one is found. If such a colour cannot be found for v_t, then entry 1 is left with no changes and the process continues to the next entry. If however, such a colour can be assigned, then v_t is added to the set V_{t-1}^1. Let c denote such an auxiliary colour assigned to v_t. If this colour is the same as $f(1)$ (the auxiliary colour that represents entry 1), then the final colour in the online conflict-free colouring of v_t is 1 and the updating process for the t-th vertex stops. Otherwise, if an auxiliary colour cannot be found or if the assigned auxiliary colour is not the same as $f(1)$, then the updating process continues to the next entry. The updating process stops at the first entry i for which v_t is both added to V_t^i and the auxiliary colour assigned to v_t is the same as $f(i)$. Then, the main colour of v_t in the final conflict-free colouring is set to i.

It is possible that v_t never gets a final colour. In this case we say that the framework does not halt. However, termination can be guaranteed by imposing some restrictions on the auxiliary colouring method and the choice of the function f. For example, if first-fit is used for the auxiliary colourings at any entry and if f is the constant function $f(i) = a_1$, for all i, then the framework is guaranteed to halt for any time t. An example of the framework for conflict-free colouring with respect to intervals is given in the example in section 6. In section 4 we derive a randomised online algorithm based on this framework. This algorithm always halts, or to be more precise halts with probability 1, and moreover it halts after a “small” number of entries with high probability. We prove that the above framework produces a valid conflict-free colouring in case it halts.

Lemma 3.1. *If the above framework halts for every vertex v_t then it produces a valid online conflict-free colouring of H.*

Proof. Let $H(V_t)$ be the hypergraph induced by the vertices already revealed at time t. Let S be a hyperedge in this hypergraph and let j be the maximum integer for which there is a vertex v of S coloured with j. We claim that exactly one such vertex in S exists. Assume to the contrary that there is another vertex v' in S coloured with j. This means that at time t both vertices v and v' were present at entry j of the table (i.e., $v, v' \in V_t^j$) and that they both got an auxiliary colour (in the auxiliary colouring of the set V_t^j) which equals $f(j)$. However, since the auxiliary colouring is a proper non-monochromatic colouring of the induced hypergraph at entry j, $S \cap V_t^j$ is not monochromatic so there must exist a third vertex $v'' \in S \cap V_t^j$ that was present at entry j and was assigned an auxiliary colour different from $f(j)$. Thus, v'' got its final colour in an entry greater than j, a contradiction to the maximality of j in the hyperedge S. This completes the proof of the lemma. □

The above algorithmic framework can also describe some well-known deterministic algorithms. For example, if first-fit is used for auxiliary colourings and f is the constant function, $f(i) = a_1$, for all i, then:

- If the input hypergraph is induced by points on a line with respect to intervals as in example 3 then the algorithm derived from the framework becomes identical to the unique maximum greedy algorithm described and analysed in [8].
- If the input is a k-inductive graph (also called k-degenerate graph), the derived algorithm is identical to the first-fit greedy algorithm for colouring graphs online. The performance of the first-fit greedy algorithm for restricted classes of graphs has been analysed in several papers [13, 17, 15]. Especially for k-inductive graphs, the first-fit greedy algorithm is analysed by Irani [15], who proved that it uses $O(k \log n)$ colours. Our framework can be used to give an alternative simpler proof of the aforementioned result (see [22] for details).

4. An online randomised conflict-free colouring algorithm

We devise a randomised online conflict-free colouring algorithm in the oblivious adversary model. In this model, the adversary has to commit to a permutation according to the order of which the vertices of the hypergraph are revealed to the algorithm. Namely, the adversary does not have access to the random bits that are used by the algorithm. The algorithm always produces a valid colouring and the number of colours used is related to the degeneracy of the underlying hypergraph in a manner described in the following theorem.

Theorem 4.1. *Let $H = (V, E)$ be a k-degenerate hypergraph on n vertices. Then, there exists a randomised online conflict-free colouring algorithm for H which uses at most $O(\log_{1+\frac{1}{4k+1}} n) = O(k \log n)$ colours with high probability against an oblivious adversary.*

The algorithm is based on the framework of section 3. In order to define the algorithm, we need to state what is (a) the set of auxiliary colours of each entry, (b) the function f, and (c) the algorithm we use for the auxiliary colouring at each entry. We use the set of auxiliary colours $A = \{a_1, \ldots, a_{2k+1}\}$. For each entry i, the representing colour $f(i)$ is chosen uniformly at random from A. We use a first-fit algorithm for the auxiliary colouring.

Our assumption on the hypergraph H (being k-degenerate) implies that at least half of the vertices up to time t that *reached* entry i (but not necessarily added to entry i), denoted by $X^t_{\geq i}$, have been actually given some auxiliary colour at entry i (that is, $\left|V^i_t\right| \geq \frac{1}{2}\left|X^t_{\geq i}\right|$). This is due to the fact that at least half of those vertices v_t have at most $2k$ neighbours in the Delaunay graph of the hypergraph induced by $X^{t-1}_{\geq i}$ (since the sum of these quantities is at most $k\left|X^t_{\geq i}\right|$ and since $V^i_t \subseteq X^t_{\geq i}$). Therefore, since we have $2k+1$ colours available, there is always an available colour to assign to such a vertex. The following lemma shows that if we use one of these available colours then the updated colouring is indeed a proper non-monochromatic colouring of the corresponding induced hypergraph as well.

Lemma 4.2. *Let $H = (V, E)$ be a k-degenerate hypergraph and let V^j_t be the subset of V at time t and at level j as produced by the above algorithm. Then, for any j and t if v_t is assigned a colour distinct from all its neighbours in the Delaunay graph $G(H(V^j_t))$ then this colour together with the colours assigned to the vertices V^j_{t-1} is also a proper non-monochromatic colouring of the hypergraph $H(V^j_t)$.*

Proof. By induction on t. The induction hypothesis is that $H(V^j_{t-1})$ is properly (that is, non-monochromatically) coloured by the auxiliary colouring. Let v_t be the vertex added to the hypergraph induced by the j-th entry at time t. Any hyperedge S that contains at least two vertices of V^j_{t-1} or does not contain v_t is not monochromatic by the induction hypothesis. Thus, we are only concerned with hyperedges of cardinality two that contain v_t and exactly one vertex of V^j_{t-1}. However, we assumed that v_t obtained a colour that is distinct from any vertex u such that $\{u, v_t\}$ is a hyperedge of $H(V^j_t)$ (Those are exactly the neighbours of v_t in the corresponding Delaunay graph). Thus, any such hyperedge $\{u, v_t\}$ is also not monochromatic. This completes the inductive step and hence the proof of the lemma. □

We also prove that for every vertex v_t, our algorithm always halts, or more precisely halts with probability 1.

Proposition 4.3. *For every vertex v_t, the algorithm halts with probability 1.*

Proof. In order for the framework not to halt for some vertex v_t, it must be the case that vertex v_t reaches every entry $i \in \mathbb{N}^+$ and in every entry i the auxiliary colour of v_t is different from $f(i)$. If an entry is empty before time t and v_t reaches that entry, then v_t gets the auxiliary colour a_1 in that entry and the probability that v_t does not get a main colour in that entry is $1 - h^{-1}$, where

$h = 2k + 1$ is the number of auxiliary colours. The aforementioned events are independent for different empty entries. At time t, all but at most $t - 1$ entries are empty. The above discussion implies the following.

$$\Pr[\text{algorithm does not halt for } v_t] =$$
$$\Pr[\text{algorithm does not assign a main colour to } v_t \text{ in any entry}] \le$$
$$\Pr[\text{algorithm does not assign a main colour to } v_t \text{ in any empty entry}] =$$
$$\Pr[\bigcap_{i\colon \text{empty entry}} (\text{algorithm does not assign a main colour to } v_t \text{ in entry } i)] =$$
$$\prod_{i\colon \text{empty entry}} \Pr[\text{algorithm does not assign a main colour to } v_t \text{ in entry } i] =$$
$$\prod_{i\colon \text{empty entry}} (1 - h^{-1}) = \lim_{j\to\infty} (1 - h^{-1})^j = 0$$

and therefore $\Pr[\text{algorithm halts for } v_t] = 1$. □

We proceed to the analysis of the number of colours used by the algorithm, proving theorem 4.1.

Lemma 4.4. *Let $H = (V, E)$ be a hypergraph and let C be a colouring produced by the above algorithm on an online input $V = \{v_t\}$ for $t = 1, \ldots, n$. Let X_i (respectively $X_{\ge i}$) denote the random variable counting the number of points of V that were assigned a final colour at entry i (respectively a final colour at some entry $\ge i$). Let $\mathbf{E}_i = \mathbf{E}[X_i]$ and $\mathbf{E}_{\ge i} = \mathbf{E}[X_{\ge i}]$ (note that $X_{\ge i+1} = X_{\ge i} - X_i$). Then:*

$$\mathbf{E}_{\ge i} \le \left(\frac{4k+1}{4k+2}\right)^{i-1} n.$$

Proof. By induction on i. The case $i = 1$ is trivial. Assume that the statement holds for i. To complete the induction step, we need to prove that $\mathbf{E}_{\ge i+1} \le (\frac{4k+1}{4k+2})^i n$. By the conditional expectation formula, we have for any two random variables X, Y that $\mathbf{E}[X] = \mathbf{E}[\mathbf{E}[X \mid Y]]$. Thus,

$$\mathbf{E}_{\ge i+1} = \mathbf{E}[\mathbf{E}[X_{\ge i+1} \mid X_{\ge i}]] = \mathbf{E}[\mathbf{E}[X_{\ge i} - X_i \mid X_{\ge i}]] = \mathbf{E}[X_{\ge i} - \mathbf{E}[X_i \mid X_{\ge i}]].$$

It is easily seen that $\mathbf{E}[X_i \mid X_{\ge i}] \ge \frac{1}{2}\frac{X_{\ge i}}{2k+1}$ since at least half of the vertices of $X_{\ge i}$ got an auxiliary colour by the above algorithm. Moreover each of those elements that got an auxiliary colour had probability $\frac{1}{2k+1}$ to get the final colour i. This is the only place where we need to assume that the adversary is oblivious and does not have access to the random bits. Thus,

$$\mathbf{E}[X_{\ge i} - \mathbf{E}[X_i \mid X_{\ge i}]] \le \mathbf{E}[X_{\ge i} - \frac{1}{2(2k+1)} X_{\ge i}] = \frac{4k+1}{4k+2}\mathbf{E}[X_{\ge i}] \le \left(\frac{4k+1}{4k+2}\right)^i n,$$

by linearity of expectation and by the induction hypotheses. This completes the proof of the lemma. □

Lemma 4.5. *The expected number of colours used by the algorithm is at most* $\log_{\frac{4k+2}{4k+1}} n + 1$.

Proof. Let I_i be the indicator random variable for the following event: some points are coloured with a main colour in entry i. We are interested in the number of colours used, that is $Y := \sum_{i=1}^{\infty} I_i$. Let $b(k,n) = \log_{\frac{4k+2}{4k+1}} n$. Then,

$$\mathbf{E}[Y] = \mathbf{E}[\sum_{1\le i} I_i] \le \mathbf{E}[\sum_{1\le i\le b(k,n)} I_i] + \mathbf{E}[X_{\ge b(k,n)+1}] \le b(k,n) + 1,$$

by Markov's inequality and lemma 4.4. □

We notice that:

$$b(k,n) = \frac{\ln n}{\ln \frac{4k+2}{4k+1}} \le (4k+2)\ln n = O(k\log n).$$

We also have the following concentration result:

$$\Pr[\text{more than } c\cdot b(k,n) \text{ colours are used}] = \Pr[X_{\ge c\cdot b(k,n)+1} \ge 1] \le \mathbf{E}_{\ge c\cdot b(k,n)+1} \le \frac{1}{n^{c-1}},$$

by Markov's inequality and by lemma 4.4.

This completes the performance analysis of our algorithm.

Remark. In the above description of the algorithm, all the random bits are chosen in advance (by deciding the values of the function f in advance). However, one can be more efficient and calculate the entry $f(i)$ only at the first time we need to update entry i, for any i. Since at each entry we need to use $O(\log k)$ random bits and we showed that the number of entries used is $O(k\log n)$ with high probability then the total number of random bits used by our algorithm is $O(k\log k\log n)$ with high probability.

5. Deterministic online algorithms with recolouring

In this section, we relax the requirement that an online algorithm has to commit to the colour of every point, by allowing the algorithm to recolour a "few" of the points that have appeared in the past. Our goal is to find deterministic online algorithms that use a logarithmic number of colours and perform a total number of recolorings which is linear in n. We manage to find such algorithms with respect to intervals and halfplanes. The algorithm for halfplanes relies on an algorithm that colours points on a disk with respect to circular arcs, where the adversary can additionally ask the algorithm to substitute a set of consecutive points on the disk with a single point (we call this a substitution move). As always, the colouring must remain conflict-free at all times.

5.1. An $O(\log n)$ colours algorithm for intervals

We consider the problem of conflict-free colouring a set of points on a line (that are revealed in an online fashion) and the conflict-free property has to hold at any given time for any interval. We describe a deterministic online conflict-free colouring algorithm for this case that is allowed to recolour just a single old point during each insertion of a new point. The algorithm is based on the framework developed in section 3 where we use 3 auxiliary colours $\{a,b,c\}$ and f is the constant function $f(l) = a$, for every l. We refer to points coloured with b or c as d-points. In order to have only a logarithmic number of entries, we slightly modify the framework (using a recolouring procedure) such that the number of points coloured with a in each entry of the table is at least one third of the total points that reach that entry. To achieve this goal, our algorithm maintains the following invariant in every level: There are at most two d-points between every pair of points coloured with a (i.e., between every pair that are consecutively coloured a among the a-points). Therefore, at least a third of the points at each entry get colour a, and two thirds are deferred for colouring in a higher entry. The total number of colours is at most $\log_{3/2} n + 1$. When a new point p arrives, it is coloured according to the following algorithm, starting from entry 1:

- If p is not adjacent to a point coloured with an auxiliary colour a then p is assigned auxiliary colour a and gets its final colour in that entry.
- We colour point p with b or c greedily as long as it does not break the invariant that between any two consecutive a's we have at most two d-points.
- It remains to handle the case where the new point p has a point coloured with a on one side and a point, say q, coloured with d on the other side, such that q has no adjacent point coloured with a. We assign to p the auxiliary colour of q (thus it is a d-point) in the current entry and in all higher entries for which q obtained an auxiliary colour and assign to it the main colour of q, and we recolour q with the auxiliary colour a (and delete the corresponding appearance of it in all higher entries of the table), and thus we recolour q with the main colour of the current entry. At this point all points have an assignment of main colours. It is not hard to check that when we recolour a point then we do not violate the invariants at any entry: Let ℓ be the entry that caused recolouring, all entries before it remain the same, the change in the entry ℓ does not break invariants, all other entries remain the same except that point p appears there instead of point q that was there before and there are no points between p and q that appear in an entry higher than ℓ.

An example of a run of the recolouring algorithm is shown in figure 1 for an input permutation $\pi = 3754612$. Vertex v_t (that is, the number π_t) appears at time t, where t ranges from 1 to 7. The first row of the table represents the order in which points appear, the last row of the table shows the current colouring. At every time step of the run, points whose colours were changed (a new colour, or a recolouring) by the last insertion are marked with bold. Recolorings happen at $t = 3$ for v_2, at $t = 5$ for v_3, and at $t = 7$ for v_6.

It can be easily checked that the recolouring algorithm produces a valid conflict-free colouring, because it is essentially an instance of the general framework: After every insertion (and a possible recolouring), the point of highest entry in each interval is uniquely coloured.

Also, it can be proven that the number of recolorings is at most $n - (\lfloor \log_2 n \rfloor + 1)$, and this is tight.

Proposition 5.1. *The number of recolorings in the above algorithm equals*

$$n - (\lfloor \log_2 n \rfloor + 1)$$

in the worst case.

Proof. An input with n vertices uses at least $\lfloor \log_2 n \rfloor + 1$ colours (see, for example, optimal offline colouring of points with respect to intervals in [3]). Whenever a new colour is introduced

	·	·	v_1	·	·	·	·
1			a				
2							
3							
	·	·	1	·	·	·	·

	·	·	v_1	·	·	·	v_2
1			a				d
2							**a**
3							
	·	·	1	·	·	·	**2**

	·	·	v_1	·	v_3	·	v_2
1			a		**d**		**a**
2					a		
3							
	·	·	1	·	**2**	·	**1**

	·	·	v_1	v_4	v_3	·	v_2
1			a	d	d		a
2				d	a		
3				**a**			
	·	·	1	**3**	2	·	1

	·	·	v_1	v_4	v_3	v_5	v_2
1			a	d	**a**	**d**	a
2				d		a	
3				a			
	·	·	1	3	**1**	**2**	1

	v_6	·	v_1	v_4	v_3	v_5	v_2
1	d		a	d	a	d	a
2	**a**			d		a	
3				a			
	2	·	1	3	1	2	1

	v_6	v_7	v_1	v_4	v_3	v_5	v_2
1	**a**	**d**	a	d	a	d	a
2		a		d		a	
3				a			
	1	**2**	1	3	1	2	1

Figure 1. An example of a run of the recolouring algorithm

during the run of the algorithm, there is no recolouring. Thus, there are at most $n - (\lfloor \log_2 n \rfloor + 1)$ recolorings, because in every other insertion at most one old point is recoloured.

Now, we are going to show a family of instances for which the above algorithm performs exactly $n - (\lfloor \log_2 n \rfloor + 1)$ recolorings. We use the *relative positions* notation for the input, that was introduced in [2, 3]. We explain this notation briefly: Each input of n requests of points is denoted by a sequence σ of n natural numbers, so that the t-th element of the sequence, i.e., σ_t, is a natural number in $[0, t-1]$, and the point requested at time t has exactly σ_t already requested points to the *left* of it.

We define, for $k \geq 1$, an instance σ^k of length $n = 2^k - 1$ for which our recolouring algorithm uses k colours and does $2^k - k - 1$ recolorings. The instance $\sigma^1 = 0$. For $k \geq 1$, the instance σ^{k+1} is defined recursively:

$$\sigma^{k+1} = \sigma^k \circ \underbrace{(2^k - 1, \dots, 2^k - 1)}_{2^k \text{ times}},$$

where '$\circ$' is the concatenation operation for finite sequences. Since, for every k, σ^k is a prefix of σ^{k+1}, we have in fact provided an unbounded length relative positions input

$$\sigma = \underbrace{2^0 - 1}_{2^0}, \underbrace{2^1 - 1, 2^1 - 1}_{2^1}, \underbrace{2^2 - 1, \dots, 2^2 - 1}_{2^2}, \dots, \underbrace{2^k - 1, \dots, 2^k - 1}_{2^k}, \dots$$

or

$$\sigma = 0, 1, 1, 3, 3, 3, 3, 7, 7, 7, 7, 7, 7, 7, 7, \dots$$

The following can be proven by induction and we omit the easy but tedious details. For each σ^k, the recolouring algorithm produces the colouring C^k, defined recursively as $C^1 = 1$ and $C^k = C^{k-1} \circ (k) \circ C^{k-1}$, for $k > 1$. Therefore, for $t < 2^k$, input σ is using at most k colours. The point inserted at $t = 2^k$, which is the first point of σ^{k+1} (or σ) that is inserted at relative position $2^k - 1$, is coloured with a new colour $k + 1$, and therefore no recolouring happens. For all subsequent $2^k - 1$ points inserted at relative position $2^k - 1$, there is a recolouring by the algorithm.

Therefore, for all points, except the ones inserted at $t = 1, 2, 4, \ldots, 2^k, \ldots$ a recolouring happens, and therefore after n insertions, $n - (\lfloor \log_2 n \rfloor + 1)$ recolorings happen in σ. □

For example, the run of the recolouring algorithm on input σ^3 is shown in figure 2, where recolorings are shown with bold. The first row of the table represents the order in which points appear and the last row of the table shows the current colouring. Recolorings happen at t = 3, 5, 6, 7.

	v_1	·	·	·	·	·	·
1	a						
2							
3							
	1	·	·	·	·	·	·

	v_1	·	v_2	·	·	·	·
1	a		d				
2			a				
3							
	1	·	2	·	·	·	·

	v_1	v_3	v_2	·	·	·	·
1	a	d	**a**				
2		a					
3							
	1	2	**1**	·	·	·	·

	v_1	v_3	v_2	·	·	·	v_4
1	a	d	a				d
2		a					d
3							a
	1	2	1	·	·	·	3

	v_1	v_3	v_2	·	·	v_5	v_4
1	a	d	a			d	**a**
2		a				d	
3						a	
	1	2	1	·	·	3	**1**

	v_1	v_3	v_2	·	v_6	v_5	v_4
1	a	d	a		d	d	a
2		a			d	**a**	
3					a		
	1	2	1	·	3	**2**	1

	v_1	v_3	v_2	v_7	v_6	v_5	v_4
1	a	d	a	d	**a**	d	a
2		a		d		a	
3				a			
	1	2	1	3	**1**	2	1

Figure 2. The run of the recolouring algorithm on input σ^3

5.2. An $O(\log n)$ colours algorithm for circular arcs

We define a hypergraph H closely related to the one induced by intervals: The vertex set of H is represented as a finite set P of n distinct points on a *circle* C and its hyperedge set consists of all intersections of the points with some *circular arc* of C. In the static case, it is not difficult to show that n points can be optimally conflict-free coloured with respect to circular arcs with $\lfloor \log_2(n-1) \rfloor + 2$ colours: There must be a point p with unique colour in P, and therefore all circular arcs that include p have the conflict-free property; the remaining $n-1$ points of $P \setminus \{p\}$ and the remaining circular arcs induce the same hypergraph as the set of intervals on $n-1$ points, which is optimally coloured with $\lfloor \log_2(n-1) \rfloor + 1$ more colours.

Here, we are interested in an online setting, where the set $P \subset C$ is revealed incrementally by an adversary, and, as usual, the algorithm has to commit to a colour for each point without knowing how future points will be requested. Algorithms for intervals can be used almost verbatim for circular arcs. In fact, the recolouring algorithm for intervals, given in section 5.1, can be used verbatim, if the notion of adjacency of points is adapted to the closed curve setting (for $n \geq 3$, each point has exactly two immediate neighbouring points, whereas in the intervals case, the two extreme points have only one neighbour). Again, in each entry ℓ, at least a third of the points is assigned auxiliary colour a, and thus at most $\log_{3/2} n + 1$ colours are used.

5.3. An $O(\log n)$ colours algorithm for circular arcs with substitution of points

We consider a variation on the problem of online conflict-free colouring with respect to circular arcs that was given in section 5.2. In this new variation, the adversary has, in addition to the insertion move of a new point, a *substitution move*:

Definition (substitution move). The adversary can substitute a set Q of already requested points, that are consecutive on the circle, with a single new point p, and the algorithm has to colour p, such that the whole set of points is conflict-free coloured with respect to circular arcs (in that new set, p is included, but all points in Q are removed).

Our algorithm for this variation of the problem relies on the one given in section 5.2. For an insertion move of the adversary, it colours the new point like in section 5.2. For a substitution move of the adversary, it colours the new point p, with the *highest* colour occurring in the points of Q. Point p also gets the entries of the unique point $q \in Q$ with the highest colour. It is not difficult to see that the colouring remains conflict-free after each move. We remark that a recolouring can happen only in an insertion move and that substitution moves do not make the algorithm introduce new colours. The following is true for every t:

Lemma 5.2. *After t moves, the above colouring algorithm uses at most $\log_{3/2} t + 1$ colours.*

Proof. During a substitution move we might break the invariant that between any pair of consecutive a's there are at most two d-points. However if in each entry we denote by $\bar{a}$ a point formerly coloured with a which was substituted, then it can be proven that between any two consecutive points coloured with a or $\bar{a}$, there are at most two d-points and thus it implies that at least one third of the points in every level are coloured either by that level or have been substituted. We call these points coloured with $\bar{a}$ *ghost* points. Moreover, we assign ghost points to substitution points as follows. If a point p substitutes a point p' coloured with a, p' becomes a ghost point and p is assigned the ghost point p'. If a point p substitutes a point q which has some ghost points, p is assigned all ghost points of q. We ignore the trivial substitution of one point coloured with a and do not create a ghost point and any assignment in this case. It is not difficult to see that at any time instance t each ghost point is assigned to exactly one non-ghost point.

We intend to make the above argument formal as follows. We will prove the stronger result that the number of colours used by the algorithm is at most $\log_{3/2} i + 1$, where i is the number of insertion moves until time t. In order to prove the previous statement it is enough to show that at each entry ℓ, the number of points getting auxiliary colour d in entry ℓ is bounded by the number of insertion moves that reached entry ℓ as follows.

$$d_\ell \leq \lfloor \tfrac{2}{3} i_\ell \rfloor \tag{5.1}$$

where d_ℓ is the number of points getting auxiliary colour d in entry ℓ and i_ℓ is the number of insertion moves that reached entry ℓ. The above inequality is true when no points have reached entry ℓ. Moreover, it remains true as long as a substitution move happens, or an insertion move happens in which the point at entry ℓ is coloured with a. The number of d's increases only if there is an insertion move where the point at level ℓ is coloured with d. We will study further this last case. For a new point p to get auxiliary colour d it must be the case that it is inserted next to a point coloured with a and a point q coloured with d such that q is adjacent to a point coloured with a. In a fixed entry ℓ, we call a maximal set of consecutive points coloured with d a *strip*. The length of a strip s is the number of d's in it and is denoted by $\operatorname{len}(s)$. It is not difficult to see that if there is at least one a in entry ℓ, as in our case, the number of strips is the same as the number of a's.

The number of insertion moves that reach entry ℓ satisfies the following equation.

$$i_\ell \geq a_\ell + d_\ell + \bar{a}_\ell \tag{5.2}$$

where a_ℓ is the number of points coloured with a, and $\bar{a}_\ell$ the number of ghost points (points substituted that were coloured with a). We have an inequality, because we omit the points substituted that were coloured with d. If a strip s has length $\operatorname{len}(s) > 2$, it necessarily contains ghost points. In fact if a strip s has length $\operatorname{len}(s)$, one can prove that points in it have been assigned at least $\lceil \frac{1}{2}\operatorname{len}(s)\rceil - 1$ ghost points. We defer the proof of the above fact to lemma 5.3. Because of all the above, inequality (5.2) implies the following.

$$i_\ell \geq a_\ell + \sum_{s:\text{ strip}} \operatorname{len}(s) + \sum_{s:\text{ strip}} \left(\lceil \tfrac{1}{2}\operatorname{len}(s)\rceil - 1\right) = \sum_{s:\text{ strip}} \lceil \tfrac{3}{2}\operatorname{len}(s)\rceil$$

The above inequality implies

$$\lfloor \tfrac{2}{3} i_\ell \rfloor \geq \Big\lfloor \tfrac{2}{3} \sum_{s:\text{ strip}} \lceil \tfrac{3}{2}\operatorname{len}(s)\rceil \Big\rfloor \geq \Big\lfloor \sum_{s:\text{ strip}} \operatorname{len}(s) \Big\rfloor = \sum_{s:\text{ strip}} \operatorname{len}(s) = d_\ell$$

which is inequality (5.1). □

Lemma 5.3. *The points in a strip s have been assigned at least $\lceil \frac{1}{2}\operatorname{len}(s)\rceil - 1$ ghost points.*

Proof. We prove the above fact by induction on t. For $t = 0$ it is trivially true. For length of a strip at most two, again it is trivially true because $\lceil \frac{1}{2}\operatorname{len}(s)\rceil - 1 = 0$. We ignore trivial substitutions of one point coloured with a because they do not change the lengths of the strips and do not create ghost points. Assume there is a strip of length greater than two. Necessarily, the last action in the strip was a substitution move, because in an insertion the algorithm never colours with d, if there are already two d points in the strip. There are two possible cases for a substitution move.

In the first case, there is a substitution of only d-points as shown below, i.e., the substitution is completely contained in one strip, say of length L', and the new strip created has length $L \leq L'$.

$$\overbrace{dddd\,\underbrace{dddd\ldots ddddd}_{\text{substitution}}\,dd}^{L'}$$

In this case, the number of ghost points in the new strip is the same as the number of ghost points in the old strip, which is, by the inductive hypothesis, at least $\lceil \frac{1}{2}L'\rceil - 1$, which is at least $\lceil \frac{1}{2}L\rceil - 1$.

In the second case, the substitution spans more than one strip, i.e., also some (non-ghost) points coloured with a. Say that the substitution spans k a's which are surrounded by $k+1$ strips of lengths $L_1, \ldots, L_{k+1}$, as shown below.

$$\overbrace{dddddd}^{L_1}\,\underbrace{a\,\overbrace{ddddd}^{L_2}\,a\ldots a\,\overbrace{ddddd}^{L_k}\,a}_{\text{substitution}}\,\overbrace{dddd}^{L_{k+1}}$$

The length of the new strip is $L \leq L_1 + L_{k+1} + 1$ if $k \geq 2$, and $L \leq L_1 + L_2$ if $k = 1$ (this last inequality is true because there can be no trivial substitution). In this case, the number of ghost points in the new strip is the same as the number of ghost points in the $k + 1$ strips plus k, which is at least

$$\sum_{i=1}^{k+1}\left(\lceil \tfrac{1}{2}L_i \rceil - 1\right) + k \geq \left\lceil \tfrac{1}{2}\sum_{i=1}^{k+1} L_i \right\rceil - 1 \geq \lceil \tfrac{1}{2}L \rceil - 1.$$

In the above, we used the inductive hypothesis for each of the $k + 1$ strips. □

5.4. An $O(\log n)$ colours algorithm for halfplanes

In this section we describe a deterministic algorithm for online conflict-free colouring points with respect to halfplanes that uses $O(\log n)$ colours and performs $O(n)$ recolorings. Thus, it can also be modified for conflict-free colouring points in the plane with respect to unit disks as described in section 6 (see proof of corollary 6.3). At every time instance t, the algorithm maintains the following invariant (V_t is the set of points that have appeared so far).

Invariant. All points (strictly) inside the convex hull of V_t are coloured with the same special colour, say '$\star$'. The set of points on the convex hull of V_t, denoted by $\mathrm{CH}(V_t)$, are coloured with another set of colours, such that every set of consecutive points on the convex hull has a point with a unique colour.

Every non-empty subset of points of V_t induced by a halfplane contains a set of consecutive points on the convex hull of V_t, and thus the whole colouring is conflict-free. If one considers the points of $\mathrm{CH}(V_t)$ in their circular order on the convex hull, it is enough to conflict-free colour them with respect to circular arcs. The number of colours used in $\mathrm{CH}(V_t)$ must be logarithmic in t.

We describe how the algorithm maintains the above invariant. A new point v_{t+1} that appears at time $t + 1$ is coloured as follows: If it is inside the convex hull of V_t, then it gets colour '$\star$'. Otherwise, the new point v_{t+1} will be on $\mathrm{CH}(V_{t+1})$, in which case we essentially use the algorithm of section 5.3 to colour it. We have two cases, which correspond to a substitution and an insertion move, respectively:

- It might be the case that v_{t+1} forces some points (say they comprise set Q) that were in $\mathrm{CH}(V_t)$ to appear in the interior of $\mathrm{CH}(V_{t+1})$, so in order to maintain the invariant, all points in Q are recoloured to '$\star$', and v_{t+1} is coloured with the maximum colour occurring in Q (this is like a substitution move of section 5.3).
- If, on the other hand, no points of $\mathrm{CH}(V_t)$ are forced into the convex hull, then point $v_{t+1} \in \mathrm{CH}(V_{t+1})$ is coloured like in an insertion move of section 5.3, with the algorithm for circular arcs. In that last case, in order to maintain logarithmic number of colours on t, one recolouring of a point in $\mathrm{CH}(V_{t+1})$ might be needed.

The total number of recolorings is guaranteed to be $O(n)$, because for every insertion, at most one recolouring happens on the new convex hull, and every point coloured with '$\star$' keeps that colour for the rest of the algorithm run.

6. Application to geometry

Our randomised algorithm has applications to conflict-free colourings of certain geometric hypergraphs studied in [7, 8, 9]. We obtain the same asymptotic result as in [7, 8, 9] but with a better constant of proportionality and using much fewer random bits. For example, if the hypergraph H is induced by intervals, it can be proven (with an analysis similar to the one given in section 4) that for any order of insertion of n points, when the auxiliary colour for each entry is chosen uniformly at random from $\{a, b, c\}$, the expected number of colours used is bounded by $\log_{3/2} n + 1$. It is interesting that the best known upper bound for dynamically colouring n points deterministically, when the whole insertion order is known in advance, is also $\log_{3/2} n + 1$ (see, for example, [3] for further details). In our algorithm the expected number of colours is bounded by $1 + \log_{3/2} n \approx 1.71 \log_2 n$, whereas in [7, 8] by $1 + \log_{8/7} n \approx 5.19 \log_2 n$, three times our bound. In the following, we provide a run example for the algorithm on intervals.

Example. Consider the case where the hypergraph is induced by points with respect to intervals. Namely, $V = \{1, \ldots, n\}$ and E consists of all possible discrete intervals of V (i.e., subsets of consecutive integers). Vertices appear one by one and at each time t we must have an online conflict-free colouring with respect to the discrete interval subsets of the t points revealed by time t. It is not difficult to see that the hypergraphs $H(V_t^i)$ can always be properly non-monochromatically online 3-coloured (say with auxiliary colours a, b, c) as follows: Each newly inserted point has at most two immediate neighbours and thus even a first-fit colouring suffices. In figure 3, we exhibit a run of the algorithm for the permutation $\pi = 253164$, seen as a mapping from time $t \in \{1, \ldots, 6\}$ to the corresponding vertex at position $\pi_{(t)}$.

i	$f(i)$	$\cdot$	v_1	$\cdot$	$\cdot$	$\cdot$	$\cdot$
1	**b**		a				
2	**a**		**a**				
3							
4							
5							
6							
		$\cdot$	**2**	$\cdot$	$\cdot$	$\cdot$	$\cdot$

i	$f(i)$	$\cdot$	v_1	$\cdot$	$\cdot$	v_2	$\cdot$
1	b		a			**b**	
2	a		a				
3							
4							
5							
6							
		$\cdot$	2	$\cdot$	$\cdot$	**1**	$\cdot$

i	$f(i)$	$\cdot$	v_1	v_3	$\cdot$	v_2	$\cdot$
1	b		a	c		b	
2	a		a	b			
3	**c**			a			
4	**a**			**a**			
5							
6							
		$\cdot$	2	**4**	$\cdot$	1	$\cdot$

i	$f(i)$	v_4	v_1	v_3	$\cdot$	v_2	$\cdot$
1	b	**b**	a	c		b	
2	a		a	b			
3	c			a			
4	a			a			
5							
6							
		1	2	4	$\cdot$	1	$\cdot$

i	$f(i)$	v_4	v_1	v_3	$\cdot$	v_2	v_5
1	b	b	a	c		b	a
2	a		a	b			**a**
3	c			a			
4	a			a			
5							
6							
		1	2	4	$\cdot$	1	**2**

i	$f(i)$	v_4	v_1	v_3	v_6	v_2	v_5
1	b	b	a	c	a	b	a
2	a		a	b	c		a
3	c			a	b		
4	a			a	b		
5	**b**				a		
6	**a**				**a**		
		1	2	4	**6**	1	2

Figure 3. A run example of the framework for hypergraphs induced by points with respect to intervals

In the end the vertices look like $\pi^{-1} = v_4 v_1 v_3 v_6 v_2 v_5$, where v_t is the vertex appearing at time t. The choices are $f(1) = b$, $f(2) = a$, $f(3) = c$, $f(4) = a$, $f(5) = b$, $f(6) = a$. The six tables correspond to $t = 1, \ldots, 6$ and at the bottom of each table the online conflict-free colouring, so far, is shown. Entries correspond to rows in the tables, where for each entry i the following data is given: the representing colour $f(i)$ and the proper non-monochromatic auxiliary colouring of the vertices in the hypergraph V_t^i with three colours a, b or c.

Observe that entries 3 and 5, respectively, do not have a vertex coloured with $f(3)$ and $f(5)$, respectively. As a consequence colours 3, 5 do not appear in the conflict-free colouring although colours 1, 2, 4, 6 do. If it is important to use consecutive colours, namely k different colours implies they are $\{1, \ldots, k\}$, the above problem can be fixed by assigning the next unused conflict-free colour to an entry i only as soon as a vertex in entry i is coloured with auxiliary colour $f(i)$. The above remedy works in our general framework, not only in the specific case of this example.

Halfplanes and unit disks. When H is the hypergraph obtained by points in the plane intersected by halfplanes or unit disks, we obtain online randomised algorithms that use $O(\log n)$ colours with high probability. Before proceeding it is necessary to prove a degeneracy result about hypergraphs induced by halfplanes.

Lemma 6.1. *Let V be a finite set of n points in the plane and let E be all subsets of V that can be obtained by intersecting V with a halfplane. Then the hypergraph $H = (V, E)$ is 3-degenerate.*

Proof. We assume that points are in general position, i.e., no three of them are on the same line. We also assume that points are inserted in some order $v_1, v_2, \ldots, v_n$. Following the notation in the definition of a degenerate hypergraph on page 4, it is enough to prove that for every t, we have

$$S_t \leq 3t \tag{6.1}$$

(we remark that we have dropped the permutation π, appearing in inequality (2.1) on page 5, because it is implied by the order $v_1, v_2, \ldots, v_n$). We prove something stronger than inequality (6.1), namely that

$$S_t + C_t \leq 3t, \tag{6.2}$$

by induction, where C_t is the number of points on the boundary of the convex hull at time t, which is always a positive number. It will be helpful to define the following differences:

$$\Delta S_t = S_t - S_{t-1},$$
$$\Delta C_t = C_t - C_{t-1}.$$

The difference ΔS_t is exactly the number of neighbours of v_t in the Delaunay graph at time t, that is, $G(H(\{v_1, \ldots, v_t\}))$. For v_t to be a neighbour of $v_{t'}$, with $t' < t$, in this Delaunay graph, there must exist a halfplane at time t which contains exactly v_t and $v_{t'}$.

First, we show that inequality (6.2) is true for small values of t. For $t \in \{1, 2, 3\}$, inequality (6.2) is true as exhibited in table 1, because the size of the convex hull is the same as the number of points and every two points are neighbours in the Delaunay graph.

Table 1. Edges in Delaunay graph for halfplanes and size of convex hull for small values of t

t	1	2	3
S_t	0	1	3
C_t	1	2	3
$S_t + C_t$	1	3	6

Then, for the inductive step, for $t > 3$, it is enough to prove that

$$\Delta S_t + \Delta C_t \leq 3, \tag{6.3}$$

because then the sum $S_t + C_t$ increases at most by 3 at every time step and therefore always remains bounded by $3t$. Denote the convex hull of points $\{v_1, \ldots, v_t\}$ with CH_t. Consider the following two cases. Either v_t lies outside of the convex hull CH_{t-1} or v_t is inside the convex hull CH_{t-1}.

Assume v_t lies outside of the convex hull CH_{t-1} (see figure 4). Then v_t lies on the boundary of

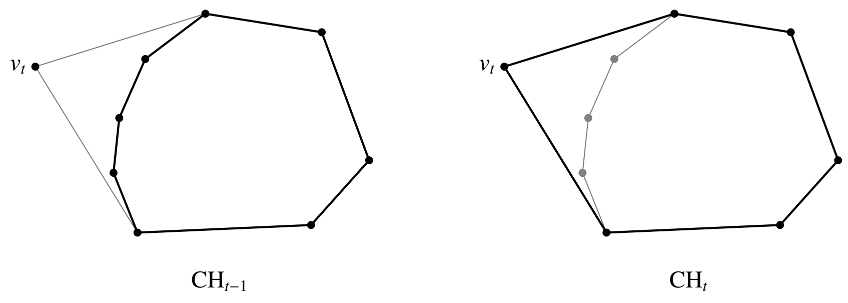

Figure 4. The new point is outside the old convex hull

the convex hull CH_t. Consider the two points u and w that are the neighbours of v_t in the cyclic order of points on the convex hull CH_t (see figure 5). Consider the line ℓ passing through u and

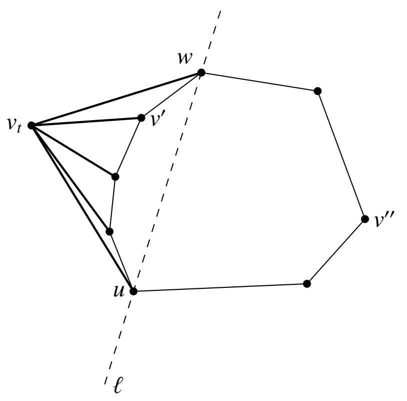

Figure 5. Delaunay graph neighbours of a new point outside the old convex hull

w. This line partitions points of CH_{t-1} in two types: (a) points on ℓ or in the same halfplane as

v_t defined by ℓ (points like u, v', w in figure 5) and (b) points on the other halfplane defined by ℓ (points like v'' in figure 5). Vertex v_t is adjacent to every vertex v' of type (a) (including u, w) in the Delaunay graph, because one can take a halfplane with defining line passing through v' and slope between the slopes of the incident sides to v' of the convex hull CH_{t-1}, and this halfplane contains only v_t and v'. On the other hand, no vertex v'' of type (b) is a neighbour in the Delaunay graph with v_t, because every halfplane that contains v_t and v'' must contain at least one of u, w. Assume there are d vertices of CH_{t-1} of type (a), with $d \geq 2$. Then, $\Delta S_t = d$ and $d-2$ of them no longer appear on the convex hull, but v_t appears on CH_t, i.e., $\Delta C_t = -(d-2)+1$. Therefore, we have proved that when v_t lies outside CH_{t-1}, $\Delta S_t + \Delta C_t = d + (-(d-2)+1) = 3$, i.e., inequality (6.3) is true.

Assume v_t is inside the convex hull CH_{t-1} (see figure 6). Then, consider any triangulation of CH_{t-1}. Point v_t is in exactly one triangle of the triangulation, call it xyz, where x, y, z are points

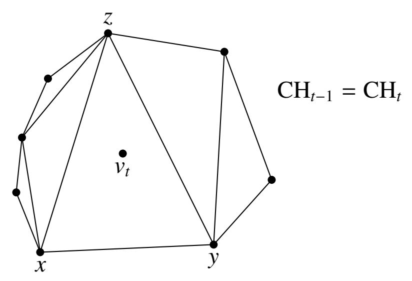

Figure 6. A triangulation of the convex hull in case the new point is inside the old convex hull

on the convex hull, corresponding to points inserted before v_t. It is not difficult to prove that every halfplane that contains v_t, contains at least one of x, y, z. Therefore v_t can have at most three neighbours in the Delaunay graph. The three neighbours case can be realised when the only points on the convex hull are x, y, z, i.e., when $t = 4$, by taking for every point $p \in \{x, y, z\}$ a halfplane that contains p, and the defining line of the halfplane (a) is passing through v_t and (b) is parallel to the line passing through the other two points in $\{x, y, z\}$. If there are more than three points in CH_{t-1}, we will prove that it is not possible for v_t to have all three neighbours x, y, z in the Delaunay graph. Assume for the sake of contradiction that there is a halfplane h_x containing only v_t and x, a halfplane h_y containing only v_t and y, and a halfplane h_z containing only v_t and z. For every point $p \in \{x, y, z\}$ define the halfline starting at v_t with direction $\overrightarrow{pv_t}$, i.e., not containing p. These halflines are shown in figure 7. These three halflines partition the plane into three areas, A_x, A_y, A_z, each one containing one of the points x, y, z, respectively. We now consider halfplanes containing at least v_t. It is not difficult to see that such a halfplane containing only x and not y or z must contain all of A_x. Similarly, such a halfplane containing only y and not x or z must contain all of A_y, and such a halfplane containing only z and not x or y must contain all of A_z. Therefore, any other point contained in CH_{t-1} except x, y, z must be contained in one of h_x, h_y, and h_z, which is a contradiction. Thus, we have proved that when v_t is in CH_{t-1}, $\Delta S_t \leq 3$ and $\Delta C_t = 0$, i.e., inequality (6.3) is true. □

Corollary 6.2. *Let H be a hypergraph induced by points and halfplanes, as in lemma 6.1. Then, the expected number of colours used by our randomised online conflict-free colouring algorithm*

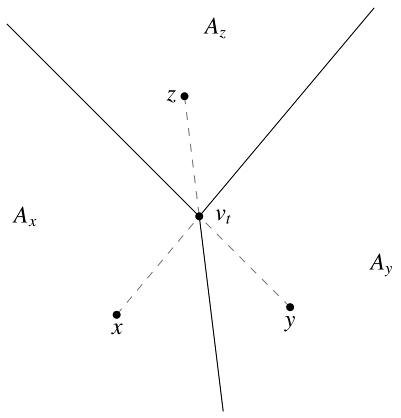

Figure 7. A partition of the plane

applied to H is at most $\log_{14/13} n + 1$*, in the oblivious adversary model. Also the actual number of colours used is* $O(\log_{14/13} n)$ *with high probability. The number of random bits is* $O(\log n)$ *with high probability.*

Proof. The proof follows from theorem 4.1 and lemmata 4.5, 6.1. □

Corollary 6.3. *Let V be a set of n points in the plane and let E be the set of all subsets of V that can be obtained by intersecting V with a unit disk. Then, there exists a randomised online algorithm for conflict-free colouring H which uses* $O(\log n)$ *colours and* $O(\log n)$ *random bits with high probability, against an oblivious adversary.*

Proof. In [9], it was observed that by an appropriate partitioning of the plane one can modify any online algorithm for conflict-free colouring points with respect to halfplanes to an online algorithm for conflict-free colouring points with respect to congruent disks. The congruent disks algorithm uses a constant multiple of the colours used by the halfplanes algorithm. Using the same technique as developed in [9] and corollary 6.2 we obtain the desired result. □

7. Discussion and open problems

We presented a framework for online conflict-free colouring any hypergraph. This framework coincides with some known algorithms in the literature when restricted to some special underlying hypergraphs. We derived a randomised online algorithm for conflict-free colouring any hypergraph (in the oblivious adversary model) and showed that the performance of our algorithm depends on a parameter which we refer to as the degeneracy of the hypergraph which is a generalisation of the known notion of degeneracy in graphs (i.e., when the hypergraph is a simple graph then our notion is similar to the classical definition of degeneracy of a graph). Specifically, if the hypergraph is k-degenerate then our algorithm uses $O(k \log n)$ colours with high probability, which is asymptotically optimal for any constant k, and $O(k \log k \log n)$ random bits. This is the first efficient online conflict-free colouring for general hypergraphs and subsumes all the previous randomised algorithmic results of [8, 9]. It also substantially improves the efficiency with respect to the amount of randomness used in the special cases studied in [8, 9]. Another interesting fact to note is that our algorithm when applied to k-inductive graphs gives an online colouring of such graphs with $O(k \log n)$ colours with high probability. In [15], it was shown that the same bound can be achieved deterministically by the first-fit greedy algorithm.

A major open problem is to find an efficient online *deterministic* algorithm for conflict-free colouring k-degenerate hypergraphs. Even for the very special case of a hypergraph induced by points and intervals (as in the example in section 6), where the number of neighbours of the Delaunay graph of every induced hypergraph is at most two, the best known deterministic online conflict-free colouring algorithm from [8] uses $\Theta(\log^2 n)$ colours. We hope that our technique together with a possible clever derandomisation technique can shed light on this problem.

As mentioned already, the framework of section 3 can describe some known algorithms such as the unique maximum greedy algorithm of [8] for online conflict-free colouring points on a line with respect to intervals. No sharp asymptotic bounds are know for the performance of unique maximum greedy, in terms of the number of colours it uses. The best known upper bound is linear (asymptotically $n/2$ from [2, 3]), whereas the best known lower bound is $\Omega(\sqrt{n})$. We believe that this new description of unique maximum greedy with the help of the framework could help analyse its performance.

In section 5 we initiate the study of online conflict-free colouring with recolouring: We provide a deterministic online conflict-free colouring for points on the real line with respect to intervals and show that our algorithm uses $\Theta(\log n)$ colours and at most one recolouring per insertion. This is in contrast with the best known deterministic online conflict-free colouring with respect to intervals that uses $\Theta(\log^2 n)$ colours in the worst case without recolouring, by [8]. We also present deterministic online algorithms for conflict-free colouring points with respect to circular arcs and halfplanes (and unit disks) that use $O(\log n)$ colours and $O(n)$ recolorings in the worst case. In the special case of intervals or circular arcs at most one point is recoloured per insertion.

It would be interesting to find a deterministic online conflict-free colouring algorithm for points in the plane with respect to halfplanes that uses $\Theta(\log n)$ colours in the worst case and recolours at most a constant number of points per insertion. We leave this as an open problem for further research.

All of our randomised algorithms assume the oblivious adversary model, in which the adversary has to commit to a specific input sequence before revealing the first vertex to the algorithm without knowing the random bits that the algorithm is going to use. The randomised model can be seen as a relaxation of the strict deterministic model: some power is taken from the adversary, or equivalently given to the algorithm, in order to achieve just a logarithmic number of colours. Another such relaxation is to give extra information to the algorithm about where each requested point will end up in the final colouring (the *absolute positions* model, which was introduced in [2, 3]). Other such relaxations are given in [2, 3] (colouring with respect to rays) and [19] (online ranking of paths). In this work we introduced yet another relaxation, the recolouring model, in which the algorithm is allowed to recolour some of the points. An interesting question is to construct $O(\log n)$ colours algorithms that rely as little as possible on their extra power (as few random bits as possible, as few recolorings as possible). Towards that goal, in a unified framework, we provided the best known results: randomised algorithms that use an expected logarithmic number of random bits, and recolouring algorithms that perform at most a linear number of recolorings.

Acknowledgements

We would like to thank Boris Aronov, Amos Fiat, Haim Kaplan, János Pach, David Peleg and Micha Sharir for helpful discussions concerning the problems studied in this paper.

References

[1] Ajwani, D., Elbassioni, K., Govindarajan, S. and Ray, S. (2007) Conflict-free coloring for rectangle ranges using $O(n^{.382})$ colors. In *Proceedings of the 19th Annual ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 181–187.

[2] Bar-Noy, A., Cheilaris, P. and Smorodinsky, S. (2006) Conflict-free coloring for intervals: from offline to online. In *Proceedings of the 18th Annual ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 128–137.

[3] Bar-Noy, A., Cheilaris, P. and Smorodinsky, S. (2008) Deterministic conflict-free coloring for intervals: from offline to online. *ACM Transactions on Algorithms* **4**(4) 44:1–44:18.

[4] Borodin, A. and El-Yaniv, R. (1998) *Online computation and competitive analysis*. Cambridge University Press.

[5] Chan, J., Chin, F., Hong, X. and Ting, H. (2008) Dynamic offline conflict-free coloring for unit disks. In *Proceedings of the 6th Workshop on Approximation and Online Algorithms (WAOA)*, pages 241–252.

[6] Cheilaris, P. (2008) *Conflict-free coloring*. Ph.D. thesis, City University of New York.

[7] Chen, K. (2006) How to play a coloring game against a color-blind adversary. In *Proceedings of the 22nd Annual ACM Symposium on Computational Geometry (SoCG)*, pages 44–51.

[8] Chen, K., Fiat, A., Kaplan, H., Levy, M., Matoušek, J., Mossel, E., Pach, J., Sharir, M., Smorodinsky, S., Wagner, U. and Welzl, E. (2007) Online conflict-free coloring for intervals. *SIAM Journal on Computing* **36**(5) 1342–1359.

[9] Chen, K., Kaplan, H. and Sharir, M. (2009) Online conflict free coloring for halfplanes, congruent disks, and axis-parallel rectangles. *ACM Transactions on Algorithms* **5**(2) 16:1–16:24.

[10] Elbassioni, K. and Mustafa, N. H. (2006) Conflict-free colorings of rectangles ranges. In *Proceedings of the 23rd International Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 254–263.

[11] Even, G., Lotker, Z., Ron, D. and Smorodinsky, S. (2003) Conflict-free colorings of simple geometric regions with applications to frequency assignment in cellular networks. *SIAM Journal on Computing* **33** 94–136.

[12] Fiat, A., Levy, M., Matoušek, J., Mossel, E., Pach, J., Sharir, M., Smorodinsky, S., Wagner, U. and Welzl, E. (2005) Online conflict-free coloring for intervals. In *Proceedings of the 16th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 545–554.

[13] Gyárfás, A. and Lehel, J. (1988) On-line and first fit coloring of graphs. *Journal of Graph Theory* **12**(2) 217–227.

[14] Har-Peled, S. and Smorodinsky, S. (2005) Conflict-free coloring of points and simple regions in the plane. *Discrete and Computational Geometry* **34** 47–70.

[15] Irani, S. (1994) Coloring inductive graphs on-line. *Algorithmica* **11**(1) 53–72.

[16] Katz, M. J., Lev-Tov, N. and Morgenstern, G. (2007) Conflict-free coloring of points on a line with respect to a set of intervals. In *Proceedings of the 19th Canadian Conference on Computational Geometry (CCCG)*, pages 93–96.

[17] Kierstead, H. A. (1988) The linearity of first-fit coloring of interval graphs. *SIAM Journal on Discrete Mathematics* **1**(4) 526–530.

[18] Pach, J. and Tóth, G. (2003) Conflict free colorings. In *Discrete and Computational Geometry, The Goodman-Pollack Festschrift*, pages 665–671. Springer Verlag.

[19] Schiermeyer, I., Tuza, Z. and Voigt, M. (2000) On-line rankings of graphs. *Discrete Mathematics* **212**(1–2) 141–147.

[20] Smorodinsky, S. (2003) *Combinatorial Problems in Computational Geometry*. Ph.D. thesis, School of Computer Science, Tel-Aviv University.

[21] Smorodinsky, S. (2007) On the chromatic number of geometric hypergraphs. *SIAM Journal on Discrete Mathematics* **21**(3) 676–687.

[22] Smorodinsky, S. (2008) A note on the online first-fit algorithm for coloring k-inductive graphs. *Information Processing Letters* **109** 44–45.