Online Conflict Free Coloring for Halfplanes, Congruent Disks, and Axis-Parallel Rectangles*

Ke Chen† Haim Kaplan‡ Micha Sharir§

April 17, 2008

Abstract

We present randomized algorithms for online conflict-free coloring (CF in short) of points in the plane, with respect to halfplanes, congruent disks, and nearly-equal axis-parallel rectangles. In all three cases, the coloring algorithms use $O(\log n)$ colors, with high probability.

We also present a deterministic algorithm for online CF coloring of points in the plane with respect to nearly-equal axis-parallel rectangles, using $O(\log^3 n)$ colors. This is the first efficient (that is, using polylog(n) colors) deterministic online CF coloring algorithm for this problem.

*Work on this paper by Ke Chen was partially supported by a NSF Grant CCR-0132901. Work by Micha Sharir was partially supported by NSF Grant CCF-05-14079, by a grant from the U.S.-Israeli Binational Science Foundation, by Grant 155/05 from the Israel Science Fund, and by the Hermann Minkowski–MINERVA Center for Geometry at Tel Aviv University.

†Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, IL 61801, USA; kechen@uiuc.edu.

‡School of Computer Science, Tel Aviv University, Tel Aviv 69978, Israel; haimk@post.tau.ac.il

§School of Computer Science, Tel Aviv University, Tel Aviv 69978, Israel, and Courant Institute of Mathematical Sciences, New York University, New York, NY 10012, USA; michas@post.tau.ac.il

1 Introduction

A *range space* $(X, \mathcal{R})$ is defined by a ground set X and a family $\mathcal{R}$ of subsets of X, which are called *ranges* (for example, $X = \mathbb{R}^2$ and $\mathcal{R}$ is the set of all disks in the plane). A coloring of a finite set $P \subseteq X$ is *conflict-free* (CF for short) for $\mathcal{R}$ if for any range $R \in \mathcal{R}$ with $P \cap R \neq \emptyset$, there is at least one point in $P \cap R$ that has a unique color among the points of $P \cap R$. Namely, for any nonempty range $R \in \mathcal{R}$, there is a color that appears exactly once in the set $P \cap R$.

The problem of CF coloring is motivated by frequency assignment in wireless networks. Specifically, in the context studied in this paper, the points of P are *base stations* (or *antennas*) with a fixed transmission radius r, and the ranges are disks of radius r, centered at the *clients*. The colors are frequencies assigned to the antennas, and the conflict-free property means that any client can always find a frequency that is assigned to a unique antenna, among those that it can reach. In this case the communication with that antenna is free from interference with other antennas that are assigned the same frequency. The goal is then to minimize the number of distinct frequencies assigned to the antennas, while maintaining the conflict-free property.

The problem was introduced by Even *et al.* [11]. They showed that one can find an assignment of $O(\log n)$ frequencies to the base stations which is conflict-free for disks in the plane, and this is tight in the worst case. Har-Peled and Smorodinsky [12] extended those results by considering other range spaces. They gave sufficient conditions for the CF chromatic number to be small for more general ranges. The dual version of the CF coloring problem was studied by Even *et al.* and Har-Peled and Smorodinsky [11, 12], where one colors the ranges so that, for any point, the set of ranges that contain the point is conflict-free. Recently, Smorodinsky [15] improved several results studied by Even *et al.* [11] by providing a deterministic coloring algorithms for those problems.

The problem has been extended to a dynamic scenario, in which the points of P (the base stations) are inserted one by one, starting with an empty set [8]. When a point is inserted, a color is assigned to it and the color cannot be changed later. The coloring should remain conflict-free at all times. Chen *et al.* [8] considered the case where P is a set of n points on the line, and $\mathcal{R}$ is the set of all intervals on the line. They present both deterministic and randomized algorithms for the problem. The best deterministic algorithm uses $O(\log^2 n)$ colors, and the best randomized algorithm uses $O(\log n)$ colors with high probability. The best known lower bound for both randomized and deterministic algorithms, which also holds in the static case, is $\Omega(\log n)$ colors [13, 14]. For more interesting variations on the online CF coloring problem for intervals see Bar-Noy *et al.* [3].

The paper of Chen *et al.* [8] contains one negative result concerning online CF coloring of points in the plane for arbitrary disks as ranges. It shows that in the worst case n colors are needed (by any coloring algorithm). That is, there are situations where each newly inserted point requires a new color. (Recall, in contrast, that $O(\log n)$ colors suffice for the static case.)

Our results. The starting point of our work has been the lower bound of Chen *et al.* [8] for CF coloring of points in the plane with respect to arbitrary disks as ranges. This lower bound is particularly discouraging since the wireless application does involve circular ranges in the plane.

Our goal has therefore been to restrict the problem so that the lower bound does not apply. One possible such restriction is to fix the radius of the disks. The lower bound proof for general disks uses disks of varying sizes (actually, they are small perturbations of a unit disk) and does not apply for unit disks. On the other hand, the case of unit disks is still a natural model for the frequency assignment application, as explained above.

The one-dimensional analog of this restriction is online CF coloring of points on the line for *unit intervals*. As was observed by Chen *et al.* [8], this problem is much easier than online CF

coloring for arbitrary intervals, and can be solved by a simple deterministic algorithm that uses only $O(\log n)$ colors.

Our main result, presented in Section 4, is a randomized algorithm for online CF coloring of points in the plane for unit disks, that uses $O(\log n)$ colors with high probability. The algorithm is a generalization of the randomized algorithm of Chen *et al.* [8] for CF coloring points on the line for intervals. As in the algorithm of Chen *et al.*, we also use the positive integers as the colors, and guarantee that the largest integer in each range is unique. The analysis of our algorithm, however, is more delicate than the one of Chen *et al.*, and does make use of the geometry of the problem. It is based on an observation that allows us, in certain cases, to charge a high color assigned to a point, to the disappearance of previously inserted points from the boundary of the convex hull of an appropriate subset of the high-colored points inserted so far. These charges imply that the expected number of points that require color at least j decreases exponentially in j, thereby implying the logarithmic bound on the number of colors.

We start in Section 2 by reviewing the randomized algorithm of Chen *et al.* [8] for CF coloring points on the line for intervals. Then we continue in Section 3 with a related problem of online CF coloring of points in the plane for *halfplanes*. This is a simple generalization of the one-dimensional CF coloring problem. Indeed, if we restrict the two-dimensional problem to sets P of points on the upper unit semi-circle, and map the inserted points by projecting them on the x-axis, then the subsets of P that can be cut off the unit circle by halfplanes, when projected on the x-axis, are the same as the subsets of the projected set that can be cut off by intervals, or by complements of intervals.

The case of halfplanes is simpler than that of unit circles. However, it already demonstrates how geometry enters the analysis in a nontrivial manner. We present a randomized algorithm for this problem that uses $O(\log n)$ colors with high probability. In section 4 we extend this technique to the case of unit disks, using similar machinery.

In Section 5 we extend the approach to the problem of online CF coloring of points in the plane for nearly equal axis-parallel rectangles, namely, rectangles for which the ratio between the largest and the smallest widths, and the ratio between the largest and the smallest heights, are both bounded by some constant. Here too we obtain a coloring that uses, with high probability, $O(\log n)$ colors.

Notice that the offline version of all the problems we consider in this paper is quite easy. Offline CF coloring of n points for any of the three kinds of ranges mentioned above can be done with $O(\log n)$ colors, using, for example, the approach of Har-Peled and Smorodinsky [12]. We recall that the known lower bound to the above problems, which also holds in the static cases, is $\Omega(\log n)$ colors [11, 13].

Finally in Section 6 we present a deterministic online algorithm for CF coloring with respect to nearly-equal axis-parallel rectangles in the plane. The algorithm uses $O(\log^3 n)$ colors.

The bounds stated so far are all combinatorial in nature, in the sense that they bound the number of colors produced by the various algorithms. It is also of interest to study the efficiency of the algorithms themselves, an issue that we also address in this paper. Specifically, we show how to implement the algorithms of Sections 3-5 so that it takes $O(\log^2 n)$ time with high probability to color each newly inserted point, and we show how to implement the deterministic algorithm of Section 6 so that we color each newly inserted point in $O(\log^2 n)$ (deterministic) time.

A preliminary version of the results in Sections 3, 4, and 5, as well as a deterministic online algorithm for CF coloring with respect to nearly-equal axis-parallel rectangles that uses $O(\log^{12} n)$ colors, appeared in [7].

Computational model. When analyzing randomized online algorithms, there is a distinction between the *oblivious adversary* model and the *adaptive adversary* model. The oblivious adversary must construct the entire input sequence in advance, while the adaptive adversary may choose each input point based on the actions of the online algorithm made so far. We refer the interested reader to Borodin and El-Yaniv [6] for a discussion of these models. The analysis of all our algorithms is in the (weaker) oblivious adversary model. There are no known online algorithms for CF coloring against an adaptive adversary using only $O(\log n)$ colors. (The $O(\log n)$ algorithm of Chen *et al.* [8] for the 1-dimensional case also works only against an oblivious adversary.) In fact, it is an open question of whether one can bound the number of colors used by any of our algorithms when the adversary is adaptive. (This is also open for the randomized algorithm of Chen *et al.* [8].)

Recently, Bar-Noy *et al.* [4] suggested a different randomized algorithm for CF coloring with respect to intervals, halfplanes, unit disks, and nearly-equal axis-parallel rectangles. Their algorithms also use $O(\log n)$ colors, with high probability, against an oblivious adversary, but need fewer random bits than our algorithms.

2 Online CF Coloring for Intervals: A Brief Overview

To motivate our 2-dimensional algorithms we first review the randomized algorithm of Chen *et al.* [8] for online CF coloring of points on the line for interval ranges. As already mentioned, we identify the colors with the integers, so that there is a total order on the set of colors. The coloring produced by the algorithm is such that the *maximum* color in any (nonempty) interval is unique.

Let p be the next point inserted. We say that p *sees* a point x (alternatively, p sees the color $c(x)$) if all the colors of points between p and x (exclusive) have color smaller than $c(x)$.[1] We say that p is *eligible* for color m if p does not see m. To give p a color, we scan all colors in increasing order. For each color i, if p is not eligible for color i we continue to color $i+1$. Otherwise, if p is eligible for color i, we set $c(p) = i$ with probability $1/2$, and continue to color $i+1$ with probability $1/2$.

It is easy to prove, by induction on the insertion order, that the maximum color in any interval is unique at any stage. To show that this algorithm uses $O(\log n)$ colors with high probability, one argues that if the algorithm reaches color i when processing a point p, then p gets the color i with probability at least $1/8$. More formally, let C_i (resp., $C_{\geq i}$) be the (random variable) set of points of color i (resp., of color $\geq i$). Then

$$\mathbf{Pr}\Big\{p \in C_i \mid p \in C_{\geq i}\Big\} \geq \frac{1}{8}\ .$$

To see this, assume that p is neither the leftmost nor the rightmost point in $C_{\geq i}$ at the time of its insertion, and let q, r be its left and right neighbors in that set. In order for p to get color i, it is sufficient that both q and r "advance" to higher colors, and that p "stays" at color i. The first two events happen together with probability at least $1/4$, and the conditional probability of the third event, conditioned on the first two occurring (and on p reaching $C_{\geq i}$), is $1/2$, since p does not see color i.[2] Hence, the probability of p to be in C_i, assuming it is in $C_{\geq i}$, is at least $1/8$, as claimed (the argument is simpler, and the probability is larger, when p is the leftmost or rightmost point in $C_{\geq i}$).

[1] Alternatively, we can say that p *sees* x if there is an interval containing p and x and no other point with color higher than the color of x.

[2] Note that this analysis strongly uses the fact that the adversary is oblivious.

This implies that

$$\mathbf{E}\left(|C_{\geq i+1}|\right) \leq \frac{7}{8}\mathbf{E}\left(|C_{\geq i}|\right) \ .$$

Since $|C_{\geq 1}| = n$, we have, for $i \geq 1$,

$$\mathbf{E}\left(|C_{\geq i+1}|\right) \leq \left(\frac{7}{8}\right)^i n \ .$$

For $i = c \log_{8/7} n$, we get that $\mathbf{E}\left(|C_{\geq i+1}|\right) \leq 1/n^{c-1}$. Hence, by Markov's inequality,

$$\mathbf{Pr}\left\{|C_{\geq i+1}| \geq 1\right\} \leq 1/n^{c-1} \ ,$$

which shows that, with high probability, the algorithm uses only $i = O(\log n)$ colors.

Efficient implementation. Chen *et al.* [8] have not considered at all the issue of an efficient implementation of their algorithm. To supplement their analysis, and to introduce a central tool to be used by our algorithms, we present here such an implementation so that an insertion of a point takes $O(\log n)$ time. This is done by storing the points in a *cartesian tree* T [16], which is defined recursively as follows. The root of T contains the point p of maximum color; the left subtree is a cartesian tree of the points to the left of p or a single leaf if there are no such points, and the right subtree is a cartesian tree of the points to the right of p or a single leaf if there are no such points. The tree T is well defined since our coloring guarantees that the maximum in each interval of points which corresponds to a subtree is unique. Note that since we use $O(\log n)$ colors, the depth of T is $O(\log n)$.

Since the symmetric order (inorder) of the points in T corresponds to the order of the points on the line from left to right, we can use T as a search tree. Furthermore we can partition the line into disjoint intervals such that each leaf v of T corresponds to one of these intervals, say $I(v)$, in the sense that a search with a point $q \in I(v)$ (q is different from the points already inserted) terminates at v. Thus leaves of T represent gaps between consecutive points, whereas internal nodes represent the point themselves.

To insert a new point p, we first locate the leaf v such that $p \in I(v)$. It is straightforward to verify that all colors which p sees correspond to points on the path from the root of T to v. Indeed, any such point q has the maximum color in some interval containing p, and thus p sees q, by definition. Conversely, any point r not on the path is not seen from p, because the least common ancestor of p and r (which does lie on the path) has larger color, which "hides" r from p. By traversing this path bottom-up we identify all colors for which p is eligible in increasing order. We flip a coin for each of these colors until we find a color for p.

Finally, we have to add p to T. Let u be the deepest node on the path from v to the root which has a point with color larger than the color of p. Let w be the child of u which is an ancestor of v. (Note that u may be equal to v and then w is not defined.) We allocate a new node v_p that contains p, and make v_p a child of u, instead of w. Finally we split the subtree rooted at w into two subtrees, rooted at the two respective children of v_p. We perform this split as follows.

Let $x_1, \ldots, x_k$ be the nodes on the path from w to v, whose associated points are to the right of p, ordered by increasing depth. For each $2 \leq i \leq k$, we make x_i the left child of x_{i-1}, if it is not already the case. The resulting tree, rooted at x_1, is the right subtree of v_p. We construct the left subtree of v_p similarly.

Since the depth of T is $O(\log n)$ with high probability this algorithm clearly takes $O(\log n)$ time with high probability, for each newly inserted point.

3 Online CF Coloring for Halfplanes

In this section we present an algorithm for CF coloring points in the plane for halfplane ranges. The algorithm is similar to the one-dimensional algorithm of Section 2 but with a different definition of when a point p sees a color m. To simplify the presentation, we assume that the points of P are in *general position*, namely that no three of them are collinear.

Let p be the next point to be inserted. We say that p *sees* a point x *at level* i (or i*-sees* x, for short) if $c(x) \geq i$ and there is a halfplane h that contains x and p and no other point of color $\geq i$. We say that p *sees the color* c *at level* i if $c \geq i$ and p sees some point of color c at level i. We say that p is *eligible* for color m if p does not see the color m at level m. Equivalently, p is eligible for color m if each halfplane h containing p either contains no point of color m or contains at least two points of color $\geq m$. To give p a color, we scan all colors in increasing order. For each color i, if p is not eligible for color i we continue to color $i+1$. Otherwise, if p is eligible for color i, we set $c(p) = i$ with probability $1/2$, and continue to color $i+1$ with probability $1/2$.

It is easy to prove by induction that the maximum color in any halfplane is unique at any stage. Indeed, consider a halfplane h at some stage which contains at least two points of maximum color i (among those of the current points in h). Let p be the second inserted point among those lying in h and receiving color i. By definition, when p was inserted it saw color i at level i (with h as a "witness" halfplane), and therefore was not eligible for this color, contradicting the assumption that it got color i.

We next show that the algorithm uses $O(\log n)$ colors with high probability. Let C_i (resp., $C_{\geq i}$) be the set of points of color i (resp., of color $\geq i$). Let $B_{\geq i} \subseteq C_{\geq i}$ be the set of those points $p \in C_{\geq i}$ that i-see at least four other points of $C_{\geq i}$ when they are inserted. Let $E_{\geq i} = C_{\geq i} \setminus B_{\geq i}$. All these sets are (set-valued) random variables, depending on the random choices made by the algorithm. (Recall that all points of $C_{\geq i}$ have gone through the eligibility test for color i when they were inserted, some passing and some failing that test.)

Lemma 3.1. *When the algorithm for coloring point p reaches color i, and p i-sees at least four points of $C_{\geq i}$ that were inserted before it (i.e., $p \in B_{\geq i}$), then p must lie outside the convex hull of the previously inserted points of $C_{\geq i}$.*

Proof. Let A be the set of points of $C_{\geq i}$ inserted before p, and let $CH(A)$ denote the convex hull of A. Assume to the contrary that $p \in B_{\geq i}$ and $p \in CH(A)$. We claim that p can i-see only *vertices* of $CH(A)$. Indeed, if p i-sees an interior point q, then any witness halfplane containing p and q must also contain a vertex of $CH(A)$, which has color $\geq i$, contradicting the definition of i-seeing.

By Carathéodory's Theorem, there exist three vertices q_1, q_2, q_3 of $CH(A)$ such that p lies in the triangle $q_1q_2q_3$. A similar argument to the one just given shows that p cannot i-see any other vertex of $CH(A)$, contradicting the assumption that $p \in B_{\geq i}$. See Figure 1. □

Let $f = |C_{\geq i}|$ and let $p_1, \ldots, p_f$ be the points in $C_{\geq i}$ in the order in which they were inserted. For $1 \leq j \leq f$, let $K_j = CH(\{p_1, \ldots, p_j\})$ (the convex hull of $\{p_1, \ldots, p_j\}$). By Lemma 3.1, if $p_j \in B_{\geq i}$ then $K_j \neq K_{j-1}$. In this case, p_j is a vertex of K_j and all the (at least four) points that p i-sees when it is inserted are consecutive vertices of K_{j-1}. All these vertices except the first and the last are not vertices of K_j, and, since the hulls keep growing, nor are they vertices of any K_ℓ, for $\ell > j$. Thus each point $p_j \in B_{\geq i}$ removes at least two vertices from K_{j-1}, and no point of P is removed more than once. See Figure 1. This implies that $|B_{\geq i}| \leq \frac{1}{2}|C_{\geq i}|$ and thus $|E_{\geq i}| \geq \frac{1}{2}|C_{\geq i}|$.

Lemma 3.2.

$$\mathbf{Pr}\Big\{p \in C_i \mid p \in E_{\geq i}\Big\} \geq \frac{1}{16}\ .$$

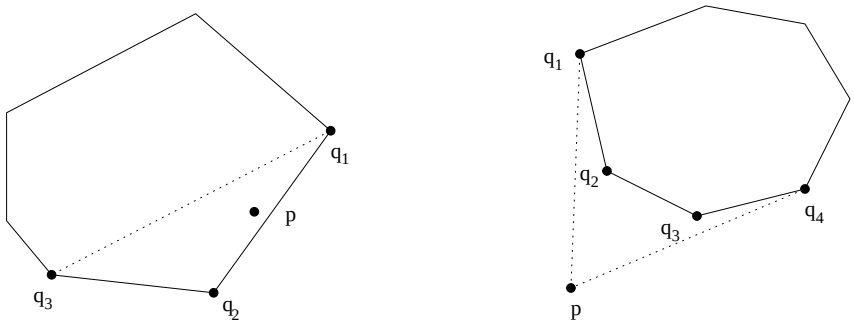


Figure 1: A point $p \in C_{\geq i}$ and the convex hull of the points in $C_{\geq i}$ inserted before p. (a) If p is inside the hull then it can i-see at most 3 points (hull vertices) of $C_{\geq i}$. (b) p is outside the hull and it i-sees q_1, q_2, q_3, and q_4. Thus p is in $B_{\geq i}$, and its insertion removes q_2 and q_3 from the hull boundary.

Proof. Fix the set $C_{\geq i}$ and consider only the coin flips that assign colors to the points of $C_{\geq i}$, after the points did reach $C_{\geq i}$ (note that, once $C_{\geq i}$ is fixed, the subsets $B_{\geq i}$ and $E_{\geq i}$ are also determined). Assume that $p \in E_{\geq i}$. By definition, the probability that p gets color i is $1/2$ the probability that p is eligible for color i.

Point p is eligible for color i if all the points of $C_{\geq i}$ that it i-sees when it is inserted did not get color i (note that this is only a sufficient condition). Since p i-sees at most three points of $C_{\geq i}$, the probability that none of them got color i is at least $1/8$. □

We thus obtain the following theorem.

Theorem 3.3. *The online CF coloring algorithm of points for halfplane ranges, presented in this section, uses $O(\log n)$ colors with high probability. More precisely, it uses at most $c \log n$ colors with probability at least $1 - \frac{1}{n^{\alpha c}}$, where α is some absolute constant.*

Proof. Using the same notation as above, since $|E_{\geq i}| \geq |C_{\geq i}|/2$, and since by Lemma 3.2 a point in $E_{\geq i}$ gets color i with probability $\geq 1/16$, we obtain that

$$\mathbf{E}\left(|C_{\geq i+1}|\right) \leq \left(1 - \frac{1}{32}\right) \mathbf{E}\left(|C_{\geq i}|\right) \ .$$

Since $|C_{\geq 1}| = n$, we have, for $i \geq 1$,

$$\mathbf{E}\left(|C_{\geq i+1}|\right) \leq \left(\frac{31}{32}\right)^i n \ .$$

For $i = b \log_{32/31} n$, we get that $\mathbf{E}\left(|C_{\geq i+1}|\right) \leq 1/n^{b-1}$. Hence, by Markov's inequality,

$$\mathbf{Pr}\Big\{ |C_{\geq i+1}| \geq 1 \Big\} \leq 1/n^{b-1} \ ,$$

from which the theorem follows. □

Efficient implementation. When a new point p arrives, we need to determine efficiently if it is eligible for color m; that is, whether it sees m at level m. Recall that p sees m at level m if and only if there is a halfplane h containing p, another point, say x, of color m, and no other point of color $\geq m$.

In the dual plane, each point x inserted so far (including p itself) is mapped to a line ℓ_x, and we associate with ℓ_x the color of x. Let $L_{\geq m}$ denote the set of all lines inserted so far, whose color is at least m (equivalently, the lines which have reached level m). Let $\mathcal{A}_{\geq m}$ denote the arrangement of these lines. We want to decide whether there exists a point h such that either above h or below h there are exactly two lines of $L_{\geq m}$, so that one of them is ℓ_p and the other has color m. We focus on the test above h; the other case is treated in a fully symmetric manner.

To perform this decision, we maintain the upper envelope E (level 0) and the level just below it (level 1) of the arrangement $\mathcal{A}_{\geq m}$. Each of these levels is an x-monotone connected sequence of segments, delimited by vertices of the arrangement. Note that the vertices of E also delimit segments of level 1 (which have additional vertices at level 1 too). See Figure 3. We can naturally split the segments of level 1 into subsequences, where each subsequence is a convex chain of segments, starting and ending at two respective adjacent vertices a, b of E. Let ℓ be the line containing a and b; it contributes the segment ab to the envelope. We call the subsequence of segments of level 1 from a to b the *pocket* of ℓ.

The decision involving a newly inserted point p is then carried out as follows. We test whether ℓ_p intersects the current upper envelope E. If so, we test whether any of the lines that lie on E below ℓ_p (including the at most two lines that ℓ_p crosses on E) has color m. If so, then, as easily verified, p is not eligible for color m, and otherwise it is eligible. See Figure 3(a). If ℓ_p does not intersect E then it is eligible for color m if and only if it does not intersect any pocket of a line of level m. See Figure 3(b). The following lemma specifies where these pockets may be located.

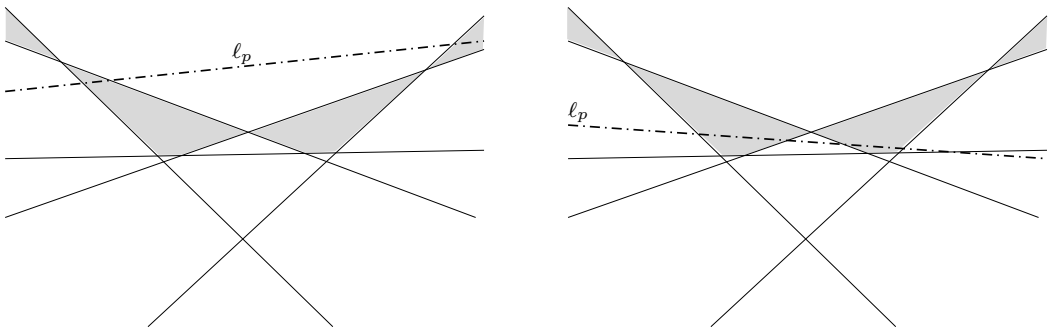


Figure 2: Adding a dual line ℓ_p and testing the eligibility of p for color m. The pockets of the first level are highlighted. (a) ℓ_p crosses the upper envelope. (b) ℓ_p passes below the upper envelope.

Lemma 3.4. *Let ℓ be a line which does not intersect the upper envelope E of a set of lines L. Let b be the vertex on E such that the slope of the line ℓ' preceding b on E is smaller than the slope of ℓ, and the slope of the line ℓ'' following b on E is larger than b. Then ℓ can intersect only the pockets of ℓ' and ℓ''.*

Proof. The line ℓ cannot intersect the line ℓ'' to the right of b since its slope is smaller than that of ℓ''. Therefore, the only pocket that ℓ can intersect to the right of b is the pocket of ℓ''. Symmetrically, the only pocket that ℓ can intersect to the left of b is the pocket of ℓ'. □

Hence, to determine whether ℓ_p is eligible for color m when ℓ_p does not intersect the upper envelope E, we locate, using binary search over the vertices of E, the vertex b for which the slope

of ℓ lies between the slopes of its two incident lines, ℓ', ℓ''. Then, if ℓ' (resp., ℓ'') has color m, we test whether ℓ_p crosses its pocket, If so, then p is not eligible for color m, and otherwise it is eligible.

To implement this algorithm, we maintain the vertices and segments of the upper envelope E of $L_{\geq m}$ in a balanced search tree T_m^0. Each internal node of T_m^0 also maintains the minimum color of a segment (i.e., of the line containing the segment) in its subtree. With this data structure, it is easy to determine whether ℓ_p intersects the upper envelope and, if it does, to determine the smallest color of a line in the portion of the envelope below ℓ_p, in overall $O(\log n)$ time.

For each line of color m on the envelope, we maintain the chain of level-1 segments in its pocket, also in a balanced search tree. If ℓ_p does not intersect E, then we find, by binary search, the vertex b on the envelope, as specified in Lemma 3.4 above. Then we check, for each of the two lines incident to b, whether it is of color m, and, if so, whether ℓ_p intersects its pocket. We do the latter step by searching in the tree of the pocket. Altogether, this test takes $O(\log n)$ time.

When a new line gets color m, we also insert it into the data structures of $L_{\geq j}$ from every $j \leq m$. Each such insertion may require splitting and concatenating a few search trees (the one representing the envelope, and up to two representing the pockets that ℓ_p crosses). Using an implementation which supports these operations, we can perform each insertion in $O(\log n)$ time for each threshold color m. Since by Theorem 3.3 the number of colors is $O(\log n)$ with high probability, we obtain that the overall cost per insertion is $O(\log^2 n)$ time with high probability.

4 Online CF Coloring for Congruent Disks

We next extend the analysis of the preceding section to the case where the ranges are congruent disks of common radius 1, say. We tile the plane with axis-parallel squares of side $1/2$, and assign to each of them a color class, so that no unit disk intersects two distinct squares with the same color class, and so that the total number of classes is a constant. Within each square we color the points independently, using the colors of the class assigned to the square.

Let Q be a square in the tiling. The coloring procedure for points in Q is identical to the one given for halfplanes, except that we say that p *i-sees* a point x if $c(x) \geq i$ and there is a unit disk D that contains x and p and no other point of color $\geq i$. We say that p *i-sees the color c at level i* if $c \geq i$ and p sees some point of color c at level i. We say that p is *eligible* for color m if p does not m-see the color m, and apply the online coloring algorithm of Section 3 to the points in Q.

Correctness follows by induction, as in the preceding section, showing that for any unit disk D that contains points from a square Q, the maximum color of the points of $Q \cap D$ is unique.

We next bound the number of colors used by the algorithm. For any unit disk D that intersects Q, the center of D lies in an axis-parallel square Q_0 that is concentric with Q and has side length $5/2$. Partition Q_0 into four disjoint equal sub-squares, $Q_0^1, \ldots, Q_0^4$, each an axis-parallel square of side length $5/4$, and all having the center of Q as a common vertex. See Figure 3. Let $o^1, \ldots, o^4$ be the centers of $Q_0^1, \ldots, Q_0^4$, respectively. It is easy to check that a unit disk centered at o^d contains Q_0^d, for $d = 1, \ldots, 4$. This implies that each unit disk which intersects Q, contains at least one of the points $o^1, \ldots, o^4$. We arbitrarily associate each such unit disk with one of the points among $o^1, \ldots, o^4$ that it contains. We denote by $\mathcal{D}^d$ the set of unit disks associated with o^d. The following is a crucial property of this partitioning.

Lemma 4.1. *Let K^d denote the convex cone with apex o^d spanned by Q, for $d = 1, \ldots, 4$. Then, for any pair of disks $D, D' \in \mathcal{D}^d$, the intersection $\partial D \cap \partial D' \cap K^d$ consists of at most one point.*

Proof. Note that the opening angle of each of the cones K^d is smaller than $\pi/2$. Assume to the contrary that $\partial D \cap \partial D' \cap K^d$ contains two points, say x and y. Then $D \cap D' \cap K^d$ contains the

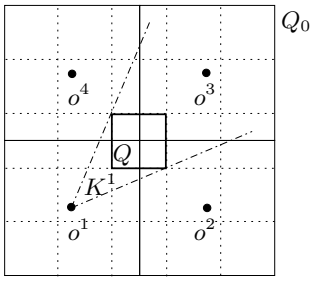


Figure 3: The partition of Q_0 into four sub-squares and the corresponding stabbing points o^d. The cone K^1 with apex o^1 spanned by Q is also shown.

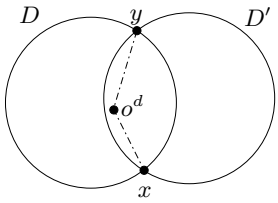


Figure 4: If $o^d \in D \cap D'$, the angle $\angle xo^d y$ has to be obtuse.

triangle $xo^d y$, which is easily seen to imply that the angle $\angle xo^d y$ is greater than $\pi/2$; see Figure 4. This however is impossible, since this angle is smaller than the opening angle of K^d. □

Let C_i (resp., $C_{\geq i}$) be the random variable which is equal to the set of points of color i (resp., of color $\geq i$). Let $B_{\geq i} \subseteq C_{\geq i}$ be the random variable that consists of any point $p \in C_{\geq i}$ that i-sees more than 36 other points of $C_{\geq i}$ when it is inserted. Let $E_{\geq i} = C_{\geq i} \setminus B_{\geq i}$.

In section 3 we controlled the sizes of the analogous sets $B_{\geq i}$, $E_{\geq i}$ by arguing that when a point of $B_{\geq i}$ is inserted, it removes at least two points from being vertices of the convex hull of $C_{\geq i}$ from this point on. To extend the argument to the case of unit disks, we replace the notion of convex hull vertices by *d-maximal vertices*, defined as follows.

Let I be a set of points in Q. For $d = 1, 2, \ldots, 4$, we define a point $p \in I$ to be *d-maximal* if there is a disk in $\mathcal{D}^d$ that contains p and no other point of I. Let $M^d(I)$ denote the subset of the d-maximal points in I.

Lemma 4.2. *Let $f = |C_{\geq i}|$ and let $p_1, \ldots, p_f$ be the points in $C_{\geq i}$ in the order in which they are inserted. Let $A_j^d = M^d(\{p_1, \ldots, p_j\})$ for $d = 1, 2, 3, 4$. If $p_j \in B_{\geq i}$ then for some $d = 1, 2, 3, 4$, $|A_{j-1}^d \setminus A_j^d| \geq 8$. Moreover, the points of $A_{j-1}^d \setminus A_j^d$ will never again become d-maximal.*

Proof. Since $p_j \in B_{\geq i}$, p_j i-sees at least 37 points p_ℓ, $\ell < j$. That is, for each such point p_ℓ, there exists a unit disk D_ℓ containing only p_j and p_ℓ, among all points $p_1, \ldots, p_j$. For $d = 1, 2, 3, 4$, let H^d denote the subset of points p_ℓ for which the disk D_ℓ is in $\mathcal{D}^d$. Clearly, for at least one $d \in \{1, 2, 3, 4\}$, $|H^d| \geq 10$. Without loss of generality, assume that $|H^1| \geq 10$. It also follows by definition that the points in H^1 are 1-maximal in $\{p_1, \ldots, p_{j-1}\}$.

Let $m := |H^1|$ and let us also denote the points in H^1 by $q_1, \ldots, q_m$. Let $D_i \in \mathcal{D}^1$ be the unit disk that contains p_j and q_i, and let γ_i denote the circle bounding D_i, for $i = 1, \ldots, m$.

Consider the situation in polar coordinates about the center $o = o^d$. Let $\theta_1 < \theta_2$ be the orientations of the two rays bounding the cone K^d defined in Lemma 4.1. We regard each γ_i as

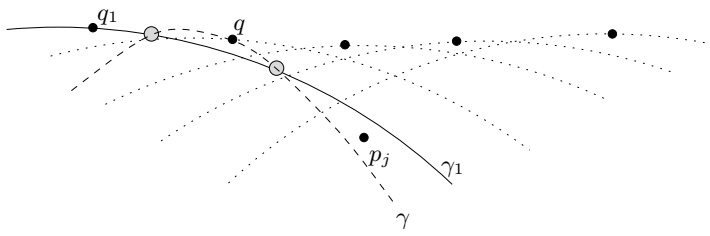


Figure 5: Illustrating the proof of Lemma 4.2. After inserting p_j every maximal point that it i-sees, except for the first and last in the θ-order, stops being maximal.

the graph of a function $\rho = \gamma_i(\theta)$, for $\theta_1 \leq \theta \leq \theta_2$. By Lemma 4.1, these graphs form a collection of θ-monotone *pseudolines*. By construction, p_j lies below (in the ρ-direction) all the graphs γ_i. Furthermore we can choose the disks D_i so that each point q_i lies on γ_i and above all the other graphs γ_j. That is, p_j lies below the lower envelope of the γ_i's, and the points q_i lie on the upper envelope of these arcs.

Without loss of generality, assume that the clockwise order (about o^d) of the points q_i along the envelope is $q_1, \ldots, q_m$. Let r be the index for which the θ-coordinate of p lies between those of q_r and q_{r+1}. We claim that all the points of H^1, except possibly for q_1 and q_m, are not in A_j^1.

Suppose, contrary to what the claim asserts, that there exists a unit disk $D \in \mathcal{D}^1$ that contains $q = q_\ell$ for some $1 < \ell < m$ but does not contain any other point of $\{p_1, \ldots, p_j\}$. Assume also that $\ell \leq r$, and let γ be the boundary of D.

The arc γ then passes below q_1, above q, and below p_j. On the other hand, the arc γ_1 passes above q_1, below q, and above p_j. Since q_1, q, p_j appear in this clockwise order about o^d, γ and γ_1 must intersect twice in K^1, contradicting the pseudoline property of these arcs. See Figure 5. The case where $j \geq r+1$ is treated similarly, with q_m playing the role of q_1.

The second assertion is obvious. □

Each point in $C_{\geq i}$ can leave A_j^d at most once, for each $d = 1, 2, 3, 4$. Therefore Lemma 4.2 implies that $|B_{\geq i}| \leq 4|C_{\geq i}|/8 = |C_{\geq i}|/2$. From here on, the proof continues exactly as in Section 3, leading to the following theorem.

Theorem 4.3. *The CF coloring algorithm of points for congruent disks presented in this section uses $O(\log n)$ colors with high probability. More precisely, it uses at most $c \log n$ colors with probability at least $1 - \frac{1}{n^{\alpha c}}$, where α is some absolute constant.*

Efficient implementation. When a new point p arrives, we need to determine efficiently if it is eligible for color m; that is, whether it sees m at level m. Recall that here p sees m at level m if and only if there is a disk of radius 1 containing p, another point, say x, of color m, and no other point of color $\geq m$.

A disk D of radius 1 whose center is in Q contains all points in Q. Such a disk could be an evidence that p sees m at level m if and only if there is only one point of level $\geq m$ and this point is actually of level m. We can easily detemine whether this is the case by maintaining the number of points of color $\geq m$ for every m.

We partition the disks whose center is outside Q into four groups (See Figure 3).

1. Disks D such that the y-coordinate of their center is larger than the y-coordinate of the upper edge of Q.

2. Disks D such that the y-coordinate of their center is smaller than the y-coordinate of the bottom edge of Q.

3. Disks D such that the y-coordinate of their center is between the y-coordinate of the bottom edge of Q and the y-coordinate of the top edge of Q, and the x-coordinate of their center is larger than the x-coordinate of the right edge of Q.

4. Disks D such that the y-coordinate of their center is between the y-coordinate of the bottom edge of Q and the y-coordinate of the top edge of Q, and the x-coordinate of their center is smaller than the x-coordinate of the left edge of Q.

Consider disks D of the first group. Disks of the other groups are treated analogously. Clearly, $D \cap Q$ is contained in the lower half of D. So for such disks it is enough to check whether their lower half contains p, another point, say x, of color m, and no other point of color $\geq m$.

To check this we switch to the dual plane, where each point x inserted so far (including p itself) is mapped to the upper half of a disk D_x of radius 1 centered at x. We associate with D_x the color of x. Let $\mathcal{D}_{\geq m}$ denote the set of all half disks inserted so far, whose color is at least m. Let $\mathcal{A}_{\geq m}$ denote the arrangement of these disks. We want to decide whether there exists a point $h \in Q$ which is contained in exactly two half disks, one of which is the disk D_p corresponding to the new point p, and another disk D_x of a point x colored m.

We perform this test analogously to our implementation of the algorithm of Section 2. The semicircles bounding the upper halves of the disks in $\mathcal{D}_{\geq m}$ form a collection of pseudo-segments (that is, each pair intersect at most once), which can easily be extended to pseudo-lines. We maintain the upper envelope of the semicircles bounding the upper half-disks, and the pockets associated with each semicircle that appears on the envelope, as balanced search trees. The algorithm uses these search trees is a manner analogous to the algorithm of Section 2, where we define a semicircle γ_1 to have "slope" larger than the slope of a semicircle γ_2, if γ_1 and γ_2 intersect, and γ_1 is above γ_2 to the left of their intersection.

It is easy to verify that the algorithm, appropriately modified, takes $O(\log^2 n)$ time with high probability for each inserted point, as above.

5 Online CF Coloring for Nearly Equal Axis-parallel Rectangles

A (possibly infinite) family $\mathcal{F}$ of axis-parallel rectangles is a *family of nearly-equal* axis-parallel rectangles, if there exists some positive constant α, such that the ratio between the largest width and the smallest width of the rectangles of $\mathcal{F}$, and the ratio between the largest height and smallest height of the rectangles of $\mathcal{F}$, are both at most α.

Consider a family $\mathcal{F}$ of nearly-equal axis-parallel rectangles. By scaling the coordinate axes, we may assume that the width and the height of any rectangle in $\mathcal{F}$ lie in $[1, \alpha]$. We tile the plane with an axis-parallel square grid whose cells have side length $1/2$. This ensures that both the width and the height of any rectangle in $\mathcal{F}$ are larger than the side length of a square tile of the grid. We assign to each grid tile a color class, so that no rectangle in $\mathcal{F}$ intersects two distinct grid tiles with the same color class. As in the case of unit disks, it is easy to verify that a constant number (indeed, $O(\alpha^2)$) of color classes suffices. We assign colors to points within each grid tile independently, using the colors of the class assigned to the tile. Let Q be an arbitrary square tile. By the discussion above, we can assume (without loss of generality) that all the points are inserted into the interior of Q.

The algorithm for online CF coloring of the points within Q is the same as the algorithm of Section 4. Here we say that p *i-sees* a point x (alternatively, p i-sees the color $c(x)$) if there is a rectangle in $\mathcal{F}$ that contains x and p and no point of color higher than $c(x)$.

The analysis is analogous to the analysis of Section 4. Here the corners of Q, denoted by o^1, o^2, o^3, and o^4, play the same role in the analysis as o^1, o^2, o^3, and o^4 in the previous section. That is, each rectangle in $\mathcal{F}$ that intersects Q contains at least one corner of Q, as is easily checked, and we arbitrarily associate it with one of the corners that it contains. Let $\mathcal{F}^i$ be the set of rectangles associated with o^i, for $i = 1, \ldots, 4$. The rest of the proof is similar to the one in Section 4, and is based on the easy observation that the boundaries of the rectangles in each fixed subfamily $\mathcal{F}^i$ behave as pseudolines within Q. Hence we have the following result.

Theorem 5.1. *The coloring algorithm always produces a conflict-free coloring, and the number of colors that it uses is $O(\log n)$, with high probability.*

Remark: If the rectangles in $\mathcal{F}$ are not nearly equal then, even in the static case, the number of colors required by the best known CF coloring algorithm is close to $n^{0.382+\varepsilon}$ [1] (see also [12, 2, 13]). The intuitive reason that we can extend our approach and improve this bound for nearly-equal rectangles is the fact that if R and R' are two nearly equal rectangles whose boundaries intersect, then any pair of boundary intersection points lie "far apart" from each other, unless R and R' slightly overlap each other near a vertex of each (and this latter case is bypassed by the analysis, as then R and R' are placed in different subfamilies $\mathcal{F}^i$). In contrast, two nearly equal (but not congruent) disks can almost overlap one another and yet the two intersections of their boundaries can be arbitrarily close to each other.

In other words, for our algorithm to work, it is crucial that the boundaries of the ranges behave like *pseudolines*. For halfplanes this holds trivially, whereas for congruent disks and nearly equal axis-parallel rectangles the property is enforced by tiling the plane, focusing on a single tile, and partitioning the ranges into subfamilies.

Efficient implementation. First we observe that p sees a color m at level m if and only if there is a square of side length α containing p, another point, say x, of color m, and no other point of color $\geq m$. This holds since every rectangle of height and width $\in [1, \alpha]$ intersecting Q has at most one corner in Q, so we can extend it to a square of side length α without changing its intersection with Q.

We split the squares whose center is outside Q into four groups as in Section 4. To handle, say, the first group we dualize each point x into an upper half of a square of width α centered at x, and proceed analogously to Section 4. Again, the resulting algorithm takes $O(\log^2 n)$ time with high probability for each newly inserted point.

6 Deterministic Online CF Coloring for Nearly-equal Axis-parallel Rectangles

In this section, we present a *deterministic* online algorithm for online CF coloring a sequence P of points in the plane for a family $\mathcal{F}$ of nearly equal axis-parallel rectangles, which uses $O(\log^3 n)$ colors. As discussed in Section 5, we can assume that the points of P all lie in a fixed square Q, whose side length is smaller than the width and the height of any rectangle of $\mathcal{F}$.

With each point $p \in P$, we associate the four quadrants delimited by the horizontal and vertical lines passing through p; we denote by NE_p, NW_p, SE_p, SW_p the northeastern, northwestern,

southeastern, and southwestern quadrants, respectively. We regard these quadrants as open; since we assume general position, no point, other than p, lies on the boundary of any of these quadrants.

By the time p is inserted, some of its quadrants may be empty (of points of the current prefix of P), and we classify p according to which of its quadrants are empty. A coarse classification of this sort is as follows:

- All four quadrants are empty. This can happen only for the first inserted point.
- Three of the quadrants are empty. There are four sub-classes of this kind. For example, if the empty quadrants are NW_p, NE_p, SE_p, we refer to p as *NE-extreme*. The other three sub-classes, of *NW-extreme* points, *SE-extreme* points, and *SW-extreme* points, are defined analogously.
- Two opposite quadrants are empty. There are two sub-classes of this kind, the *inclining backbone points*, for which NW_p and SE_p are empty, and the *declining backbone points*, for which NE_p and SW_p are empty.
- Two adjacent quadrants are empty. There are four sub-classes of this kind, the *highest, lowest, rightmost, and leftmost points* (at the time of their insertion).
- Only one quadrant is empty. There are four sub-classes of this kind, the *NE-maximal points*, for which NE_p is empty, and the analogously defined *NW-maximal points*, *SE-maximal points*, and *SW-maximal points*.
- None of the quadrants is empty. We call p an *interior point*.

See Figure 6.

We color each of these 16 classes using a different set of colors, using only $O(\log^3 n)$ colors in each class.

The structure of each class. We ignore the first point and the interior points (at the time of insertion); see below for details.

The other classes have certain monotone structure. The NE-extreme points, for example, form a single monotone increasing sequence, and each newly inserted NE-extreme point is added at the top-right end of that sequence. Moreover, any rectangle $R \in \mathcal{F}$ intersects this sequence in a *contiguous* subsequence (which, in case R contains the top-right or the bottom-left vertex of Q, is a suffix or a prefix, respectively). Similar properties (with the respective sequences being either monotone increasing or monotone decreasing) hold for NW-extreme, SE-extreme, and SW-extreme points.

The inclining backbone points also form a single monotone increasing sequence, but new inclining backbone points can be inserted anywhere in the sequence. A similar property holds for the declining backbone points (the sequence is monotone decreasing). In both cases, a rectangle $R \in \mathcal{F}$ intersects any of these sequences in a contiguous subsequence.

The sequence of highest points, sorted in increasing y-order, has the property that a newly inserted highest point is inserted at its end. This case, however, is more involved than the preceding cases, because, for any rectangle range $R \in \mathcal{F}$, the intersection of R with that sequence can consist of many pairwise disjoint contiguous subsequences, so we cannot regard R as inducing a single interval of that sequence. We will treat the highest points, and, symmetrically, the lowest, leftmost, and rightmost points in a different manner; see below.

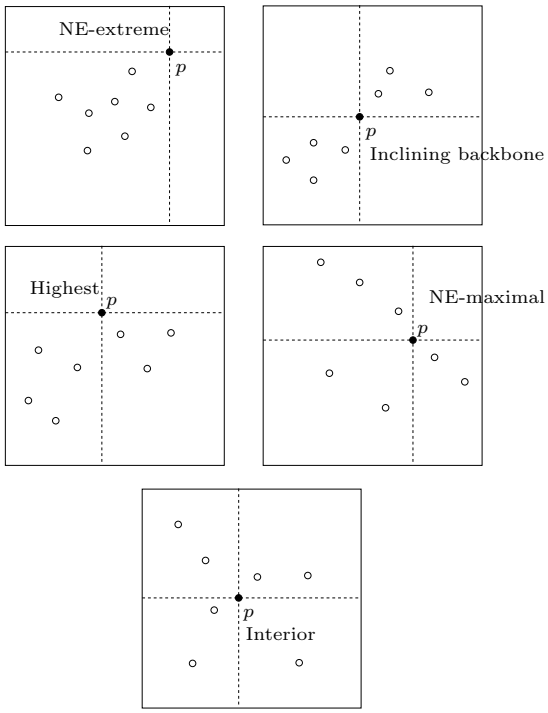


Figure 6: The classification of a newly inserted point $p \in P$.

Finally, the monotonicity structure of the NE-maximal points, say, is also involved. The points that were NE-maximal at the time of insertion and are still NE-maximal at some later given time, form a single monotone decreasing sequence. However, when a new NE-maximal point is inserted, it may replace a (contiguous) subsequence of NE-maximal points, making them all interior. See below for the handling of these points, as well as the NW-maximal, SE-maximal, and SW-maximal points.

The building blocks. We use the following two algorithms as building blocks. The first is an online algorithm for CF coloring of points on the line for interval ranges, but with the additional operation that allows to replace a consecutive sequence of points by a single new point. The second is an online algorithm for CF coloring of points on the line for interval ranges, for the special case where each point is inserted to the right of all the preceding points. We will consider two variants of this second algorithm, one in the normal setting defined above, and one which also supports the additional operation of replacing a *suffix* of the current sequence by the newly inserted point.

We note that while the problems are formulated for points on a line, they apply to any linearly ordered set of points, and we will indeed apply them to certain linearly ordered subsets of our planar point set.

Chen *et al.* [8] present a deterministic algorithm that CF colors points on the line for interval ranges with $O(\log^2 n)$ colors. The colors assigned by this algorithm are pairs (i, j) of integers, where i is the *level* of the color and j is a color assigned to points in that level. The colors are ordered lexicographically, and the maximum color in each interval at any stage is unique. We adapt this algorithm to support the operation of replacing a consecutive sequence σ of points with a single point p, by giving p the maximum color associated with a point in σ.

We claim that the modified algorithm maintains a CF coloring of the points at all times, and

still uses only $O(\log^2 n)$ colors. Indeed, the proof is by induction on the insertion order. Note first that new colors are created only when a point is inserted without replacing an existing subsequence. Consider an interval range I at some stage, and suppose to the contrary that I contains more than one point with maximum color c. Let p be the last point in I of this color to be inserted, and let p' be another point of color c in I, so that no point between p and p' has that color.

If p has replaced a subsequence σ then, at the moment just before p has been inserted, I, extended if necessary so as to contain all elements of σ, contains more than one point of color c (which is still maximal in (the extended) I): the point of σ from which p has inherited its color, and the point p' (which is clearly not in σ). This however contradicts the induction assumption, and thus rules out this case.

Suppose then that p has been inserted without replacing a subsequence. The coloring algorithm of Chen *et al.* [8] first assigns a *level* to p, skipping levels where p *sees* a point of that level (i.e., similar to the standard definition, no point of higher level lies in between p and the other point) both to its left and to its right, and then it colors p as a point of the *run* of its level (maximal contiguous subsequence not encompassing any point of higher level) that it joins, which it does as either the rightmost point or the leftmost point of the run. By assumption, p and p' have the same level. Since this level is maximal in I, they must be consecutive points in the same run. But then the coloring algorithm of Chen *et al.* [8] ensures that p is given, within this level, a color different from that of p', again a contradiction that establishes the claim.

The fact that the modified algorithm still uses only $O(\log^2 n)$ colors is argued as by Chen *et al.* [8], where it suffices to show that it produces only $O(\log n)$ levels. This is done by charging each newly inserted point p to the intervals of the runs of lower levels that it "destroys" (by being inserted into the interval); more precisely, p is charged to one endpoint of each interval, which is the endpoint that has "created" the interval when it was inserted (as an extreme point in the run, it creates only one interval). This recursive charging induces, for any point p of level i, a *binomial tree* of level i (and size 2^i) spanned by the points of P and rooted at p. This implies that the maximum level is at most $\log n$. The argument for the modified algorithm proceeds in the same way, with the twist that, when an inserted point p replaces a subsequence σ, it inherits the charges of the point $q \in \sigma$ of highest color. It is easily verified that a point p at level i is still the root of a binomial tree of level i, and the proof continues as above.

We refer to this modified algorithm as Algorithm **I1**.

A simple algorithm, also presented in Chen *et al.* [8], serves our second purpose: Consider first the case where no suffix replacement takes place. For $i \geq 1$, let $b(i)$ be the position of the rightmost '1'-bit in the binary representation of i. The algorithm colors the ith point with the number $b(i)$. For example, the first ten colors $b(1)$, ..., $b(10)$ are $1, 2, 1, 3, 1, 2, 1, 4, 1, 2$. As observed in Chen *et al.* [8], this is a valid CF coloring for intervals.

If suffix replacements are allowed, we use the following modification of the algorithm. First, if a newly inserted point p replaces a suffix σ, we give p the highest color of a point in σ. Second, when a point p is inserted without replacement, we give p the smallest color that it does not see (p sees a color c if no point after the last c-colored point has larger color); see [8] for more details. It is easily verified that the second rule produces exactly the coloring described in the preceding paragraph, when no suffix replacements take place. (It is this algorithm that the algorithm **I1**, or the replacement-free original algorithm of Chen *et al.* [8], uses to give colors to points within a fixed level.)

It is also easy to verify that the modified algorithm produces a valid CF-coloring, and that it uses only $O(\log n)$ colors. For more details, consult [8].

We refer to (both variants of) this algorithm as Algorithm **I2**.

The coloring algorithm. Let p be the next point to be inserted. If p is interior (relative to the prefix of P up to, and including p) then we give it a special color 0. If p is the first point to be inserted, we give it another special color $0'$. Otherwise, we use a separate set of colors for each of the other remaining classes; for the sake of convenience, we think of each of these color classes as the integers or, in case we employ the **I1** algorithm, as the lexicographically ordered set of pairs of integers.

Coloring NE-, NW-, SE-, and SW-extreme points. Each of these sets is colored using the **I2** algorithm without suffix replacement. The preceding discussion implies that the algorithm is indeed applicable in this case. Hence, these classes require only $O(\log n)$ colors.

Coloring backbone points. We color each of the two sub-classes of backbone points using the **I1** algorithm, with a total of $O(\log^2 n)$ colors. Again, the preceding discussion concerning the structure of backbone points justifies the use of this algorithm.

Coloring NE-, NW-, SE-, and SW-maximal points. We assign to each NE-maximal point two colors, which we denote as the *B-color* and the *G-color*. The B-color is obtained by applying Algorithm **I1** to the chain C_{NE} of current NE-maximal points, ordered by the increasing x-order (or decreasing y-order) of its points. When a newly inserted NE-maximal point p is added, it may eliminate a contiguous subchain $\sigma(p)$ of C_{NE} (whose points now become interior), in which case the algorithm assigns to p the highest B-color in $\sigma(p)$.

However, unlike the preceding classes, this coloring in itself need not be conflict-free, at least not in the strong sense of a unique maximum color. To see why, denote by S_{NE} the set of points that were NE-maximal at the time of insertion. A rectangle $R \in \mathcal{F}$ may intersect S_{NE} in a subset that contains both currently NE-maximal points and interior points (that were NE-maximal when inserted), and the overall maximum B-color in R need not be unique (it is guaranteed to be unique only among the currently NE-maximal points).

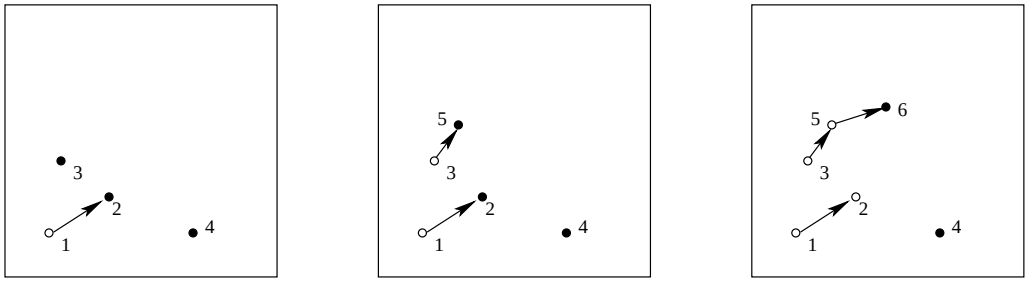
To overcome this difficulty, we use the second set of *G-colors*. To introduce them, we define a directed graph G on S_{NE}, each of whose edges connects a pair of points p, q, where p is the (unique) point of $\sigma(q)$ of the highest B-color (which is also the B-color of q). See Figure 7. It follows that G is a collection of vertex-disjoint paths, and that the B-colors of all the points on the same path are equal. Moreover, each path is a monotone increasing chain (in both the x- and y-coordinates); when a path is extended by a new point p, both its x- and y-coordinates are larger than those of the previous last point on the path.

We assign G-colors to the points on each path separately, using the same set of colors, by applying Algorithm **I2** without suffix replacement (the preceding argument implies that the algorithm is indeed applicable in this setup).

The final color assigned to a NE-maximal point p is the pair $(B\text{-color}(p), G\text{-color}(p))$. Hence, the number of colors used is $O(\log^3 n)$.

Symmetric procedures, with different sets of colors (both B-colors and G-colors) are applied to the NW-, SE-, and SW-maximal points. Hence, the coloring algorithm uses a total of $O(\log^3 n)$ colors for these classes.

Coloring highest, lowest, rightmost, and leftmost points. Consider the coloring of the highest points. We regard each highest point p as being both NE-maximal and NW-maximal, and



(a) (b) (c)

Figure 7: Illustrating the directed paths formed by the points of S_{NE}. The numbers beside the points show the order they are inserted. The points on the current NE-maximal chain are black. In (a), the B-colors of p_1, p_2, p_3, p_4 are $1, 1, 2, 3$, respectively. In (b), p_5 dominates p_3 and is assigned the B-color of p_3, which is 2. In (c), p_6 dominates p_2 and p_5, and is assigned the higher of the B-colors of p_2 and p_5, which is 2. The G-colors of the points p_3, p_5, p_6 are $1, 2, 1$, respectively.

apply the preceding procedure twice, to color p by the quadruple

$$\left(B^{(NE)}\text{-color}(p),\ G^{(NE)}\text{-color}(p),\ B^{(NW)}\text{-color}(p),\ G^{(NW)}\text{-color}(p)\right)$$

of the two resulting B-colors and the two resulting G-colors. We emphasize though that the coloring of the highest points is done *independently* (with *disjoint* sets of colors) of the coloring of the "standard" NE-maximal and NW-maximal points. The reason for regarding the highest points as being both NE-maximal and NW-maximal is to prepare for both situations in which a rectangle in $\mathcal{F}$ intersects the underlying square in a quadrant that contains its NE-corner, or in a quadrant that contains its NW-corner. In the former case we want to regard the highest points as being NE-maximal, and in the latter case, as being NW-maximal; see below for more details.

To argue that we do not use too many colors, we first note that a newly inserted highest point p is always inserted at the end of the current NE-maximal chain C'_{NE} of the highest points, and similarly for the current NW-maximal chain C'_{NW} of highest points. Hence, both B-colors can be assigned using the **I2** algorithm with suffix replacement, rather than the **I1** algorithm. Hence, the number of B-colors, of either kind, that the algorithm generates is only $O(\log n)$.

This still leaves us with a potential number of $O(\log^4 n)$ colors. However, we note that when a new highest point p is inserted, it will replace a suffix of *exactly one* of the chains C'_{NE}, C'_{NW} (and will be added, without replacement, at the end of the other chain). Hence, p will acquire exactly one G-link, either from a point in the previous NE-maximal chain of highest points that it has replaced, or from a point in the previous NW-maximal chain that it has replaced. Consequently, one of its G-colors will always be 1, as it will be the *first* point on the respective G-path. See Figure 8. Hence, the algorithm colors highest points (and, symmetrically, lowest, rightmost, and leftmost points) using only $O(\log^3 n)$ colors, which is thus a bound on the grand total number of colors that it uses.

Correctness. Let R be a rectangle in $\mathcal{F}$, and consider some stage during the online process. Without loss of generality, assume that R contains the top-right corner of Q. If R contains the first point of P then it has the unique color $0'$. Otherwise, it must contain some point p whose NE-quadrant is empty now, and thus was empty at the time of insertion. Thus p is (at the time of insertion) either an NE-extreme point, a declining backbone point, a highest point, a rightmost

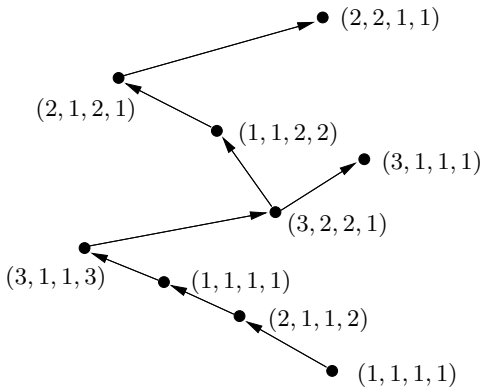


Figure 8: The G-links and the coloring of the highest points. The quadruple associated with a point p is $\left(B^{(NE)}\text{-color}(p),\ G^{(NE)}\text{-color}(p),\ B^{(NW)}\text{-color}(p),\ G^{(NW)}\text{-color}(p)\right)$.

point, or just a "plain" NE-maximal point. Thus R must intersect at least one of these classes of points, and we argue for each of these cases separately.

If R contains NE-extreme points, then R intersects the sequence of these points in a suffix, and the correctness follows from the correctness of the **I2** algorithm, which is applied to this sequence. Similarly, if R contains declining backbone points, the correctness follows from the correctness of the **I1** algorithm, which is applied to this sequence.

Suppose next that R contains NE-maximal points. We claim that the maximum color in $R \cap S_{NE}$, under the lexicographical order, is unique. Let $p \in R$ be a point having that color, which we write as $(b_{\max}, g_{\max})$.

We claim that, among the G-paths that intersect R, the G-path of p is the only G-path whose points have B-color $b_{\max}$. Indeed, assume to the contrary that there is another G-path of B-color $b_{\max}$ which intersects R. Then one of the following cases must arise.

1. Both G-paths have a representative in the current chain C_{NE} of NE-maximal points. These two representatives must be in R and since R intersects C_{NE} in a contiguous subsequence σ, by the properties of Algorithm **I1**, σ must contain a point with B-color larger than $b_{\max}$, which contradicts the choice of p.

2. One of the G-paths does not have a representative in the current chain C_{NE} of NE-maximal points. In this case the last point on that path is dominated by a point with B-color larger than $b_{\max}$, which must be in R. This again contradicts the choice of p.

Since R intersects the G-path of p in a suffix σ, it follows, by the properties of Algorithm **I2**, that $g_{\max}$ is the largest G-color of a point in σ, and occurs only once in σ. Hence, p is indeed the only point of B-color $b_{\max}$ and G-color $g_{\max}$ in R.

Finally, consider the case where R contains, say, highest points (the case of rightmost points is symmetric). For each highest point p, consider only the first two components

$$\left(B^{(NE)}\text{-color}(p),\ G^{(NE)}\text{-color}(p)\right)$$

of the color of p. Arguing exactly as in the case of NE-maximal points, we conclude that R has a highest point p with a unique "sub-color" of this form, so necessarily the "whole" color of p is also unique in R.

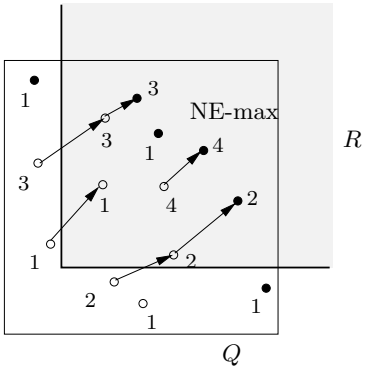


Figure 9: Illustrating the correctness of the CF coloring of NE-maximal points. The points on the current NE-maximal chain are black. The numbers beside the points are their B-colors.

This completes the proof of correctness of the algorithm, and allows us to conclude with the main result of this section:

Theorem 6.1. *One can* deterministically *online color a sequence of n points in the plane, such that the coloring is always conflict-free with respect to a family of nearly-equal axis-parallel rectangles. The algorithm uses $O(\log^3 n)$ colors.*

Efficient implementation. When a point p arrives we have to classify it into the appropriate one of the 16 classes defined at the beginning of this section. For that we maintain a planar data structure for orthant emptiness queries, implemented as a priority search tree or a range tree (see [5]). We perform four queries to this data structure with the orthants whose origin is at p, and classify p according to the answers to these queries. Then we insert p into the data structure.[3] With any of the standard data structures mentioned above both query and update take $O(\log n)$ time.

Each class of points is maintained in a separate linear conflict free data structure, by the algorithms I2 and I1. Thus, to complete the implementation we now describe an efficient implementation of I1; the implementation of I2 is similar and simpler.

For each j, let $P_{\geq j}$ be the set of points at level $\geq j$. We maintain $P_{\geq j}$ in a balanced search tree T_j (e.g., a red-black tree), where the points are ordered from left to right as they are ordered on the line. A point p is eligible for level j if either its predecessor or its successor in T_j is of level strictly greater than j, which we can decide in $O(\log n)$ time. Once we find the first level for which p is eligible, say m, we insert p into T_m and give it a color at level m.

To be able to assign colors at level m, we maintain each consecutive sequence (run) of points of level m in $P_{\geq m}$ in a cartesian tree, as defined in the implementation of the algorithm given in Section 2. When p is eligible for level m, we add it either as the leftmost point or as the rightmost point of a run of points of level m. In either case we find the color to assign to p using the cartesian tree representing this run, as described in Section 2. This takes $O(\log n)$ time. We can also insert p into the cartesian tree in $O(\log n)$ time.

Finally, we insert p into all the trees T_j, for $j < m$. If p is inserted in the middle of a run of consecutive points of level j in T_j, then p splits this run into two sub-runs. We split the cartesian tree representing this run into two cartesian trees, one for each of the two new runs. It is straightforward to check that we can implement each such split in $O(\log n)$ time.

[3] A range tree can support emptiness queries by storing at each primary node the minimum and maximum y-coordinate of a point in the corresponding range. Then it is straightforward to support insertions in $O(\log n)$ time.

Consider next the operation where we replace a consecutive sequence of points by the point of maximum color in the sequence. In this case, we have to update all trees of level smaller than the level of the surviving point. We update each of these trees by splitting out the relevant subsequence in $O(\log n)$ time. In each such tree we may also have to split two cartesian trees that represent runs that overlap the portion that we delete. This also takes $O(\log n)$ time.

Summing up, we can implement the algorithm in $O(\log^2 n)$ (deterministic) time per point. (If p is inserted at level m, we spend $O(\log n)$ time for each level smaller than m.) Since the number of points at level $\geq m$ is at most $n/2^m$, our implementation requires linear space.

7 Conclusions

In this paper, we presented randomized online CF coloring algorithms (against oblivious adversaries) for several range spaces in the plane, using $O(\log n)$ colors with high probability. We have also presented the first efficient *deterministic* algorithm for CF coloring points in the plane with respect to nearly-equal axis-parallel rectangles (excluding the weaker algorithm given in the preliminary version of this paper [7]); this algorithm works also against a non-oblivious adversary.

Interestingly, we were unable to extend the deterministic algorithm to other ranges (in particular, halfplanes, and congruent disks) in the plane, and we leave as open the problem of finding any deterministic algorithm for these ranges that uses only polylogarithmically many colors. Another open problem is to obtain randomized algorithms with comparable performances against non-oblivious adversaries. As noted, this is also a challenge for the simpler 1-dimensional variant studied by Chen *et al.* [8].

Acknowledgments

The authors thank Sariel Har-Peled for helpful discussions on the problems studied in the paper and useful comments on the manuscript. In particular, the simplified presentation in Section 6 was initially inspired by his suggestions. Thanks are also due to two annonymous referees for their useful comments. In particular, studying the efficiency of the coloring algorithm itself, which was not considered before, even for the case of points on the line [8], was motivated by a remark of one of the referees.

References

[1] D. Ajwani, K. Elbassioni, S. Govindarajan and S. Ray, Conflict-free colorings for rectangle ranges using $O(n^{.382+\varepsilon})$ colors, *Proc. 19th Annu. ACM Sympos. Parallelism Algorithms Architectures (SPAA'07)*, 2007, 181–187.

[2] N. Alon, M. Krivelevich and B. Sudakov, Coloring graphs with sparse neighborhoods, *J. Combinat. Theory, Ser. B* 77 (1999), 73–82.

[3] A. Bar-Noy, P. Cheilaris and S. Smorodinsky, Conflict-free coloring for intervals: from offline to online, *Proc. 18th Annu. ACM Sympos. Parallelism Algorithms Architectures (SPAA'06)*, 2006, 128–137.

[4] A. Bar-Noy, P. Cheilaris, S. Olonetsky and S. Smorodinsky, Online conflict-free coloring for hypergraphs, *Proc. 34th Internat. Colloq. on Automata, Languages and Programming (ICALP'07)*, 2007, 219–230.

[5] M. de Berg, M. van Kreveld, M. Overmars and O. Schwarzkopf, *Computational Geometry: Algorithms and Applications*, 2nd Edition, Springer Verlag, Heidelberg, 2000.

[6] A. Borodin and R. El-Yaniv, *Online Computation and Competitive Analysis*, Cambridge University Press, New York, 1998.

[7] K. Chen, How to play a coloring game against a color-blind adversary, *Proc. 22nd Annu. ACM Sympos. Comput. Geom. (SCG'06)*, 2006, 44–51.

[8] K. Chen, A. Fiat, H. Kaplan, M. Levy, J. Matoušek, E. Mossel, J. Pach, M. Sharir, S. Smorodinsky, U. Wagner and E. Welzl, Online conflict-free coloring for intervals, *SIAM J. Comput.* 36 (2006), 1342–1359.

[9] X. Chen, J. Pach, M. Szegedy and G. Tardos, Delaunay graphs of points sets in the plane with respect to axis-parallel rectangles, *Proc. 19th ACM-SIAM Sympos. Discrete Algo. (SODA'08)*, 2008, 94–101.

[10] K. Elbassioni and N. Mustafa, Conflict-free colorings of rectangle ranges, *Proc. 23rd Internat. Sympos. Theoret. Aspects Comput. Sci. (STACS'06)*, 2006, 254–263.

[11] G. Even, Z. Lotker, D. Ron and S. Smorodinsky, Conflict-free colorings of simple geometric regions with applications to frequency assignment in cellular networks, *SIAM J. Comput.* 33 (2004), 94–136.

[12] S. Har-Peled and S. Smorodinsky, On conflict-free coloring of points and simple regions in the plane, *Discrete Comput. Geom.* 34 (2005), 47–70.

[13] J. Pach and G. Tóth, Conflict-free colorings, in *Discrete and Computational Geometry, The Goodman-Pollack Festschrift*, (B. Aronov, S. Basu, J. Pach and M. Sharir, editors), Springer Verlag, Heidelberg, 2003, 665–671.

[14] S. Smorodinsky, *Combinatorial Problems in Computational Geometry*, Ph.D. Thesis, School of Computer Science, Tel-Aviv University, 2003.

[15] S. Smorodinsky, On the chromatic number of some geometric hypergraphs, *SIAM J. Discrete Math.* 21 (2007), 676–687.

[16] J. Vuillemin, A unifying look at data structures, *Comm. ACM* 23 (1980), 229–239.