ONLINE CONFLICT-FREE COLORINGS FOR HYPERGRAPHS*

AMOTZ BAR-NOY†, PANAGIOTIS CHEILARIS‡, SVETLANA OLONETSKY§, AND SHAKHAR SMORODINSKY¶

Abstract. We provide a framework for online conflict-free coloring any hypergraph. We introduce the notion of degenerate hypergraph, which characterizes hypergraphs that arise in geometry. We use our framework to obtain an efficient randomized online algorithm for conflict-free coloring any k-degenerate hypergraph with n vertices. Our algorithm uses $O(k \log n)$ colors with high probability and this bound is asymptotically optimal, because there are families of k-degenerate hypergraphs that need that many colors. Moreover, our algorithm uses $O(k \log k \log n)$ random bits with high probability. As a corollary, we obtain asymptotically optimal randomized algorithms for online conflict-free coloring some hypergraphs that arise in geometry. Our algorithm uses exponentially fewer random bits than previous algorithms.

We introduce algorithms that are allowed to perform a few recolorings of already colored points. We provide deterministic online conflict-free coloring 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 $O(\log n)$ colors and perform number of recolorings at most linear in n.

Key words. conflict-free coloring, online, randomized, algorithm, hypergraph

AMS subject classifications. 05C15, 52C99, 68Q25, 68W20, 68W40

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-coloring 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-coloring. Then $\chi(H)$ is called the *chromatic number* of H. A *conflict-free* coloring of H is a coloring of V with the further restriction that for any hyperedge $S \in E$ there exists a vertex $v \in S$ with a unique color (i.e., no other vertex of S has the same color as v). Both proper coloring and conflict-free coloring of hypergraphs are generalizations of vertex coloring of graphs (the definition coincides when the underlying hypergraph is a simple graph). Therefore, computing such hypergraph colorings is at least as hard as computing vertex colorings for simple graphs.

The study of conflict-free colorings originated in the work of Even et al. [8] and Smorodinsky [17] who were motivated by the problem of frequency assignment in cellular networks. Specifically, 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

*A preliminary version of this work appeared in the 34th International Colloquium on Automata, Languages and Programming (ICALP 2008).

†Brooklyn College and Graduate Center of the City University of New York, USA.

‡Rényi Institute of Mathematics, Budapest, Hungary. Part of this work was done while the author was in the Graduate Center of the City University of New York, USA.

§Tel-Aviv University, Israel.

¶Ben-Gurion University, Israel.

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 minimize the number of assigned frequencies ("colors") 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. [8] 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. [8] showed that any hypergraph induced by a family $\mathcal{R}$ of n discs in the plane admits a conflict-free coloring with only $O(\log n)$ colors and that this bound is tight in the worst case. Furthermore, such a coloring can be found in deterministic polynomial time. However, in [8], it was also shown that finding the minimum number of colors needed to conflict-free color 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 [8] were further extended in [11] by combining more involved probabilistic and geometric ideas. The main result of [11] is a general randomized algorithm which conflict-free colors any set of n "simple" regions (not necessarily convex) whose union has "low" complexity, using a "small" number of colors. In addition to the practical motivation, this new coloring model has drawn much attention of researchers through its own theoretical interest and such colorings have been the focus of several recent works (see, e.g., [7, 8, 9, 11, 13, 15, 17, 18, 4]). To capture a dynamic scenario where antennas can be added to the network, Fiat et al. [9] initiated the study of online conflict-free coloring 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 all intersections of the points with some 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 P after the t-th point has been inserted. Each time a point is inserted, the algorithm needs to assign a color $C(p)$ to it, which is a positive integer. Once the color has been assigned to p, it cannot be changed in the future. The coloring should remain conflict-free at all times. That is, for any interval I that contains points of $P(t)$, there is a color that appears exactly once in I. Among other results, [9] provided a randomized algorithm for online conflict-free coloring n points on the line with $O(\log n \log\log n)$ colors 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 coloring n points on the line with $\Theta(\log^2 n)$ colors in the worst case.

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

We present a general framework for online conflict-free coloring any hypergraph. Interestingly, this framework is a generalization of some known coloring algorithms. For example the unique maximum greedy algorithm of [9] 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 *randomized algorithm* and show that the maximum number of colors used is a function of the degeneracy of the hypergraph. We define the notion of a k-degenerate hypergraph as a generalization of the same notion for simple graphs. Specifically we show that if the hypergraph is k-degenerate, then our algorithm uses $O(k \log n)$ colors with high probability, against an *oblivious adversary* (see [3]). 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 going to use.

As demonstrated in [9], the problem of online conflict-free coloring the very special hypergraph induced by points on the real line with respect to intervals is highly non-trivial. The best randomized online conflict-free coloring algorithm of [9] uses $O(\log n \log \log n)$ colors. Kaplan and Sharir [13] studied the special hypergraph induced by points in the plane with respect to halfplanes and unit disks and obtained a randomized online conflict-free coloring with $O(\log^3 n)$ colors with high probability. Recently, the bound $\Theta(\log n)$ just for these two special cases was obtained independently by Chen [5] (see also [6]). Our algorithm is more general and uses only $\Theta(\log n)$ colors; an interesting evidence to our algorithm being fundamentally different from the ones in [5, 9, 13], when used for the special case of hypergraphs that arise in geometry, is that our algorithm uses exponentially fewer random bits. The algorithms of [5, 13] use $\Theta(n)$ random bits and our algorithm uses $O(\log n)$ random bits.

Another interesting corollary of our result is that we obtain a randomized online coloring for k-inductive graphs with $O(k \log n)$ colors with high probability. This case was studied by Irani [12] who showed that the first-fit greedy algorithm achieves the same bound deterministically.

Deterministic online conflict-free coloring with recoloring. We initiate the study of online conflict-free coloring where at each step, in addition to the assignment of a color to the newly inserted point, we allow some recoloring of other points. The bi-criteria goal is to minimize the total number of recolorings done by the algorithm and the total number of colors used by the algorithm. We introduce an online algorithm for conflict-free coloring points on the line with respect to intervals, where we recolor at most one already assigned point at each step. Our algorithm uses $\Theta(\log n)$ colors. This is in contrast with the $O(\log^2 n)$ colors used by the best known deterministic algorithm by [9] that does not recolor points. We also provide an online algorithm for conflict-free coloring points on the plane with respect to halfplanes that uses $\Theta(\log n)$ colors and the total number of recolorings is $O(n)$. For this problem no deterministic algorithm was known before.

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

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

2. Preliminaries. We start with some basic definitions:

DEFINITION 2.1. *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 2.2. *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).*

Here is a graph theoretic common definition:

DEFINITION 2.3. *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 2.4. *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

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

that is, $d(v_{\pi(j)})$ is the number of neighbors 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 coloring. Let $H = (V, E)$ be any hypergraph. Our goal is to define a framework that colors the vertices 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 color to the newly revealed vertex v_t. This color cannot be changed in future times $t' > t$. The coloring 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* colors (not to be confused with the set of *main* colors used for the conflict-free coloring: $\{1, 2, \ldots\}$). Let $f : \mathbb{N}^+ \to A$ be some fixed function. We now 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 coloring of $H(V_t^i)$ with at most h colors. We say that $f(i)$ is the color 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 color can be assigned to v_t such that the auxiliary coloring of V_{t-1}^1 together with the color of v_t is a proper non-monochromatic coloring of $H(V_{t-1}^1 \cup \{v_t\})$. Any (proper non-monochromatic) coloring procedure can be used by the framework. For example a first-fit greedy method in which all colors in the order $a_1, \ldots, a_h$ are checked until one is found. If such a color 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 color can be assigned, then v_t is added to the set V_{t-1}^1. Let c denote such an auxiliary color assigned to v_t. If this color is the same as $f(1)$ (the auxiliary color that is associated with entry 1), then the final color in the online conflict-free coloring of v_t is 1 and the updating process for the t-th vertex stops. Otherwise, if an auxiliary color cannot be found or if the assigned auxiliary color 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 color assigned to v_t is the same as $f(i)$. Then, the color of v_t in the final conflict-free coloring is set to i.

It is possible that v_t never gets a final color. In this case we say that the framework does not halt. However, termination can be guaranteed by imposing some restrictions on the auxiliary coloring method and the choice of the function f. For example, if first-fit is used for the auxiliary colorings 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 instantiation of the framework for conflict-free coloring with respect to intervals is given in the example in section 6. In section 4 we derive a randomized 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 coloring in case it halts.

LEMMA 3.1. *If the above framework halts for any vertex v_t then it produces a valid online conflict-free coloring 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 colored with j. We claim that exactly one such vertex in S exists. Assume to the contrary that there is another vertex v' in S colored 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 color (in the auxiliary coloring of the set V_t^j) which equals $f(j)$. However, since the auxiliary coloring is a proper non-monochromatic coloring 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 color different from $f(j)$. Thus, v'' got its final color 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 colorings 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 6.1 then the algorithm derived from the framework becomes identical to the unique maximum greedy algorithm described and analyzed in [9].
- 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 coloring graphs online. The performance of the first-fit greedy algorithm for restricted classes of graphs has been analyzed in several papers [10, 14, 12]. Especially for k-inductive graphs, the first-fit greedy algorithm is analyzed by Irani [12], who proved that it uses $O(k \log n)$ colors. Our framework can be used to give an alternative simpler proof of the aforementioned result (see [19] for details).

4. An online randomized conflict-free coloring algorithm. There is a randomized online conflict-free coloring algorithm in the oblivious adversary model that always produces a valid coloring and the number of colors 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 randomized online conflict-free coloring algorithm for H which uses at most $O(\log_{1+\frac{1}{4k+1}} n) = O(k \log n)$ colors 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 the function f, the set of auxiliary colors of each entry and the algorithm we use for the auxiliary coloring at each entry. We use the set $A = \{a_1, \ldots, a_{2k+1}\}$. For each entry i, the representing color $f(i)$ is chosen uniformly at random from A. We use a first-fit algorithm for the auxiliary coloring.

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 color 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 $2k$ neighbors 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 $2k+1$ colors available, there is always an available color to assign to such a vertex. The following lemma shows that if we use one of these available colors then the updated coloring is indeed a proper non-monochromatic coloring 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 color distinct from all its neighbors in the Delaunay graph $G(H(V^j_t))$ then this color together with the colors assigned to the vertices V^j_{t-1} is also a proper non-monochromatic coloring of the hypergraph $H(V^j_t)$.*

Proof. By induction on t. The induction hypothesis is that $H(V^j_{t-1})$ is properly non-monochromatically colored by the auxiliary coloring. 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 color that is distinct from any vertex u such that $\{u, v_t\}$ is a hyperedge of $H(V^j_t)$ (Those are exactly the neighbors 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 to not halt for 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 color 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 color a_1 in that entry and the probability that v_t does not get a main color in that entry is $1 - h^{-1}$, where $h = 2k + 1$ is the number of auxiliary colors. 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.

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

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

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

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

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

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 $\mathbf{E}_{\geq i+1} \leq (\frac{4k+1}{4k+2})^i n$. 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,

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

It is easily seen that $\mathbf{E}[X_i \mid X_{\geq i}] \geq \frac{1}{2}\frac{X_{\geq i}}{2k+1}$ since at least half of the vertices of $X_{\geq i}$ got an auxiliary color by the above algorithm. Moreover each of those elements that got an auxiliary color had probability $\frac{1}{2k+1}$ to get the final color 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,

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

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

LEMMA 4.5. *The expected number of colors used by the above 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 colored with a main color in entry i. We are interested in the number of colors used, that is $Y := \sum_{i=1}^{\infty} I_i$. Let $b(k,n) = \log_{\frac{4k+2}{4k+1}} n$. Then,

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

by Markov's inequality and lemma 4.4. ∎

We notice that:

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

We also have the following concentration result:

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

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 recoloring. In this section, we relax the requirement that an online algorithm has to commit to the color of every point, by allowing the algorithm to recolor 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 colors 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 colors 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 coloring must remain conflict-free at all times.

5.1. An $O(\log n)$ colors algorithm for intervals. We describe a deterministic online conflict-free coloring algorithm for intervals that is allowed to recolor 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 colors $\{a, b, c\}$ and f is the constant function $f(l) = a$, for every l. We refer to points colored with b or c as d-points. In order to have only a logarithmic number of entries, we slightly modify the framework (using a recoloring procedure) such that the number of points colored 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 colored with a (i.e., between every pair that are consecutively colored a among the a-points). Therefore, at least a third of the points at each entry get color a, and two thirds are deferred for coloring in a higher entry. The total number of colors is at most $\log_{3/2} n + 1$. When a new point p arrives, it is colored according to the following algorithm, starting from entry 1:

- If p is not adjacent to a point colored with an auxiliary color a then p is assigned auxiliary color a and gets its final color in that entry.
- We color 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 colored with a on one side and a point, say q, colored with d on the other side, such that q has no adjacent point colored with a. We assign to p the auxiliary color of q (thus it is a d-point) in the current entry and in all higher entries for which q obtained an auxiliary color and assign to it the main color of q, and we recolor q with the auxiliary color a (and delete the corresponding appearance of it in all higher entries of he table), and thus we recolor q with the main color of the current entry. At this point all points have an assignment of main colors. It is not hard to check that when we recolor a point then we do not violate the invariants at any entry: Let ℓ be the entry that caused recoloring, 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 run of the recoloring algorithm is shown in figure 5.1 for input $\pi = 3754612$. Vertex v_t appears at time t, where t ranges from 1 to 7. The first row of the table represents the order in which points appeared, the last row of the table shows current color allocation. At every time step of the run, points whose colors were changed (a new color, or a recoloring) 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 recoloring algorithm produces a valid conflict-free coloring, because it is essentially an instance of the general framework: After every insertion (and a possible recoloring), the point of highest entry in each interval is uniquely colored.

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 $\lfloor \log_2 n \rfloor + 1$ colors (see, for example, optimal static coloring of points with respect to intervals in [2]). Whenever a new

	·	·	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

FIG. 5.1. *An example run of the recoloring algorithm*

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

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 [1, 2]. 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 recoloring algorithm uses k colors and does $2^k - k - 1$ recolorings. The instance $\sigma^1 = 0$. For $k \geq 1$, the instance σ^{k+1} is defined recursively:

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

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

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

or

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

The following can be proven by induction and we omit the easy but tedious details. For each σ^k, the recoloring algorithm produces the coloring 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 colors. 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 colored with a new color $k+1$, and therefore no recoloring happens. For all subsequent $2^k - 1$ points inserted at relative position $2^k - 1$, there is a recoloring by the algorithm. Therefore, for all points, except the ones inserted at $t = 1, 2, 4, \ldots, 2^k, \ldots$ a recoloring happens, and therefore after n insertions, $n - (\lfloor \log_2 n \rfloor + 1)$ recolorings happen in σ. □

For example, the run of the recoloring algorithm on input σ^3 is shown in figure 5.2, where recolorings are shown with bold.

	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

FIG. 5.2. *The run of the recoloring algorithm on input* σ^3

5.2. An $O(\log n)$ colors 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 colored with respect to circular arcs with $\lfloor \log_2(n-1) \rfloor + 2$ colors: There must be a point p with unique color 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 colored with $\lfloor \log_2(n-1) \rfloor + 1$ more colors. 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 color 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 recoloring 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 2 immediate neighboring points, whereas in the intervals case, the two extreme points have only one neighbor). Again, in each entry ℓ, at least a third of the points is assigned auxiliary color a, and thus at most $\log_{3/2} n + 1$ colors are used.

5.3. An $O(\log n)$ colors algorithm for circular arcs with substitution of points. We consider a variation on the problem of online conflict-free coloring 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*:

> The adversary can substitute a set Q of already requested *consecutive* points with a single new point p, and the algorithm has to color p, such that the whole set of points is conflict-free colored 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 colors the new point like in section 5.2. For a substitution move of the adversary, it colors the new point p, with the *highest* color occurring in the points of Q. Point p also gets the entries of the unique point $q \in Q$ with the highest color. It is not difficult to see that the coloring remains conflict-free after each move. We remark that a recoloring can happen only in an insertion move and that substitution moves do not make the algorithm introduce new colors. The following is true for every t:

LEMMA 5.2. *After t moves, the above coloring algorithm uses at most $\log_{3/2} t + 1$ colors.*

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 we denote in each entry a point colored with a which was substituted by $\bar{a}$, then it can be proven that between any two consecutive points colored 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 colored either by that level or have been substituted. We call these points colored with $\bar{a}$ *ghost* points. Moreover, we assign ghost points to substitution points as follows. If a point p substitutes a point p' colored 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 colored with a and do not create a ghost point and any assignment in this case. It is not difficult to see that at any point in time 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 colors 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 color d in entry ℓ is bounded by the number of insertion moves that reached entry ℓ as follows.

$$d_\ell \leq \lceil \tfrac{2}{3} i_\ell \rceil \tag{5.1}$$

where d_ℓ is the number of points getting auxiliary color 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 colored with a. The number of d's increases only if there is an insertion move where the point at level ℓ is colored with d. We will study further this last case. For a new point p to get auxiliary color d it must be the case that it is inserted next to a point colored with a and a point q colored with d such that q is adjacent to a point colored with a. In a fixed entry ℓ, we call a maximal set of consecutive points colored with d a *strip*.

The length of a strip s is the number of d's in it and is denoted by $\mathrm{len}(s)$. 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 colored with a, and $\bar{a}_\ell$ the number of ghost points (points substituted that were colored with a). We have an inequality, because we omit the points substituted that were colored with d. If a strip s has length $\mathrm{len}(s) > 2$, it necessarily contains ghost points. In fact if a strip s has length $\mathrm{len}(s)$, one can prove that points in it have been assigned at least $\lceil \frac{1}{2}\mathrm{len}(s) \rceil - 1$ 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.

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

The above inequality implies

$$\lfloor \tfrac{2}{3} i_l \rfloor \geq \lfloor \tfrac{2}{3} \sum_{s\colon \text{strip}} \lceil \tfrac{3}{2}\mathrm{len}(s) \rceil \rfloor \geq \lfloor \sum_{s\colon \text{strip}} \mathrm{len}(s) \rfloor = \sum_{s\colon \text{strip}} \mathrm{len}(s) = d_\ell$$

which is inequality (5.1). ◻

LEMMA 5.3. *The points in a strip s have been assigned at least $\lceil \frac{1}{2}\mathrm{len}(s) \rceil - 1$ 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 $\lceil \frac{1}{2}\mathrm{len}(s) \rceil - 1 = 0$. We ignore trivial substitutions of one point colored 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 colors 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 in figure 5.3, i.e., the substitution is completely contained in one strip, say of length L', and the new strip created has length $L \leq L'$. In this case, the number of ghost points in the

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

FIG. 5.3. *A substitution move contained in one strip*

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 colored 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 in figure 5.4. 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

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

FIG. 5.4. *A substitution move spanning more than a strip*

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

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

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

5.4. An $O(\log n)$ colors algorithm for halfplanes. In this section we describe a deterministic algorithm for online conflict-free coloring points with respect to halfplanes that uses $O(\log n)$ colors and performs $O(n)$ recolorings. Thus, it can also be modified for conflict-free coloring 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):

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

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 coloring 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 color them with respect to circular arcs. The number of colors 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 colored as follows: If it is inside the convex hull of V_t, then it gets color '$\star$'. 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 color 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 recolored to '$\star$', and v_{t+1} is colored with the maximum color 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 colored 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 colors on t, one recoloring 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 recoloring happens on the new convex hull, and every point colored with '$\star$' keeps that color for the rest of the algorithm run.

6. Application to geometry. Our randomized algorithm has applications to conflict-free colorings of certain geometric hypergraphs studied in [5, 9, 13, 6]. We obtain the same asymptotic result as in [5] and [6] but with better constant of proportionalities 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 color for each entry is chosen uniformly at random from $\{a, b, c\}$, the expected number of colors used is bounded by $\log_{3/2} n + 1$. It is interesting that the best known upper bound for dynamically coloring n points deterministically, when the whole insertion order is known in advance, is also $\log_{3/2} n + 1$ (see, for example, [2] for further details). In our algorithm the expected number of colors is bounded by $1 + \log_{3/2} n \approx 1.71 \log_2 n$, whereas in [5] and [6] by $1 + \log_{8/7} n \approx 5.19 \log_2 n$, three times our bound. 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 coloring 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-colored (say with auxiliary colors a, b, c) as follows: Each newly inserted point has at most two immediate neighbors and thus even a first-fit coloring suffices. In figure 6.1, 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 $\pi_{(t)}$.

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 coloring, so far, is shown. Entries correspond to rows in the tables, where for each entry i the following data is given: the representing color $f(i)$ and the proper non-monochromatic auxiliary coloring of the vertices in the hypergraph V_t^i with three colors a, b or c.

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

When H is the hypergraph obtained by points in the plane intersected by halfplanes or unit disks, we obtain online randomized algorithms that use $O(\log n)$ colors 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 of definition 2.4 on page 4, it is enough to prove that for every t, we have

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

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

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

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

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

i	$f(i)$	v_4	v_1	v_3	·	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	·	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

FIG. 6.1. *A run example of the framework for hypergraphs induced by points with respect to intervals*

(we remark that we have dropped the permutation π, appearing in inequality (2.1) on page 4, 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 neighbors of v_t in the Delaunay graph $G(H(\{v_1, \ldots, v_t\}))$. For v_t to be a neighbor of $v_{t'}$, with $t' < t$, in the 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 6.1, because the size of the convex hull is the same as the number of points and every two points are neighbors in the Delaunay graph.

TABLE 6.1
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 6.2). Then v_t lies on

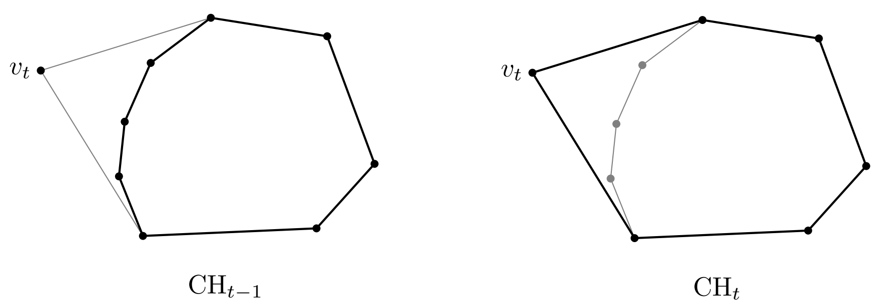

FIG. 6.2. *The new point is outside the old convex hull*

the boundary of the boundary of the convex hull CH_t. Consider the two points u and w that are the neighbors of v_t in the cyclic order of points on the convex hull CH_t (see figure 6.3). Consider the line ℓ passing through u and w. This line partitions points of

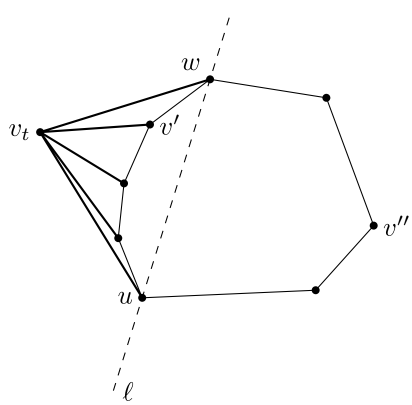

FIG. 6.3. *Delaunay graph neighbors of a new point outside the old convex hull*

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 6.3) and (b) points on the other halfplane defined by ℓ (points like v'' in figure 6.3). 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 neighbor 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 6.4). Then, consider any triangulation of CH_{t-1}. Point v_t is in exactly one triangle of the triangulation, call

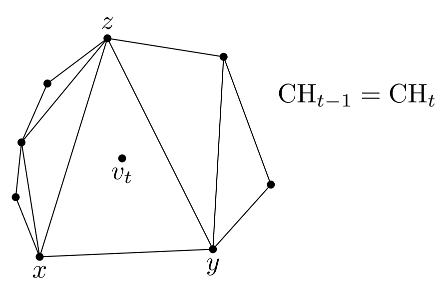

FIG. 6.4. *A triangulation of the convex hull in case the point v_t is in the convex hull CH_{t-1}*

it xyz, where x, y, z are points 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 neighbors in the Delaunay graph. The three neighbors case can be realized 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 neighbors 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 $\overrightarrow{pv_t}$, not containing p. These halflines are shown in figure 6.5. 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 as in lemma 6.1. Then, the expected number of colors used by our randomized online conflict-free coloring algorithm applied*

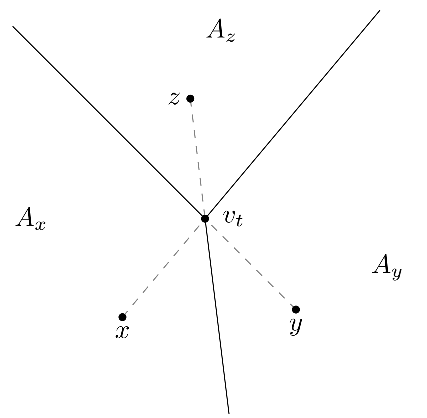

FIG. 6.5. *A partition of the plane*

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

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

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 randomized online algorithm for conflict-free coloring H which uses $O(\log n)$ colors and $O(\log n)$ random bits with high probability, against an oblivious adversary.*

Proof. In [13], it was observed that by an appropriate partitioning of the plane one can modify any online algorithm for conflict-free coloring points with respect to halfplanes to an online algorithm for conflict-free coloring points with respect to congruent disks. The congruent disks algorithm uses a constant multiple of the colors used by the halfplanes algorithm. Using the same technique as developed in [13] and corollary 6.2 we obtain the desired result. ∎

7. Discussion and open problems. We presented a framework for online conflict-free coloring any hypergraph. This framework coincides with some known algorithms in the literature when restricted to some special underlying hypergraphs. We derived a randomized online algorithm for conflict-free coloring 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 generalization 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; see definition 2.3). Specifically, if the hypergraph is k-degenerate then our algorithm uses $O(k \log n)$ colors 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 coloring for general hypergraphs and subsumes all the previous randomized algorithmic results of [5, 9, 13, 6]. It also substantially improves the efficiency with respect to the amount of randomness used in the special cases studied in [5, 9, 13, 6]. Another interesting fact to note is that our algorithm when applied to k-inductive graphs gives an online coloring of such graphs with $O(k \log n)$ colors with high probability. In [12], it was shown that the same bound can be achieved deterministically by the first-fit greedy algorithm.

It would be interesting to find an efficient online *deterministic* algorithm for conflict-free coloring 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 neighbors of the Delaunay graph of every induced hypergraph is at most two), the best known deterministic online conflict-free coloring algorithm from [9] uses $\Theta(\log^2 n)$ colors. We hope that our technique together with a possible clever derandomization 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 [9] for online conflict-free coloring points on a line with respect to intervals. No sharp asymptotic bounds are know for the performance of unique maximum greedy. The best known upper bound is linear (asymptotically $n/2$ from [1, 2]), whereas the best known lower bound is $\Omega(\sqrt{n})$. We believe that this new approach could help analyze the performance of unique maximum greedy.

In section 5 we initiate the study of online conflict-free coloring with recoloring: We provide a deterministic online conflict-free coloring for points on the real line with respect to intervals and show that our algorithm uses $\Theta(\log n)$ colors and at most one recoloring per insertion. This is in contrast with the best known deterministic online conflict-free coloring for this case that uses $\Theta(\log^2 n)$ colors in the worst case without recoloring, by [9]. We also present deterministic online algorithms for conflict-free coloring points with respect to circular arcs and halfplanes (and unit disks) that use $O(\log n)$ colors and $O(n)$ recolorings in the worst case. In the special case of intervals or circular arcs at most one point is recolored per insertion.

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

All of our randomized 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 and the expected number of colors is analyzed. The randomized 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 colors. Another such relaxation is to give extra information to the algorithm about where each requested point will end up in the final coloring (the *absolute positions* model, which was introduced in [1, 2]). Other such relaxations are given in [1, 2] (coloring with respect to rays) and [16] (online ranking of paths). In this work we introduced yet another relaxation, the recoloring model, in which the algorithm is allowed to recolor some of the points. An interesting question is to construct $O(\log n)$ colors 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: randomized algorithm that use an expected logarithmic number of random bits, and recoloring 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] Amotz Bar-Noy, Panagiotis Cheilaris, and Shakhar Smorodinsky, *Conflict-free coloring for intervals: from offline to online*, in Proceedings of the 18th Annual ACM Symposium on Parallelism in Algorithms and Architectures (SPAA), 2006, pp. 128–137.

[2] ———, *Deterministic conflict-free coloring for intervals: from offline to online*, ACM Transactions on Algorithms, 4 (2008), pp. 44:1–44:18.

[3] Allan Borodin and Ran El-Yaniv, *Online computation and competitive analysis*, Cambridge University Press, 1998.

[4] Panagiotis Cheilaris, *Conflict-free coloring*, PhD thesis, City University of New York, 2008.

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

[6] Ke Chen, Amos Fiat, Haim Kaplan, Meital Levy, Jiří Matoušek, Elchanan Mossel, János Pach, Micha Sharir, Shakhar Smorodinsky, Uli Wagner, and Emo Welzl, *Online conflict-free coloring for intervals*, SIAM Journal on Computing, 36 (2007), pp. 1342–1359.

[7] Khaled Elbassioni and Nabil H. Mustafa, *Conflict-free colorings of rectangles ranges*, in Proceedings of the 23rd International Symposium on Theoretical Aspects of Computer Science (STACS), 2006, pp. 254–263.

[8] Guy Even, Zvi Lotker, Dana Ron, and Shakhar Smorodinsky, *Conflict-free colorings of simple geometric regions with applications to frequency assignment in cellular networks*, SIAM Journal on Computing, 33 (2003), pp. 94–136. Also in *Proceedings of the 43rd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, 2002.

[9] Amos Fiat, Meital Levy, Jiří Matoušek, Elchanan Mossel, János Pach, Micha Sharir, Shakhar Smorodinsky, Uli Wagner, and Emo Welzl, *Online conflict-free coloring for intervals*, in Proceedings of the 16th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), 2005, pp. 545–554.

[10] András Gyárfás and Jenö Lehel, *On-line and first fit coloring of graphs*, Journal of Graph Theory, 12 (1988), pp. 217–227.

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

[12] Sandy Irani, *Coloring inductive graphs on-line*, Algorithmica, 11 (1994), pp. 53–72.

[13] Haim Kaplan and Micha Sharir, *Online CF coloring for halfplanes, congruent disks, and axis-parallel rectangles*. Manuscript, 2004.

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

[15] János Pach and Géza Tóth, *Conflict free colorings*, in Discrete and Computational Geometry, The Goodman-Pollack Festschrift, Springer Verlag, 2003, pp. 665–671.

[16] Ingo Schiermeyer, Zsolt Tuza, and Margit Voigt, *On-line rankings of graphs*, Discrete Mathematics, 212 (2000), pp. 141–147.

[17] Shakhar Smorodinsky, *Combinatorial Problems in Computational Geometry*, PhD thesis, School of Computer Science, Tel-Aviv University, 2003.

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

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