Online Conflict-Free Coloring for Intervals[1]

Ke Chen[2] Amos Fiat[3] Haim Kaplan[4] Meital Levy[5] Jiří Matoušek[6]
Elchanan Mossel[7] János Pach[8] Micha Sharir[9] Shakhar Smorodinsky[10]
Uli Wagner[11] Emo Welzl[12]

December 19, 2006

Abstract

We consider an online version of the conflict-free coloring of a set of points on the line, where each newly inserted point must be assigned a color upon insertion, and at all times the coloring has to be *conflict-free*, in the sense that in every interval I there is a color that appears exactly once in I. We present deterministic and randomized algorithms for achieving this goal, and analyze their performance, that is, the maximum number of colors that they need to use, as a function of the number n of inserted points. We first show that a natural and simple (deterministic) approach may perform rather poorly, requiring $\Omega(\sqrt{n})$ colors in the worst case. We then derive two efficient variants of this simple algorithm. The first is deterministic and uses $O(\log^2 n)$ colors, and the second is randomized and uses $O(\log n)$ colors with high probability. We also show that the $O(\log^2 n)$ bound on the number of colors used by our deterministic algorithm is tight on the worst case.

We also analyze the performance of the simplest proposed algorithm when the points are inserted in a random order, and present an incomplete analysis that indicates that, with high probability, it uses only $O(\log n)$ colors. Finally, we show that in the extension of this problem to two dimensions, where the relevant ranges are disks, n colors may be required in the worst case.

[1]Work by Ke Chen was partially supported by a NSF award CCR-0132901. Work by Haim Kaplan was partially supported by German Israeli Foundation (GIF) Grant no. 2051-1156-6/2002. Work by Elchanan Mossel was supported by a Miller fellowship in Statistics and Computer Science, U.C. Berkeley. Work by Micha Sharir was supported by a grant from the U.S.-Israeli Binational Science Foundation, by a grant from the Israel Science Fund (for a Center of Excellence in Geometric Computing), by NSF Grants CCR-97-32101 and CCR-00-98246, and by the Hermann Minkowski–MINERVA Center for Geometry at Tel Aviv University. Part of the work was carried out during a visit to Charles University, which was supported by COMBSTRU. Work by Uli Wagner was supported by COMBSTRU. Part of the work on the paper has been carried out at MSRI, Berkeley, when several of the authors have visited this institute during the Fall of 2003. This paper combines and extends results from [6, 12].

[2]Department of Computer Science, University of Illinois; 201 N. Goodwin Ave.; Urbana, IL 61801; USA; kechen@uiuc.edu

[3]School of Computer Science, Tel Aviv University, Tel Aviv, Israel; fiat@post.tau.ac.il

[4]School of Computer Science, Tel Aviv University, Tel Aviv, Israel; haimk@post.tau.ac.il

[5]School of Computer Science, Tel Aviv University, Tel Aviv, Israel; levymeit@post.tau.ac.il

[6]Department of Applied Mathematics and Institute for Theoretical Computer Science (ITI), Charles University, Prague, Czech Republic; matousek@kam.mff.cuni.cz

[7]Department of Statistics, University of California at Berkeley, Berkeley, CA, USA; mossel@stat.berkeley.edu

[8]Courant Institute of Mathematical Sciences, New York University, New York, NY, USA; pach@cims.nyu.edu

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

[10]Institute for Theoretical Computer Science, ETH Zürich, Switzerland; sshakhar@inf.ethz.ch

[11]Department of Applied Mathematics, Charles University, Prague; uli@kam.mff.cuni.cz

[12]Institute for Theoretical Computer Science, ETH Zürich, Switzerland; emo@inf.ethz.ch

1 Introduction

Let P be a set of n points in $\mathbb{R}^d$ and $\mathcal{R}$ a set of subsets of $\mathbb{R}^d$, called *ranges* (e.g., the set of all disks in the plane). A coloring of P is called *conflict-free* (CF for short) with respect to $\mathcal{R}$ if for each $r \in \mathcal{R}$ with $P \cap r \neq \emptyset$, there is at least one color that appears exactly once in r.

We consider the following dynamic scenario of conflict-free coloring of points on the line, with respect to interval ranges. We maintain a finite set $P \subset \mathbb{R}$. Initially, P is empty, and we repeatedly insert points into P, one point at a time. We denote by $P(t)$ the set P after the t-th point has been inserted. Each time we insert a point p, we need 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, as in the static case, for any interval I that contains points of $P(t)$, there is a color that appears exactly once in I.

The static version of CF-coloring has been studied recently in several papers [10, 11, 13, 15, 16] in considerably more general settings, involving point sets in higher dimensions, and ranges that are disks, balls, axis-parallel boxes, or more general ranges that satisfy certain geometric conditions. The study of this problem is 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*. The base stations are interconnected by an external fixed backbone network. 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, on the other hand, continuously scan frequencies in search of a base station with good reception. The fundamental problem of frequency-assignment in cellular networks is to assign frequencies to base stations so that every client, located within the receiving range of at least one station, can be served by some base station, in the sense that the client is located within the range of the station and no other station within its reception range has the same frequency (such a station would be in "conflict" with the given station due to mutual interference). The goal is to 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 *antennae*. 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 (base station) and its radius of transmission is fixed and is given (the transmission radii of the antennae are not necessarily equal). Even et al. [11] have shown that one can find an assignment of frequencies to the antennae 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 [11, 14, 15]. When the given antennae all have the same radius of transmission (say, unit radius), the problem is easily seen to be equivalent to that of coloring n points in the plane such that for any unit radius disk that contains more than one of the given points, at least one of the colors in that disk is unique. This is the scenario whose online version is studied in this paper. We do not address the dual version, in which the goal is to color n given ranges so that, for each point p that lies in their union, there is a color that appears exactly once among the ranges that contain p. See [11, 13, 15] for many variants of both (static) versions of the problem.

To capture a dynamic scenario where antennae can be added to the network, we introduce and study an online version of the CF coloring problem, as described above. As we show in this paper, the online version of the problem is considerably harder, even in the one-dimensional case, where the static version is trivial and fully understood. We begin by proposing a natural, simple, and obvious coloring algorithm (to which we refer as the UniMax greedy algorithm), but show that in the worst case it has poor performance. Specifically, the UniMax greedy algorithm may require $\Omega(\sqrt{n})$ colors in the worst case. We still do not have any nontrivial (i.e., sublinear) upper bound on the performance of the algorithm.

The UniMax greedy algorithm is indeed greedy in nature, but there are several different greedy approachs, and we briefly discuss another greedy alternative, about which almost nothing is known.

We next remedy the situation, by presenting two more efficient algorithms. We describe a 2-stage deterministic variant of the UniMax greedy algorithm, and show that the maximum number of colors that it uses is $\Theta(\log^2 n)$. We also describe a randomized version of the UniMax greedy algorithm, which uses, with high probability,[13] only $O(\log n)$ colors.

The best known general lower bound for this problem is $\Omega(\log n)$, which holds also for the static case (see [11, 14, 15]), so there still remains a gap between the upper and lower bounds in the deterministic case.

Our randomized algorithm works against an oblivious adversary. That is, we assume that the sequence of points is fixed by the adversary before starting feeding them one by one to the online algorithm. The adversary cannot choose its next point based on the actions or the random choices that the online algorithm has made so far. The coloring that our algorithm produces is conflict free no matter which random choices it makes. For further discussion on different kinds of adversaries in online computations, see [5] for the general case, and [3] for the specific case of online CF-coloring.

Next, we return to the UniMax greedy algorithm, which can be inefficient in the worst case, and analyze its performance when the points are inserted in a *random order*. We reduce the problem to a certain stationary stochastic process, and present partial analysis of its performance, as well as a fairly reasonable set of conjectures, strongly supported by simulations, that indicate that the expected number of colors that the simple algorithm uses in this case is only $O(\log n)$.

Finally, we consider the extension of the online version to point sets in the plane. Unfortunately, we show that, in the simple case where the ranges that are required to be conflict-free are disks (or arbitrary radii), n colors may be needed in the worst case. Nevertheless, (much) better solutions might still exist for random distributions of the points, for other ranges, or for relaxed versions of the problem, in which each range has a color that appears in it at least once and at most k times, for some constant k [15]. A recent follow-up study by Chen, Kaplan and Sharir [7] (see also [6]) gives randomized online CF coloring algorithms for points in the plane, with respect to halfplanes, unit disks, or nearly equal axis-parallel rectangles. The algorithms use $O(\log n)$ colors, with high probability. An even more recent result of Bar Noy et al. [4] provides a general randomized online CF-coloring algorithm, which achieves the same performance as [7] in the special cases just mentioned.

There are many open problems that our study raises: Obtain, if possible, an improved algorithm-independent deterministic lower bound for online CF coloring for intervals; get a better understanding of the problem behavior in the plane and in higher dimensions; design and analyze other strategies, and so on (see additional problems posted later throughout the paper). We note that CF coloring is closely related to the problem of *vertex ranking* in graphs (see, e.g., [9]). Some of our algorithms, that maintain the property that the *maximum* color in any interval is unique, actually perform online vertex ranking in *paths*. Extending our analysis to online vertex ranking in other

[13]This means that the probability of failure is at most $1/p(n)$, where $p(n)$ is polynomial in n, whose degree can be made arbitrarily large by adjusting the constants of proportionality in the performance bound.

kinds of graphs (trees, for example) raises yet another set of interesting open problems.

2 The UniMax Greedy Coloring Algorithm

Instead of the usual conflict-free property, we wish to maintain the following stronger *Unique Maximum Invariant* (in which we assume that the colors are positive integers):

> *At any given step t and for any interval I that contain points of $P(t)$, there is only one element of $P(t) \cap I$ that attains the* maximum *color in that set.*

This invariant implies that the coloring of $P(t)$ is conflict-free, at any time t. It is indeed a stronger condition: Conflict-free coloring only requires that for each interval there exists a color (not necessarily the maximum) that appears there only once.

We employ the following simple-minded algorithm for inserting a point p into the current set $P(t)$. In a nutshell, the rule is simply to assign to p the *smallest* possible color that maintains the invariant. This rule is implemented as follows. We say that the newly inserted point p *sees* a point x if all the colors of the points between p and x (exclusive) are smaller than $c(x)$. In this case we also say that p sees the color $c(x)$. We then give p the smallest color that it does not see. (Note that a color can be seen from p either to the left or to the right, but not in both directions; see below.) We refer to this algorithm as the *Unique Maximum Greedy* algorithm, or the UniMax greedy algorithm, in short.

Below is an illustration of the coloring rule of the UniMax greedy algorithm. The left column gives the colors (integers in the range $1, 2, \dots, 6$) assigned to the points in the current set P and the location of the next point to be inserted (indicated by a period). The right column gives the colors "seen" by the new point. The colors seen to the left precede the $\cdot$ and those seen to the right succeed the period.

$1\cdot$	$[1\cdot]$
$1 \cdot 2$	$[1 \cdot 2]$
$1 \cdot 32$	$[1 \cdot 3]$
$12 \cdot 32$	$[2 \cdot 3]$
$121 \cdot 32$	$[21 \cdot 3]$
$121 \cdot 432$	$[21 \cdot 4]$
$121 \cdot 3432$	$[21 \cdot 34]$
$1215 \cdot 3432$	$[5 \cdot 34]$
$1215 \cdot 13432$	$[5 \cdot 134]$
$12152 \cdot 13432$	$[52 \cdot 134]$
$121526 \cdot 13432$	$[6 \cdot 134]$

Correctness. The correctness of the algorithm is established by induction on the insertion order. First, note that no color can be seen twice from p: This is obvious for two points that lie both to the left or both to the right of p. If p sees the same color at a point u to its left and at a point v to its right then the interval $[u, v]$, before p is inserted, does not have a unique maximum color, so this case is impossible too. Next, if p is assigned color c, any interval that contains p still has a unique maximum color: This follows by induction when the maximum color is greater than c. If the maximum color is c then it cannot be shared by another point u in the interval, because then p would have seen the nearest such point, and thus would not be assigned color c. It is also easy to see that the algorithm assigns to each newly inserted point the smallest possible color that maintains the invariant of a unique maximum color in each interval. This makes the algorithm *greedy* with respect to the unique maximum condition.

Special insertion orders. Denote by $C(P(t))$ the sequence of colors assigned to the points of $P(t)$, in left-to-right order along the line. Let $c_{\max}(P(t))$ denote the maximum color in $C(P(t))$.

The *complete binary tree sequence* S_k of order k is defined recursively as $S_1 = (1)$ and $S_k = S_{k-1}\|(k)\|S_{k-1}$, for $k > 1$, where $\|$ denotes concatenation. Clearly, $|S_k| = 2^k - 1$.

For each pair of integers $a < b$, denote by $C_0(a, b)$ the following special sequence. Let k be the integer satisfying $2^{k-1} \le b < 2^k$. Then $C_0(a, b)$ is the subsequence of S_k from the a-th place to the b-th place (inclusive). For example, $C_0(5, 12)$ is the subsequence $(1, 2, 1, 4, 1, 2, 1, 3)$ of $(1, 2, 1, 3, 1, 2, 1, 4, 1, 2, 1, 3, 1, 2, 1)$.

Lemma 2.1. *(a) If each point is inserted into P to the right of all preceding points, then $C(P(t)) = C_0(1, t)$.*
(b) If each point is inserted into P to the left of all preceding points, then $C(P(t)) = C_0(2^k - t, 2^k - 1)$, where k satisfies $2^{k-1} \le t < 2^k$.
(c) If each point is inserted into P either from the left or from the right then $C(P(t))$ is some subsequence of the form $C_0(a, b)$, where $b \le |P(t)|$.

Proof: Easy, and omitted. $\Box$

2.1 Lower bound for the UniMax greedy algorithm

Theorem 2.2. *The UniMax greedy algorithm may require $\Omega(\sqrt{n})$ colors in the worst case for a set of n points.*

Proof: For each integer k, define the sequence

$$C_k = (1, 2, 1, 3, 2, 1, \ldots, k-1, k-2, \ldots, 1, k, k-1, \ldots, 1).$$

Note that C_k is the concatenation of k sequences $D_1\|D_2\| \cdots \|D_k$, where $D_j = (j, j-1, \ldots, 2, 1)$. Put $n_k = k(k+1)/2$. We prove the following property, from which the assertion of the theorem is an immediate corollary.

$(*)$ There exists an insertion order of n_k points for which the color sequence produced by the UniMax greedy algorithm is C_k.

The proof proceeds by induction on k. We note that the claim easily holds for $k = 1, 2$. Suppose that there is an insertion sequence S_k for which the UniMax greedy algorithm produces the color sequence C_k. We insert the next point in between D_{k-1} and D_k, and observe that it is assigned color $k+1$. We then insert a point between D_{k-2} and D_{k-1}, which is assigned color k. Proceeding in this manner from right to left, we insert k points between consecutive subsequences D_{j-1}, D_j. The color sequence now becomes

$$D_2\|D_3\|D_4\| \cdots \|D_k\|D_{k+1}.$$

To complete the step, we insert one additional point to the left of the whole sequence, which gets the color 1, thereby producing the color sequence C_{k+1}. This completes the proof of $(*)$, and thus of the theorem. $\square$

Open problem: Obtain an upper bound for the maximum number of colors that the algorithm uses for n inserted points. We conjecture that the bound is close to the $\Omega(\sqrt{n})$ lower bound. At the moment, we do not have any sublinear upper bound.

2.2 Related algorithms

The First-Fit algorithm—another greedy strategy. The UniMax greedy algorithm is greedy for maintaining the unique-maximum invariant, namely, that in each interval the *maximum color* appears exactly once. Perhaps it is more natural to consider a greedy approach in which we only want to enforce the standard CF property. That is, we want to assign to each newly inserted point the *smallest* color for which the CF property continues to hold. There are cases where this *First-Fit* greedy algorithm uses fewer colors than the UniMax greedy algorithm: Consider an insertion of five points in the order (1 3 2 4 5). The UniMax greedy algorithm produces the color sequence (1 3 2 1 4), whereas the First-Fit algorithm produces the coloring (1 3 2 1 2).

Very recently, after the original submission of this paper, Bar Noy et al. [3] have shown that in the worst case the First-Fit algorithm uses about $n/2$ colors. More precisely, there are sequences with $2i+3$ elements that force the algorithm to use $i+3$ colors, and this bound is tight.

CF coloring for unit intervals. Consider the special case where we want the CF property to hold only for *unit intervals*. In this case, $O(\log n)$ colors suffice: Partition the line into the unit intervals $J_i = [i, i+1)$, for $i \in \mathbb{Z}$. Color the intervals J_i with even i as white, and those with odd i as black. Note that any unit interval meets only one white and one black interval. We color the points in each J_i independently, using the same set of "light colors" for each white interval, and the same set of "dark colors" for each black interval. For each J_i, we color the points that it contains using the UniMax greedy algorithm, except that new points inserted into J_i in between two previously inserted points get a special color, color 0. It is easily checked that the resulting coloring is CF with respect to unit intervals. Since we effectively only insert points into any J_i to the left or to the right of the previously inserted points, Lemma 2.1(c) implies that the algorithm uses only $O(\log n)$ (light and dark) colors. We remark that this algorithm satisfies the unique maximum color property for unit-length intervals.

We note that, in constrast with the static case (which can always be solved with $O(1)$ colors), $\Omega(\log n)$ colors may be needed in the worst case. Indeed, consider a left-to-right insertion of n points into a sufficiently small interval. Each contiguous subsequence σ of the points will be a suffix of the whole sequence at the time the rightmost element of σ is inserted. Since such a suffix can be cut off the current set by a unit interval, it must have a unique color. Hence, at the end of insertion, *every* subsequence must have a unique color, which implies (see [11, 15]) that $\Omega(\log n)$ colors are needed.

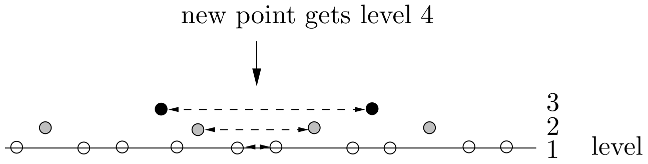

Figure 1: Illustrating the 2-stage deterministic algorithm. An insertion order that realizes the depicted assignment of levels to points is to first insert all level-1 points from left to right, then insert the level-2 points from left to right, and then the level-3 points.

3 An Efficient Deterministic Algorithm

In this section we modify the UniMax greedy algorithm into a deterministic 2-stage coloring scheme, and show that it uses only $O(\log^2 n)$ colors. We refer to this algorithm as the *leveled UniMax greedy algorithm*.

Let x be the point which we currently insert. We assign a color to x in two steps. First we assign x to a *level*, denoted by $\ell(x)$. Once x is assigned to level $\ell(x)$ we give it an actual color among the set of colors dedicated to $\ell(x)$. We maintain the invariant that each color is used by at most one level. Formally, the colors that we use are pairs $(\ell(x), c(x)) \in \mathbb{Z}^2$, where $\ell(x)$ is the level of x, and $c(x)$ is its integer color within that level.

Modifying the definition from the UniMax greedy algorithm, we say that point x *sees* point y (or that point y is *visible* to x) iff for every point z between x and y, $\ell(z) < \ell(y)$. When x is inserted we set $\ell(x)$ to be the smallest level ℓ such that either to the left of x or to the right of x (or in both directions) there is no point y visible to x at level ℓ.

To give x a color, we now consider only the points of level $\ell(x)$ that x can see. That is, we discard every point y such that $\ell(y) \neq \ell(x)$, and every point y such that $\ell(y) = \ell(x)$ and there is a point z between x and y such that $\ell(z) > \ell(y)$. We apply the UniMax greedy algorithm so as to color x with respect to the sequence P_x of the remaining points, using the colors of level $\ell(x)$ only. That is, we give x the color $(\ell(x), c(x))$, where $c(x)$ is the smallest color that ensures that the coloring of P_x maintains the unique maximum color condition. This completes the description of the algorithm. See Figure 1 for an illustration.

We begin the analysis of the algorithm by making a few observations on its performance:

(a) Suppose that a point x is inserted and is assigned to level $i > 1$. Since x was not assigned to any level $j < i$, it must see a point ℓ_j at level j that lies to its left, and another such point r_j that lies to its right. Let $E_j(x)$ denote the interval $[\ell_j, r_j]$. Note that, by definition, these intervals are *nested*, that is, $E_j(x) \subset E_k(x)$ for $j < k < i$. See Figure 1.

(b) We define a *run* at level i to be a maximal sequence of points $x_1 < x_2 < \cdots < x_k$ at level i, such that all points between x_1 and x_k that are distinct from $x_2, x_3, \ldots, x_{k-1}$ are assigned to levels smaller than i. Whenever a new point x is assigned to level i and is inserted into a run of that level, it is always inserted either to the left or to the right of all points in the run. Moreover, the actual color that x gets is determined solely from the colors of the points already in the run. See Figure 1.

(c) The runs keep evolving as new points are inserted. A run may either grow when a new point of the same level is inserted at its left or right ends (note that other points at smaller levels may separate the new point from the former end of the run), or split into two runs when a point of a higher level is inserted somewhere between its ends.

(d) As in observation (a), the points at level i define *intervals*, called *i-intervals*. Any such interval E is a contiguous subsequence $[x, y]$ of P, so that x and y are both at level i, and all the points between x and y have smaller levels. E is formed when the second of its endpoints, say x, is inserted. We say that x *closes* the interval E, and refer to it as a *closing point*. Note that, by construction, x cannot close another interval.

(e) Continuing observation (a), when x is inserted, it *destroys* the intervals $E_j(x)$, for $j < i$, that it is inserted into, and only these intervals. That is, each of these intervals now contains a point with a level greater than that of its endpoints, so it is no longer a valid interval. We charge x to the set of the closing endpoints of all these intervals. Clearly, none of these points will ever be charged again by another insertion (since it is the closing endpoint of only one interval, which is now destroyed). We maintain a forest F, whose nodes are all the points of P. The leaves of F are all the points at level 1. When a new point x is inserted, we make it a new root of F, and the parent of all the closing points that it charges. Since these points have smaller levels than x, and since none of these points becomes a child of another parent, it follows that F is indeed a forest.

Note that the non-closing points can only be roots of trees of F. Note also that a node at level i has exactly $i-1$ children, exactly one at each level $j < i$. Hence, each tree of F is a *binomial tree* (see [8]); if its root has level i then it has 2^i nodes.

This implies that if m is the maximal level assigned after n points have been inserted, then we must have $2^m \le n$, or $m \le \log n$. That is, the algorithm uses at most $\log n$ levels.

We next prove that our algorithm uses only $O(\log n)$ colors at each level. We recall the way runs evolve: They grow by adding points at their right or left ends, and they split into a prefix and suffix subruns, when a point with a larger level is inserted in their middle.

Lemma 3.1. *At any time during the insertion process, the colors assigned to the points in a run form a sequence of the form $C_0(a, b)$ (as defined in Section 2). Moreover, when the j-th smallest color of level i is given to a point x, the run to which x is appended has at least $2^{j-2} + 1$ elements (including x).*

Proof: The proof proceeds by induction through the sequence of insertion steps, and is based on the following observation. Let σ be a contiguous subsequence of the complete binary tree sequence S_{k-1}, and let x be a point added, say, to the left of σ. If we assign to x color $c(x)$, using the UniMax greedy algorithm, then $(c(x))\|\sigma$ is a contiguous subsequence of either S_{k-1} or S_k. The latter happens only if σ contains $S_{k-2}\|(k-1)$ as a prefix. Symmetric properties hold when x is inserted to the right of σ. We omit the straightforward proof of this observation. □

As a consequence, we obtain the following result.

Theorem 3.2. *(a) The algorithm uses at most $(2 + \log n)\log n$ colors.*
(b) At any time, the coloring is conflict-free.
(c) In the worst case the algorithm may be forced to use $\Omega(\log^2 n)$ colors after n points are inserted.

Proof: (a) We have already argued that the number of levels is at most $\log n$. Within a level i, the k-th smallest color is assigned when a run contains at least 2^{k-2} points. Hence $2^{k-2} \le n$, or $k \le 2 + \log n$, and (a) follows.

To show (b), consider an arbitrary interval I. Let ℓ be the highest level of a point in I. Let $\sigma = (y_1, y_2, \ldots, y_j)$ be the sequence of the points in I of level ℓ. Since ℓ is the highest level in I,

σ is a contiguous subsequence of some run, and, by Lemma 3.1, the sequence of the colors of its points is also of the form $C_0(a', b')$. Hence, there is a point $y_i \in \sigma$ which is uniquely colored among $y_1, y_2, \ldots, y_j$ by a color of level ℓ.

To show (c), we construct a sequence P so as to force its coloring to proceed level by level. We first insert 2^{k-1} points from left to right, thereby making them all be assigned to level 1, and to be colored with k different colors of that level. Let P_1 denote the set of these points. We next insert a second batch of 2^{k-2} points from left to right. The first point is inserted between the first and second points of P_1, the second point between the third and fourth points of P_1, and so on, where the j-th new point is inserted between the $(2j-1)$-th and $(2j)$-th points of P_1. By construction, all points in the second batch are assigned to level 2, and they are colored with $k-1$ different colors of that level. Let P_2 denote the set of all points inserted so far. P_2 is the concatenation of 2^{k-2} triples, where the levels in each triple are $(1, 2, 1)$. We now insert a third batch of 2^{k-3} points from left to right. The first point is inserted between the first and second triples of P_2, the second point between the third and fourth triples of P_2, and so on, where the j-th new point is inserted between the $(2j-1)$-th and $(2j)$-th triples of P_2. By construction, all points in the third batch are assigned to level 3, and they are colored with $k-2$ different colors of that level.

The construction is continued in this manner. Just before inserting the i-th batch of 2^{k-i} points, we have a set P_{i-1} of $2^{k-1} + \cdots + 2^{k-i+1}$ points, which is the concatenation of 2^{k-i+1} tuples, where the sequences of levels in each of these tuples are all identical, and are equal to the "complete binary tree sequence" $C_0(1, 2^{i-1} - 1)$, as defined in Section 2 (whose elements now encode levels rather than colors). The points of the i-th batch are inserted from left to right, where the j-th point is inserted between the $(2j-1)$-th and $(2j)$-th tuples of P_{i-1}. By construction, all points in the i-th batch are assigned to level i, and they are colored with $k-i+1$ different colors of that level. Proceeding in this manner, we end the construction by inserting the $(k-1)$-th batch, which consists of a single point that is assigned to level k. Altogether we have inserted $n = 2^k - 1$ points, and forced the algorithm to use $k + (k-1) + \cdots + 1 = k(k+1)/2 = \Omega(\log^2 n)$ different colors. $\square$

Remark: One can modify the algorithm so that the set of colors that it uses can be identified with (a subset of a prefix of) the integers, and so that it maintains the property of the UniMax greedy algorithm: At any time t and for any interval I, there is a unique point in I with maximum color. The modified algorithm also uses $O(\log^2 n)$ colors.

Specifically, we proceed as follows. Suppose first that n is known in advance. Order the pairs $(k, i) \in \{1, \ldots, \log n\} \times \{1, \ldots, 2 + \log n\}$ lexicographically, i.e., $(k, i) < (k', i')$ if $k < k'$ or ($k = k'$ and $i < i'$). Let $f(k, i)$ be the rank of the pair (k, i) in this lexicographic order. Then the set of numbers $f(k(p), i(p))$, where $p \in P$ is assigned level $k(p)$ and the $i(p)$-th color within that level, is (a subset of) a prefix of the integers, and the unique maximum color property is satisfied.

If n is not known in advance, we apply the same strategy as the one discussed at the end of the preceding section. That is, when the number of inserted points reaches one of the values 2^{2^i}, for $i \geq 0$, we start coloring new points with a completely new set of colors, which are mapped lexicographically onto integer values that are larger than the largest integer color used so far.

4 An Efficient Randomized Algorithm

We next modify the UniMax greedy deterministic algorithm into the following randomized algorithm, which we call the *randomized UniMax greedy algorithm*. The randomized UniMax greedy algorithm does not partition the points into levels but assigns a color directly, using the following randomized variant of the UniMax greedy strategy.

Let p be the next point inserted. Recall that p *sees* a point x (alternatively, the color $c(x)$) if all the colors of points between p and x (exclusive) have color smaller than $c(x)$. 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 $1/2$.

By the same reasoning as for the UniMax greedy algorithm, the coloring produced by the randomized UniMax greedy algorithm is conflict free at any stage. We next show that it used $O(\log n)$ colors with high probability.

Lemma 4.1. *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 which is equal to the set of points of color i (resp., of color $\geq i$). Then*

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

Proof. Assume first that p is neither the leftmost nor the rightmost point at the time of its insertion. Let Q be the set of points inserted before p. Fix a point $p_\ell \in Q$ to the left of p, and a point $p_r \in Q$ to the right of p. Consider all random choices (which we refer to as "executions") of the randomized UniMax algorithm, in which p_ℓ, p, and p_r end up as three consecutive points in $C_{\geq i}$.

In at least $1/2$ of these executions p_ℓ gets a color greater than i (either because it is ineligible for color i, or because the coin toss has moved it to color $i+1$); similarly, in at least $1/2$ of these executions p_r gets a color greater than i. Since the coin tosses of p_r are independent of those of p_ℓ, both p_ℓ and p_r get color greater than i in at least a $1/4$ of these executions. In these cases, p is eligible for color i, and with probability $1/2$ does get that color. Hence, in at least $1/8$ of the above executions p gets color i. Since this is true for every choice of p_ℓ and p_r, the lemma follows.

If p is the leftmost or rightmost point, then it is eligible for color i (assuming it has reached $C_{\geq i}$) with probability $1/2$, and if p is the first inserted point then it is eligible for color i with probability 1. Hence, the preceding argument implies the lemma in this case too. ∎

Theorem 4.2. *The randomized UniMax greedy algorithm uses $O(\log n)$ colors with high probability.*

Proof. Using the same notation as above, Lemma 4.1 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,

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

from which the lemma follows. ∎

Remark: We leave it as an open problem to determine whether a non-oblivious adversary can cause the algorithm to use more than $\Theta(\log n)$ colors.

5 Random Insertion Order

In this section we consider the special case where the points are inserted in a *random* order, and where we color them by the UniMax greedy algorithm of Section 2. We have simulated the execution of the UniMax greedy algorithm under such an insertion order. The results of the simulation strongly suggest the following conjecture:

Conjecture 5.1. *For each integer $k \geq 1$, the expected frequency of the color k in $C(P(t))$, as generated by the UniMax greedy algorithm, converges to $\frac{1}{3}\left(\frac{2}{3}\right)^{k-1}$, as $t \to \infty$.*

Assuming Conjecture 5.1, the following is an easy consequence.

Corollary 5.2. *If each point is inserted into P at a random place, the expected value of $c_{\max}(P(t))$, under the UniMax greedy algorithm, is $O(\log t)$. This also holds with high probability, if the constant of proportionality is chosen sufficiently large.*

Proof. Let $P(n)$ be a set of n points inserted in a random order. Let X_k be a random variable counting the number of points in $P(n)$ that were colored with k by the UniMax greedy algorithm. Let I_k be the indicator variable for the color k to appear at all.

We are interested in the number of colors used, that is $Y := \sum_k I_k$.

Assume that $\mathbf{E}(X_k) = \frac{1}{3}(\frac{2}{3})^{k-1}n$. Then, using Markov's Inequality, $\mathbf{E}(I_k) = \mathbf{Pr}\{I_k = 1\} = \mathbf{Pr}\{X_k \geq 1\} \leq \mathbf{E}(X_k)$. Hence,

$$
\begin{aligned}
\mathbf{E}(Y) = \mathbf{E}\left(\sum_{1\leq k} I_k\right) &= \mathbf{E}\left(\sum_{1\leq k<1+\log_{3/2} n} I_k\right) + \mathbf{E}\left(\sum_{k\geq 1+\log_{3/2} n} I_k\right) \\
&\leq 1 + \log_{3/2} n + \sum_{k\geq 1+\log_{3/2} n} \frac{1}{3}\left(\frac{2}{3}\right)^k n \\
&\leq 1 + \log_{3/2} n + \sum_{i\geq 0} \frac{1}{3}\left(\frac{2}{3}\right)^i \\
&= \log_{3/2} n + 2 \ .
\end{aligned}
$$

Arguing as in the proof of Theorem 4.2, we also have

$$
\mathbf{Pr}\Big\{\text{more than } c\log_{3/2} n \text{ colors are used}\Big\} = \mathbf{Pr}\Big\{I_{\lceil c\log_{3/2} n\rceil} = 1\Big\} \leq \frac{1}{3}\left(\frac{2}{3}\right)^{\lceil c\log_{3/2} n\rceil - 1} n \leq \frac{1}{2n^{c-1}}.
$$

$\square$

At this stage, we do not have a complete proof of Conjecture 5.1. We do have some partial results that we now present. In particular, they show that Conjecture 5.1 holds for $k = 1, 2, 3$. Completing the proof is one of the major open problems raised in this paper.

Lemma 5.3. *The expected number of points assigned the color 1, after a random insertion of t points, is $\frac{t+1}{3}$, for $t \geq 2$.*

Proof: Denote by X_i the random variable whose value is the number of 1's after the insertion of the first i points. Then $X_{i+1} = X_i + Y_i$, where Y_i is an indicator variable, equal to 1 if the $(i+1)$-st point p_{i+1} is colored by 1, and to 0 otherwise. Note that p_{i+1} is colored by 1 if and only if it is inserted at a place that is not adjacent to any point colored 1. Each of the current X_i 1-colored points has two adjacent insertion places, and all these places are distinct, because $P(i)$ does not contain two adjacent points colored 1. Hence, out of the $i+1$ available insertion places, $i+1-2X_i$ will cause p_{i+1} to be colored 1. Taking expectations, we obtain

$$\mathbf{E}(X_{i+1}) = \mathbf{E}(X_i) + \mathbf{E}(Y_i) = \mathbf{E}(X_i) + \mathbf{E}(\mathbf{E}(Y_i \mid X_i)) =$$
$$\mathbf{E}(X_i) + \mathbf{E}\left(\frac{i+1-2X_i}{i+1}\right) = \mathbf{E}(X_i) + \frac{i+1-2\mathbf{E}(X_i)}{i+1},$$

or $\mathbf{E}(X_{i+1}) = \frac{i-1}{i+1}\mathbf{E}(X_i) + 1$, for $i \geq 2$. The solution of this recurrence, with the initial value $\mathbf{E}(X_2) = 1$, is easily seen to be $\mathbf{E}(X_t) = \frac{t+1}{3}$, for $t \geq 2$. $\square$

Analysis for $k \geq 2$. We next present a framework for estimating the expected number of points that are assigned the color k, for $k \geq 2$. We apply this framework to get a complete solution for $k = 2, 3$. We fix k, and define a *k-state* to be any valid contiguous sequence of colors in $\{1, \ldots, k\}$ that may show up in $C(P(t))$, delimited on both sides by $*$, which designates a color greater than k. The validity of a state means that it satisfies the unique-maximum color invariant: Any contiguous nonempty subsequence of s has a unique largest element. We refer to the portion of a state that excludes the $*$'s as its *core*.

Denote by S_k the set of all k-states. For example, the set S_2 consists of the following states:

$$s_1 = \langle **\rangle, \quad s_2 = \langle *1*\rangle, \quad s_3 = \langle *2*\rangle, \quad s_4 = \langle *12*\rangle, \quad s_5 = \langle *21*\rangle, \quad s_6 = \langle *121*\rangle. \tag{1}$$

For example, the following sequence $C(P(t)) = (1\ 2\ 1\ 3\ 2\ 4\ 2\ 1\ 3\ 5\ 1\ 2\ 3)$ is decomposed into the following sequence of 2-states:

$$\Big(\langle *121*\rangle, \quad \langle *2*\rangle, \quad \langle *21*\rangle, \quad \langle **\rangle, \quad \langle *12*\rangle, \quad \langle **\rangle\Big)$$

We denote by S_k^+ the subset of S_k consisting of those k-states that contain the color k (necessarily at a unique location), and by S_k^- the subset of those states that do not contain k. We refer to states in S_k^+ (resp., S_k^-) as *major* k-states (resp., *minor* k-states). The *size* $|s|$ of a k-state s is the length of its core plus 1; it designates the number of places in s at which a new point can be inserted. For example, for 2-states we have $S_2^- = \{s_1, s_2\}$, $S_2^+ = \{s_3, s_4, s_5, s_6\}$. Also, we have $|s_1| = 1$, $|s_2| = |s_3| = 2$, $|s_4| = |s_5| = 3$, and $|s_6| = 4$.

Let $s \in S_k^+$. It has the form $(*ukv*)$, where u and v can be regarded as the cores of two respective $(k-1)$-states s_L and s_R. We refer to s_L and s_R as the *left wing* and the *right wing* of s, respectively. We have $|s| = |s_L| + |s_R|$. Care should be exercised in the treatment of s_L and s_R. Specifically, we will consider the actual sequence of colors $C(P(t))$ as a concatenation of states, which depends on the choice of k. We denote by $C^{(k)}(t)$ the (unique) partition of $C(P(t))$ into the concatenation of k-states, and refer to it as the *k-scenario*. Then, for a major state $s \in S_k^+$, its left and right wings are not counted as separate states in the k-scenario, but they are counted as states in the $(k-1)$-scenario.

We need one more notion. When we insert a new point into a k-state s, there are two possible outcomes: (i) The point gets a color smaller than or equal to k, in which case s is transformed to another, *single* state in S_k. (ii) The point gets a color greater than k, in which case s is split into two new k-states. Note that, for case (ii) to occur, s must be a *major* state (if s were minor, we could have assigned the color k to the new point). Moreover, in this case one of the two new states, s', must be a major state, and the other, s'' must be minor. We refer to this case by saying that s *spawns* s'' and is *transformed* into s'. (Note that not every insertion into a major state necessarily causes a spawning.)

It is easy to show that the size $|S_k|$ of S_k satisfies $|S_{k+1}| = |S_k| + |S_k|^2$, so $|S_k|$ is doubly exponential in k. We have $|S_1| = 2$, $|S_2| = 6$, $|S_3| = 42$, and $|S_4| = 1806$.

Let k be fixed. For states $s, r \in S_k$, we denote by a_{sr} the expected change in the number of states r that are generated by an insertion of a new point, conditioned on having chosen an insertion place at a state s (within $C^{(k)}$). For example, for $k = 2$, we have (see (1) for the notation)

$$a_{s_4s_1} = a_{s_4s_2} = a_{s_4s_3} = a_{s_4s_6} = \frac{1}{3}, \quad \text{and} \quad a_{s_4s_4} = -\frac{2}{3}$$

(in two of the three possible insertion places, s_4 is destroyed by the insertion, and in the third insertion it survives, so the net expected increase in the number of s_4-states is $0 \cdot \frac{1}{3} + (-1) \cdot \frac{2}{3} = -\frac{2}{3}$). Put $w_{sr} = |s| a_{sr}$, and let W denote the resulting matrix (w_{sr}).

We first provide some intuitive and informal derivation of the equations that we will rigorously derive shortly. Let $M_s^{(t)}$ denote the random variable equal to the number of k-states s in $C(P(t))$. Define the *frequency* of state s at time t to be $X_s^{(t)} = M_s^{(t)}/(t+1)$. Note that $|s|X_s^{(t)}$ is the frequency of the insertion places that belong to occurrences of s in $C(P(t))$. In particular, $\sum_{s \in S_k} |s| X_s^{(t)} = 1$, for each t. We also have

$$(t+2)\mathbf{E}(X_r^{(t+1)}) = (t+1)\mathbf{E}(X_r^{(t)}) + \sum_{s \in S_k} |s| a_{sr} \mathbf{E}(X_s^{(t)}). \tag{2}$$

Indeed, $|s|X_s^{(t)}$ is the probability that the next insertion place belongs to an occurrence of state s in $C(P(t))$, and a_{sr} is the corresponding conditional expected change in the number of occurrences of state r. Since $M_r^{(t)} = (t+1)X_r^{(t)}$ (resp., $M_r^{(t+1)} = (t+2)X_r^{(t+1)}$) is the number of occurrences of state r at time t (resp., $t+1$), the equality follows.

Letting $t \to \infty$, applying an informal limit process to (2), and denoting the limit of $\mathbf{E}(X_s^{(t)})$ as X_s, for $s \in S_k$, we arrive at the equations

$$X_r = \sum_{s \in S_k} |s| a_{sr} X_s = \sum_{s \in S_k} w_{sr} X_s.$$

We now proceed to justify this process rigorously.

Existence of limiting frequencies. The random insertion order defines in a natural way a *multi-type branching process* (see [2]). We briefly review the ingredients of the theory of branching processes that we need to apply. A (discrete) branching process of this kind manipulates objects (referred to as "particles") that can have a finite number m of types. Each type i is associated with weights $(\xi_{i,J})$, where J is a multi-set of types. The weight $\xi_{i,J}$ should be thought of as the relative frequency at which a particle of type i gives birth to the multi-set J (for each type j that appears μ times in J, the particle generates μ new particles of type j). Each particle giving birth dies immediately after doing so. Set $\xi_i = \sum_J \xi_{i,J}$. The process may then be formally defined as follows. Let $S(t)$ be the population at time t. Choose $x \in S(t)$ with probability $\xi_{i(x)}/\sum_{y\in S(t)} \xi_{i(y)}$, where $i(u)$ is the type of particle u. Then x gives birth to the multiset J with probability $\xi_{i(x),J}/\xi_{i(x)}$ and then dies.

In our case, the different particle types correspond to different state types in S_k. A state s of length ℓ has total weight ℓ. If some insertions into s produce the single state s' (without spawning), then $\xi_{s,\{s'\}} = j$, where j is the number of places at which this occurs. If some j insertions produce two states s', s'' (by spawning) then $\xi_{s,\{s',s''\}} = j$. The entries of our transition matrix W are then defined as $w_{sr} = \sum_{r\in J} \xi_{s,J}$, for $r \neq s$, and $w_{ss} = \left(\sum_{r\in J} \xi_{s,J}\right) - |s|$. See pp. 200–202 in [2] for a similar construction of a transition matrix for general multi-type processes (where the matrix is called the *infinitesimal generator* of a corresponding semigroup of mean matrices).

A standard trick in the theory of branching processes is to embed discrete branching processes of the kind described above into continuous-time branching processes, in which particles give birth in continuous time. More specifically, $S(t)$ evolves in continuous time. For any fixed time t, we associate, with each $x \in S(t)$ and each multi-set J, an exponenetial random variable with rate $w_{i(x),J}$. We then take the one with the smallest actual value—suppose this is the variable $w_{i(x'),J'}$, and it has the value h. Now the population at time $t+h$ is obtained from the population at time t by killing x' and replacing it by J'. We now obtain a new population $S(t+h)$, a new collection of exponential random variables, and the process continues.

If we extract from the continuous branching process only those times at which new particles are born, we obtain exactly the same discrete process that we started with (see [2] for details). In the terminology of the theory of branching processes, the discrete and continuous processes are the same, up to a time change. The reason for this round-about reasoning is that the theory of continuous-time branching process is better developed, and provides machinery for proving the existence of limit frequencies and for analyzing their properties. In particular, the limiting frequencies for the new continuous process (whose existence is established next) are identical to those of the original discrete process.

It is easy to see that the (continuous) branching process just defined is *supercritical*, and satisfies the $Z \log Z$ moment condition (see, e.g., [2] for background and details). It therefore follows (see, e.g., Theorem 2, p. 206, in [2]) that the limiting frequencies exist almost surely. We let X_s denote the expected limit relative frequency of state s in $C(P(t))$, when $t \to \infty$, where the non-limit frequencies are as defined above.

In addition, Theorem 2, p. 206, in [2], just cited, asserts that the limiting distribution $X = (X_s)_{s\in S_k}$ is given by the eigenvector of W^T corresponding to the largest eigenvalue. In our case, this does indeed coincide with our informal derivation, and means that X satisfies the linear system

$$(W^T X)_r = \sum_{s\in S_k} w_{sr} X_s = X_r, \qquad r \in S_k. \tag{3}$$

For example, for $k = 2$, the transition weights a_{sr} between the six states listed in (1) are given in the following matrix A, where $A_{ij} = a_{s_i s_j}$. (The fourth row of A has already been discussed.)

$$A = \begin{pmatrix} -1 & 1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & \frac{1}{2} & \frac{1}{2} & 0 \\ 0 & 0 & -1 & \frac{1}{2} & \frac{1}{2} & 0 \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} & -\frac{2}{3} & 0 & \frac{1}{3} \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} & 0 & -\frac{2}{3} & \frac{1}{3} \\ \frac{1}{2} & \frac{1}{2} & 0 & \frac{1}{4} & \frac{1}{4} & -\frac{1}{2} \end{pmatrix},$$

and

$$W = \begin{pmatrix} -1 & 1 & 0 & 0 & 0 & 0 \\ 0 & -2 & 0 & 1 & 1 & 0 \\ 0 & 0 & -2 & 1 & 1 & 0 \\ 1 & 1 & 1 & -2 & 0 & 1 \\ 1 & 1 & 1 & 0 & -2 & 1 \\ 2 & 2 & 0 & 1 & 1 & -2 \end{pmatrix}.$$

The system of equations for the limit distribution is $W^T X = X$. To normalize X, we extend it by the equation

$$\sum_i |s_i| X_i = X_1 + 2X_2 + 2X_3 + 3X_4 + 3X_5 + 4X_6 = 1,$$

which expresses the fact that the sum of lengths of the 2-states that compose $C(P(t))$ is equal to $|C(P(t))|$ (see above for a similar equation for the non-limit frequencies $X_s^{(t)}$). The solution of the extended system is

$$X = \left(\frac{1}{9}, \frac{1}{9}, \frac{2}{45}, \frac{1}{15}, \frac{1}{15}, \frac{2}{45}\right).$$

In particular, the expected limit frequency of color 1 is $X_2 + X_4 + X_5 + 2X_6 = \frac{1}{3}$ (in accordance with Lemma 5.3), and the expected limit frequency of color 2 is $X_3 + X_4 + X_5 + X_6 = \frac{2}{9}$. We have thus verified Conjecture 5.1 for $k = 2$:

Lemma 5.4. *The limit frequency of color 2 is* $2/9$.

Analysis of 3-states. The same machinery can be applied to the 42 states in S_3. The solution of (3) for $k = 3$ is presented in Table 1.

By adding up the frequencies of all major 3-states (those that contain the color 3), we verify Conjecture 5.1 for $k = 3$:

Lemma 5.5. *The limit frequency of color 3 is* $4/27$.

Open problem: Find closed-form expressions for the state frequencies for $k = 3$ (using the data in Table 1), and for $k > 3$. This may lead to a simple inductive proof of Conjecture 5.1.

Further analysis of k-states. The system (3) becomes considerably harder to solve explicitly for larger values of k, so we look for simpler relationships. Put

$$N_k = \sum_{s \in S_k^+} X_s, \qquad Z_k = \sum_{s \in S_k^-} X_s.$$

Note that N_k is the expected frequency of color k. Recall that Conjecture 5.1 says that $N_k = \frac{1}{3}\left(\frac{2}{3}\right)^{k-1}$.

Lemma 5.6. *For each $k \geq 2$ we have $2N_k + Z_k = N_{k-1} + Z_{k-1}$.*

Proof: Let s be a state in S_k^+, and let s_L (resp., s_R) denote the state obtained by taking the portion of s to the left (resp., right) of (the unique) k, and appending $*$ at the right (resp., left). If we repeat this splitting process to each state of S_k^+ in $C(P(t))$, and add to the output all states in S_k^- (which we leave intact), we obtain the set of all states of S_{k-1} that appear in $C(P(t))$. The sum of the frequencies of these states is clearly $N_{k-1} + Z_{k-1}$. On the other hand, by our construction, this sum is $2N_k + Z_k$, so the lemma follows. □

The following conjecture is *equivalent* to Conjecture 5.1.

Conjecture 5.7. *$N_k = Z_k$ for each $k \geq 1$.*

We verify the conjecture for $k = 1$, where $N_1 = Z_1 = \frac{1}{3}$, for $k = 2$, where $N_2 = Z_2 = \frac{2}{9}$, and for $k = 3$, where $N_3 = Z_3 = \frac{4}{27}$ (see Table 1).

Assuming that Conjecture 5.7 holds, and combining it with Lemma 5.6, we obtain $3N_k = 2N_{k-1}$, for $k \geq 2$, and $N_1 = \frac{1}{3}$. Hence

$$N_k = \frac{1}{3}\left(\frac{2}{3}\right)^{k-1}.$$

The converse direction is established in a similar manner: Conjecture 5.1 and Lemma 5.6 imply

$$Z_k = Z_{k-1} + N_{k-1} - 2N_k = Z_{k-1} - \frac{1}{9}\left(\frac{2}{3}\right)^{k-2},$$

for $k \geq 2$, and $Z_1 = \frac{1}{3}$. The solution of this recurrence is $Z_k = \frac{1}{3}\left(\frac{2}{3}\right)^{k-1} = N_k$, thus showing that the two conjectures are indeed equivalent.

6 Lower Bound for Online CF-Coloring in the Plane

We finally show that online conflict-free coloring of points in the plane, with respect to disks (of arbitrary radii), may require n colors in the worst case, and is therefore quite impractical. (Nevertheless, as mentioned in the introduction, the problem can be solved with much fewer colors for other kinds of ranges; see [6, 7].)

Theorem 6.1. *There exists a sequence P of n points in the plane, so that when these points are inserted according to their order in P, any online conflict-free coloring scheme with respect to disks has to use n different colors.*

Proof: We construct a sequence $P = (p_1, p_2, \ldots, p_n)$ with the following property

> (*) *For every $t = 2, 3, \ldots, n$, the edges of the Delaunay triangulation of the set $\{p_1, p_2, \ldots, p_t\}$ include all the edges $\{p_i, p_t\}$, $i = 1, 2, \ldots, t-1$.*

We prove the following stronger statement by induction on n:

> *For every n, every choice of distinct points $q_1, q_2, \ldots, q_n$ on the unit circle $\mathbb{S}^1$, and every $\varepsilon > 0$ there exists a sequence $(p_1, p_2, \ldots, p_n)$ with the property (*) such that $\|p_i - q_i\| \leq \varepsilon$, and p_i lies on the radius oq_i, for every i.*

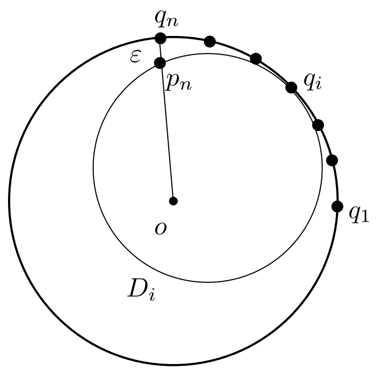

Figure 2: The construction that requires n colors for the planar case with disk ranges.

For the induction step, given $q_1, \ldots, q_n$ and $\varepsilon < \frac{1}{2}$, let p_n be obtained by moving q_n by ε towards the center o of $\mathbb{S}^1$. We note that the Delaunay graph of $\{q_1, q_2, \ldots, q_{n-1}, p_n\}$ contains all edges $\{q_i, p_n\}$, $i = 1, 2, \ldots, n$. Indeed, there is a circle γ_i tangent to $\mathbb{S}^1$ from the inside at q_i and passing through p_n, and the closed disk D_i bounded by γ_i contains q_i, p_n, and no other q_j. See Figure 2. Let $\delta_i > 0$ denote the minimum distance from any q_j, $j \neq i$, to D_i.

We apply the induction hypothesis with $q_1, \ldots, q_{n-1}$ and with $\varepsilon^* < \min\{\varepsilon, \delta_1, \ldots, \delta_{n-1}\}$, obtaining a sequence $(p_1, \ldots, p_{n-1})$. We can now verify that, by construction, for every $i = 1, 2, \ldots, n-1$, the disk D_i contains p_i and p_n but no other p_j. $\Box$

7 Conclusion

The paper still leaves many open problems, some of which have already been listed earlier. Here are several concluding open problems.

(a) Theorem 6.1, and the initial encouraging results of Chen, Kaplan and Sharir [7] and of Bar Noy et al. [4], as reviewed in the introduction, raise many interesting open problems, such as: (i) Obtain *deterministic* algorithms with good performance for the cases studied in [7], viz. where the ranges are halfplanes, congruent disks, and nearly equal axis-parallel rectangles. (ii) Improve further the performance of the algorithms of [7]. (iii) Find solutions with good performance for other ranges, such as arbitrary axis-parallel rectangles. (iv) Extend the results to $d \geq 3$ dimensions.

(b) It is likely that the bound in Theorem 6.1 improves significantly if the points are chosen from some random distribution, extending our conjecured bounds of Section 5.2 to two (and higher?) dimensions.

(c) Can one obtain better upper bounds for online *k-CF-coloring* ($k \geq 2$) of points in the plane with respect to disks? Namely, online color the points so that, at any given time t and for any disk D, there is at least one color that is assigned to at least one but at most k points of $P(t) \cap D$. For $k = 1$, this is the CF-coloring problem, where we have just shown a lower bound of n, but perhaps this can be improved when $k \geq 2$. See [15] for results concerning k-CF-coloring in the static case.

(d) Finally, can one obtain an efficient randomized algorithm, for the online CF-coloring problem on the line, which works against an adaptive adversary (that is, an adversary that observes the actions of the randomized online algorithm and decides where to insert the next point based on these actions)?

Acknowledgement

The first author would like to thank Sariel Har-Peled for helpful discussions.

References

[1] N. Alon and J. Spencer, *The Probabilistic Method*, J. Wiley and Sons, New York, NY, 1992.

[2] K. B. Athreya and P. E. Ney, *Branching Processes*, Die Grundlehren der mathematischen Wissenschaften, Band 196, Springer-Verlag, New York, 1972.

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

[4] A. Bar-Noy, P. Cheilaris, S. Smorodinsky, Online conflict-free colorings for hypergraphs, manuscript, 2006.

[5] A. Borodin and R. El Yaniv, *Online Computations and Competitive Analysis*, Cambridge University Press, 1998.

[6] K. Chen, How to play a coloring game against a color-blind adversary, to appear in *Proc. 22nd ACM Sympos. Comput. Geom.*, 2006.

[7] K. Chen, H. Kaplan and M. Sharir, Online CF coloring for halfplanes, congruent disks, and axis-parallel rectangles, Manuscript, 2006.

[8] T.H. Cormen, C.E. Leiserson, and R.L. Rivest, *Introduction to Algorithms*, MIT Press, Cambridge, MA, 1990.

[9] J.S. Deogun, T. Kloks, D. Kratsch, and H. Müller, On vertex ranking for permutations and other graphs, *Proc. 11th Annu. Sympos. Theoretical Aspects of Computer Science*, P. Enjalbert, E.W. Mayr, K.W. Wagner, (eds.), Lecture Notes in Computer Science 775, Springer-Verlag, Berlin, pp. 747–758, 1994.

[10] K. Elbassioni and N. Mustafa, Conflict-free colorings for rectangle ranges, *23rd Internat. Sympos. Theoretical Aspects of Computer Science* (STACS 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(1):94–136, 2003.

[12] A. Fiat, M. Levy, J. Matoušek, E. Mossel, J. Pach, M. Sharir, S. Smorodinsky, U. Wagner, and E. Welzl, Online conflict-free coloring for intervals, in *Proc. 16th ACM-SIAM Sympos. Discrete Algo.*, 2005, 545–554.

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

[14] 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 (Eds.), Springer-Verlag, Heidelberg, 2003, pp. 665–671.

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

[16] S. Smorodinsky, On the chromatic number of some geometric hypergraphs, *Proc. 17th Annual ACM-SIAM Sympos. Discrete Algorithms* (SODA 2006), 316–323.

State	Frequency	Numerator	As a fraction	With factored denominator
0	0.03704	1764000	1/27	$1/3^3$
1	0.03704	1764000	1/27	$1/3^3$
12	0.02222	1058400	1/45	$1/3^2 5^1$
21	0.02222	1058400	1/45	$1/3^2 5^1$
2	0.01481	705600	2/135	$2/3^3 5^1$
121	0.01481	705600	2/135	$2/3^3 5^1$
13	0.00800	381024	1/125	$1/5^3$
31	0.00800	381024	1/125	$1/5^3$
12321	0.00388	184800	11/2835	$11/3^4 5^1 7^1$
3	0.00948	451584	32/3375	$32/3^3 5^3$
131	0.00652	310464	22/3375	$22/3^3 5^3$
123	0.00366	174440	89/24300	$89/2^2 3^5 5^2$
321	0.00366	174440	89/24300	$89/2^2 3^5 5^2$
1231	0.00737	350840	179/24300	$179/2^2 3^5 5^2$
1321	0.00737	350840	179/24300	$179/2^2 3^5 5^2$
32	0.00167	79576	203/121500	$203/2^2 3^5 5^3$
23	0.00167	79576	203/121500	$203/2^2 3^5 5^3$
132	0.00686	326536	833/121500	$833/2^2 3^5 5^3$
231	0.00686	326536	833/121500	$833/2^2 3^5 5^3$
213121	0.00156	74466	197/126000	$197/2^4 3^2 5^3 7^1$
121312	0.00156	74466	197/126000	$197/2^4 3^2 5^3 7^1$
2321	0.00233	111160	397/170100	$397/2^2 3^5 5^2 7^1$
1232	0.00233	111160	397/170100	$397/2^2 3^5 5^2 7^1$
232	0.00093	44464	397/425250	$397/2^1 3^5 5^3 7^1$
121321	0.00191	90755	2593/1360800	$2593/2^5 3^5 5^2 7^1$
123121	0.00191	90755	2593/1360800	$2593/2^5 3^5 5^2 7^1$
12312	0.00307	146405	4183/1360800	$4183/2^5 3^5 5^2 7^1$
21321	0.00307	146405	4183/1360800	$4183/2^5 3^5 5^2 7^1$
12131	0.00344	163928	20491/5953500	$20491/2^2 3^5 5^3 7^2$
13121	0.00344	163928	20491/5953500	$20491/2^2 3^5 5^3 7^2$
1312	0.00485	231208	28901/5953500	$28901/2^2 3^5 5^3 7^2$
2131	0.00485	231208	28901/5953500	$28901/2^2 3^5 5^3 7^2$
1213	0.00588	279848	34981/5953500	$34981/2^2 3^5 5^3 7^2$
3121	0.00588	279848	34981/5953500	$34981/2^2 3^5 5^3 7^2$
213	0.00835	397528	49691/5953500	$49691/2^2 3^5 5^3 7^2$
312	0.00835	397528	49691/5953500	$49691/2^2 3^5 5^3 7^2$
2132	0.00198	94467	31489/15876000	$31489/2^5 3^4 5^3 7^2$
2312	0.00198	94467	31489/15876000	$31489/2^5 3^4 5^3 7^2$
21312	0.00240	114326	57163/23814000	$57163/2^4 3^5 5^3 7^2$
1213121	0.00099	47206	23603/23814000	$23603/2^4 3^5 5^3 7^2$
12132	0.00104	49397	49397/47628000	$49397/2^5 3^5 5^3 7^2$
23121	0.00104	49397	49397/47628000	$49397/2^5 3^5 5^3 7^2$

Table 1: The frequencies of 3-states. The second column is the numerator of the frequency under the common denominator $47628000 = 2^5 3^5 5^3 7^2$.