Deterministic Conflict-Free Coloring for Intervals: from Offline to Online

AMOTZ BAR-NOY
Brooklyn College and the Graduate Center, City University of New York
PANAGIOTIS CHEILARIS
City University of New York and National Technical University of Athens
and
SHAKHAR SMORODINSKY
Ben-Gurion University

We investigate deterministic algorithms for a frequency assignment problem in cellular networks. The problem can be modeled as a special vertex coloring problem for hypergraphs: In every hyperedge there must exist a vertex with a color that occurs exactly once in the hyperedge (the conflict-free property). We concentrate on a special case of the problem, called conflict-free coloring for intervals. We introduce a hierarchy of four models for the above problem: (i) static, (ii) dynamic offline, (iii) dynamic online with absolute positions, (iv) dynamic online with relative positions. In the dynamic offline model, we give a deterministic algorithm that uses at most $\log_{3/2} n + 1 \approx 1.71 \log_2 n$ colors and exhibit inputs that force any algorithm to use at least $3 \log_5 n + 1 \approx 1.29 \log_2 n$ colors. For the online absolute positions model, we give a deterministic algorithm that uses at most $3\lceil \log_3 n \rceil \approx 1.89 \log_2 n$ colors. To the best of our knowledge, this is the first deterministic online algorithm using $O(\log n)$ colors, in a non-trivial online model. In the online relative positions model, we resolve an open problem by showing a tight analysis on the number of colors used by the first-fit greedy online algorithm. We also consider conflict-free coloring only with respect to intervals that contain at least one of the two extreme points.

Categories and Subject Descriptors: F.2 [**Theory of Computation**]: Analysis of algorithms and problem complexity

General Terms: Algorithms, Theory

A preliminary version of this work appeared in the *Proceedings of the 18th Annual ACM Symposium on Parallelism in Algorithms and Architectures* (SPAA), 2006.
Amotz Bar-Noy, Computer and Information Science Department, Brooklyn College, 2900 Bedford Avenue, Brooklyn, NY, 11210, USA, and the Graduate Center, City University of New York, 365 Fifth Avenue, New York, NY, 10016; amotz@sci.brooklyn.cuny.edu. Supported by the CUNY Collaborative Incentive Research Grants Program Round 11 (2004–2006).
Panagiotis Cheilaris, The Graduate Center, City University of New York, 365 Fifth Avenue, New York, NY, 10016, USA; philaris@sci.brooklyn.cuny.edu. Panagiotis Cheilaris is also affiliated with the National Technical University of Athens, Greece. Supported by the European Social Fund (75%) and National Resources (25%), under the program EPEAEK II, 'Heraclitus'. Also supported by the CUNY Collaborative Incentive Research Grants Program Round 11 (2004–2006).
Shakhar Smorodinsky, Department of Mathematics, Ben-Gurion University, Be'er Sheva, 84105, Israel; shakhar@math.bgu.ac.il; http://www.math.bgu.ac.il/~shakhar/. Supported by the NSF Mathematical Sciences Postdoctoral Fellowship award 0402492.

Additional Key Words and Phrases: Online algorithms, cellular networks, frequency assignment, conflict free, coloring

1. INTRODUCTION

A *vertex coloring* of a graph $G = (V, E)$ is a function $C : V \to \mathbb{N}^+$ such that for every edge $\{v_1, v_2\} \in E$: $C(v_1) \neq C(v_2)$. A *hypergraph* $G = (V, E)$ is a generalization of a graph for which *hyperedges* can be arbitrary-sized non-empty subsets of V. There are several ways to define vertex coloring in hypergraphs: On one extreme, it is required that for every hyperedge that contains more than two vertices, not all colors are the same (there are at least two colors); on the other extreme, it is required that for every edge, no color is repeated (all the colors are different). In between these two extremes, there is another possible generalization: A vertex coloring C of hypergraph G is called *conflict-free* if in every hyperedge e there is a vertex whose color is unique among all other colors in the hyperedge. Formally:

$$\forall e \in E : \exists v \in e : \forall v' \in e : v' \neq v \to C(v') \neq C(v) \ .$$

Conflict-free coloring models frequency assignment for cellular networks. A cellular network consists of two kinds of nodes: *base stations* and *mobile agents*. Base stations have fixed positions and provide the backbone of the network; they are modeled by vertices in V. Mobile agents are the clients of the network and they are served by base stations. This is done as follows: Every base station has a fixed frequency; this is modeled by the coloring C, i.e., colors represent frequencies. If an agent wants to establish a link with a base station it has to tune itself to this base station's frequency. Since agents are mobile, they can be in the range of many different base stations. The range of communication of every agent is modeled by a hyperedge $e \in E$, which is the set of base stations that can communicate with the agent. To avoid interference, the system must assign frequencies to base stations in the following way: For any range, there must be a base station in the range with a frequency that is not reused by some other base station in the range. This is modeled by the conflict-free property. One can solve the problem by assigning n different frequencies to the n base stations. However, using many frequencies is expensive, and therefore, a scheme that reuses frequencies, where possible, is preferable.

The study of conflict-free colorings was originated in the work of Even et al. [2003] and Smorodinsky [2003]. In addition to the practical motivation described above, this new coloring model has drawn much attention of researchers through its own theoretical interest and such colorings have been the focus of several recent papers (see, e.g., [Even et al. 2003; Smorodinsky 2003; Pach and Tóth 2003; Har-Peled and Smorodinsky 2005; Fiat et al. 2005; Elbassioni and Mustafa 2006; Chen et al. 2006; Alon and Smorodinsky 2006; Ajwani et al. 2007; Smorodinsky 2007]).

Fiat et al. [2005] considered the special case of the problem where the hypergraph is defined as follows: Vertices are identified by points that lie on a line and E consists of all subsets of V defined by intervals intersecting at least one vertex. A line with n points has $n(n+1)/2$ such subsets (for every $i \in \{1, \ldots, n\}$, there are $n - i + 1$ different subsets containing i points). For $n = 5$, these subsets are shown in figure 1.

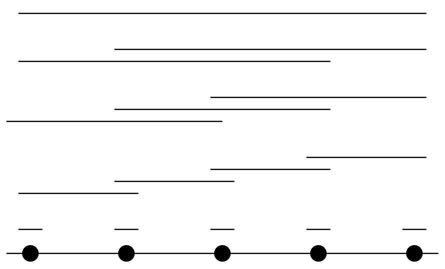

Fig. 1. Points on a line and intervals

We call these subsets intervals since for our purpose, two intervals are equivalent if they contain the same vertices. We represent colorings by listing the colors of points from left to right in a *string*. For example, for the points in figure 1 ($n = 5$), 12312 is a conflict-free coloring, whereas 12123 is not (because the interval containing the four leftmost points does not have a unique color).

Conflict-free coloring for intervals is important because it can model assignment of frequencies in networks where the agents' movement is approximately unidimensional, e.g., the cellular network that covers a single long road and has to serve agents that move along this road. In some kinds of networks, like wireless sensor networks, the density of base stations can be very high and non-uniform. Moreover, the ranges of the agents can also be non-uniform. To ensure the absence of conflicts we require the conflict-free property for all possible intervals. Also, conflict-free coloring for intervals plays a role in the study of conflict-free coloring for more complicated range spaces and it is a special case of conflict free coloring points on the plane with respect to disks, when all points to be colored are collinear (see [Even et al. 2003]).

The *static* version of the problem, where the n points are to be colored simultaneously, is solved in [Even et al. 2003]. For $n = 2^k - 1$, the coloring C^k is defined recursively as follows: $C^1 = 1$ and $C^{k+1} = C^k \circ (k+1) \circ C^k$ (where $\circ$ is the concatenation operator for strings). The coloring C^k is conflict-free and uses k colors for $2^k - 1$ points. For n with $n < 2^k - 1$, the prefix (in fact, any substring) of length n of C^k is conflict-free. Even et al. [2003] also show that this coloring with $1 + \lfloor \lg n \rfloor$ colors is the best possible. Observe that C^k has the property that the maximum color in each interval is always unique. Coloring intervals with a unique maximum is called *vertex ranking* of paths, or *ordered coloring* of paths, and in that context similar results were obtained in [Iyer et al. 1988; Katchalski et al. 1995].

The problem becomes more interesting when the vertices are given online by an adversary. Namely, at every given time step $t \in \{1, \ldots, n\}$, a new vertex $v_t \in V$ is given and the algorithm must assign v_t a color such that the coloring is a conflict-free coloring of the hypergraph that is induced by the vertices $V_t = \{v_1, \ldots, v_t\}$. Once v_t is assigned a color, that color cannot be changed in the future. It is desirable to avoid recoloring for the following technical reason: If a base station changes color, there might be disruption of service for all agents connected to it.[1] We are interested in an online setting, in which the algorithm has no knowledge of how vertices will

[1]A model in which a small number of recolorings is allowed is presented in [Bar-Noy et al. 2007].

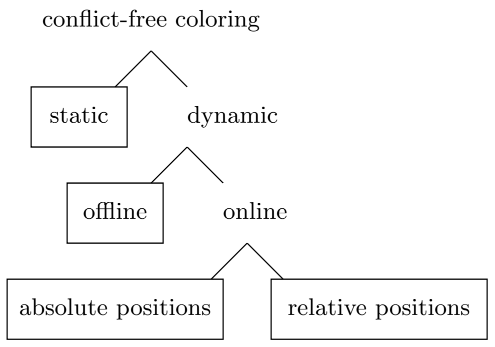

Fig. 2. Models for conflict-free coloring for intervals

be requested in the future. For this version of the problem, in the case of intervals, Fiat et al. [2005] provide several algorithms. Their deterministic algorithm uses $O(\log^2 n)$ colors in the worst case; the result is tight because their algorithm requires $\Omega(\log^2 n)$ colors on some inputs. Their randomized algorithm uses $O(\log n \log\log n)$ colors with high probability. Recently, randomized algorithms that use $O(\log n)$ colors with high probability have been obtained ([Chen 2006; Chen et al. 2007; Bar-Noy et al. 2007]). All of the randomized algorithms assume the slightly weaker *oblivious* adversary model, in which the adversary has to commit on a specific input sequence before revealing the first vertex to the algorithm (see [Borodin and El-Yaniv 1998]).

Our contribution. We introduce a *hierarchy* of four models for the above conflict-free coloring problem for hypergraphs: (i) static, (ii) dynamic offline, (iii) dynamic online with absolute positions, and (iv) dynamic online with relative positions. Below we define these four models. The relationship among them is shown in figure 2.

—In the *static model*, the complete hypergraph G is given, and a conflict-free coloring for G must be found by the algorithm.

In dynamic models, a *sequence* $\{G_t\}_{t=1}^n$ of hypergraphs (with $G = G_n$) is given where G_t has t vertices and, for $t > 1$, G_{t-1} is an induced subhypergraph[2] of G_t; for every t a conflict-free coloring for G_t must be found that, for $t > 1$, extends the coloring of G_{t-1} (i.e., the algorithm can not change colors of vertices). Alternatively, the input is a permutation of the vertices of the final hypergraph, and G_t, the hypergraph to be colored at every time step, is the subhypergraph of G induced by the first t vertices in the permutation.

—In the *dynamic offline model*, the complete sequence $\{G_t\}_{t=1}^n$ is given.

In dynamic online models, the sequence $\{G_t\}_{t=1}^n$ is revealed incrementally, at discrete time steps $t = 1, \ldots, n$, i.e., at time t, G_t is given, and a color for the new vertex v_t must be found without knowledge of future $G_{t'}$, where $t' > t$.

[2]A hypergraph $G = (V', E')$ is an *induced subhypergraph* of $G = (V, E)$ if $V' \subseteq V$ and $E' = \{e \cap V' \mid e \in E\}$. We say that G' is *induced* by V'.

Table I. Number of colors used in deterministic algorithms for intervals

model	lower bound	upper bound
dyn. onl., rel. pos.	$1 + 3\log_5 n$ [this paper]	$O(\log^2 n)$ [Fiat et al. 2005]
dyn. onl., abs. pos.	$1 + 3\log_5 n$ [this paper]	$3\lceil \log_3 n \rceil$ [this paper]
dyn. offline	$1 + 3\log_5 n$ [this paper]	$1 + \log_{3/2} n$ [this paper]
static	$1 + \lfloor \log_2 n \rfloor$ [Even et al. 2003]	$1 + \lfloor \log_2 n \rfloor$ [Even et al. 2003]

Above results scaled to logarithm with base 2 and asymptotically:

model	lower bound	upper bound
dyn. onl., rel. pos.	$1.29 \log_2 n$	$O(\log^2 n)$
dyn. onl., abs. pos.	$1.29 \log_2 n$	$1.89 \log_2 n$
dyn. offline	$1.29 \log_2 n$	$1.71 \log_2 n$
static	$1.00 \log_2 n$	$1.00 \log_2 n$

—In the *dynamic online with absolute positions model*, in addition to $\{G_t\}_{t=1}^n$ being revealed incrementally, the final $G_n = G$ is given from the start as a *vertex-labeled* hypergraph and for each time t, G_t is also given as a vertex-labeled hypergraph, with the induced subhypergraph isomorphism between G_t and G_n *preserving* the labels. This means that the algorithm knows for every new vertex v_t where it is going to lie (i.e., its 'absolute' position) in the final hypergraph G.

—In the *dynamic online with relative positions model*, no information about the final hypergraph G is given (not even its size n). The only information might be the structure of the final hypergraph (for example, when we color points on a line with respect to intervals). Thus, for every new vertex v_t, we only know its 'relative' position with respect to already inserted points, by means of the information in G_t.

From the applications point of view, all models are interesting. In the static model, all base stations are activated at the same time. The dynamic offline model captures a scenario where the order of activation of base stations is known from the start. Additionally, comparison of dynamic online algorithms against algorithms in the dynamic offline model is more fair than against algorithms in the static model. The dynamic online with absolute positions model is also motivated, because revealing absolute positions is not unnatural in many scenarios: One can think of all base stations being at fixed positions, which are known to the algorithm in advance. This means that the algorithm is aware of the final hypergraph that models the situation where all base stations are activated. In the start, no base station is activated. Base stations are constructed or activated one by one, in some order, in response to increasing network traffic, and thus the order of activation is not known from the start. Every new station has to be given a color by the algorithm, such that the conflict-free property is maintained. Finally, the dynamic online with relative positions model is the one with the fewest restrictions on the adversary and can capture a situation where the exact positions of new base stations can not be planned in advance.

In the case of intervals, the four models produce a *hierarchy* of models, in the sense that an adversary in a higher model has more power and an algorithm for a higher model works also for a lower model. A summary of results for deterministic algorithms for conflict-free coloring with respect to intervals is given in table I. All the online algorithms considered so far in the literature work in the *relative positions* model. Our main technical results concern two deterministic algorithms that use $2\lfloor \lg(n+1) \rfloor$ and $3\lceil \log_3 n \rceil \approx 1.89 \lg n$ colors, respectively, in the slightly changed online model of *absolute positions*. We only present here the $3\lceil \log_3 n \rceil$ algorithm and for the other algorithm, we refer the interested reader to [Bar-Noy et al. 2006]. Our deterministic algorithms work against an unrestricted adversary, i.e., they do not assume the (weaker) oblivious adversary. We also exhibit sequences of length $n = 3^k$ that need at least $1 + 2\log_3 n \approx 1.26 \lg n$ colors in any dynamic model, and sequences of length $n = 5^k$ that need at least $1 + 3\log_5 n \approx 1.29 \lg n$ colors in any dynamic model. The above lower bound instances are constructed with the dynamic offline model in mind, but because of the hierarchy, they apply also to the two dynamic online models, as shown in table I. It is not difficult to also prove gaps for the lower bounds between the dynamic offline and the absolute positions models and between the absolute positions and the relative positions models, but they are constant additive, not constant multiplicative (like the gap in the lower bound between the static and the dynamic offline models). For the offline model, we describe an algorithm that uses at most $1 + \log_{3/2} n \approx 1.71 \lg n$ colors in the dynamic offline model. In the relative positions model, we resolve an open problem posed in [Fiat et al. 2005]: We give a tight analysis on the performance of the *first-fit greedy online algorithm* (FF), and prove that it uses $\lfloor n/2 \rfloor + 1$ colors in the worst case.

Finally, we discuss coloring with respect to a specific subset of all intervals. One interesting case is coloring with respect to *rays* (or *halflines*), namely the subset of intervals that contain at least one of the two extreme points. For this case, we show a strong separation between static and dynamic offline models, in the sense that in the static model three colors suffice for any n, whereas in the dynamic offline model $\lfloor \lg n \rfloor + 1$ colors might be necessary. On the other hand, $\lfloor \lg (n-2) \rfloor + 3$ colors suffice even in the relative positions model.

Paper organization. In section 2, we introduce two ways to describe input to dynamic and online algorithms in the case of intervals. In section 3, we consider the dynamic offline model. In section 4, we discuss $O(\log n)$ algorithms in the absolute positions model. In section 5, we analyze the worst-case behavior of the first-fit greedy algorithm. In section 6, we study conflict-free coloring with respect to rays. In section 7, we discuss some of the results and mention open problems.

2. PRELIMINARIES

We show two ways to represent inputs for dynamic models, in the intervals case. We will be using them in subsequent sections.

In the *relative positions model*, the sequence of points inserted can be described by the position in which each new point is inserted, relative to previously inserted points. If $i-1$ points have already been inserted, the i-th point can be inserted in any of i positions described by an integer in the range $[0, i-1]$: 0 is for the new point in the start of the sequence (before any other point), and $k > 0$ is for the new point immediately after the k-th already inserted point.

An insertion sequence of length n is represented by a string of n integers, σ, where $0 \leq \sigma_{(i)} \leq i-1$. If we consider insertion sequences of the same length n ordered lexicographically, then the first and last elements in that order are: $s_n^{\text{first}} = [0,0,0,\dots,0]$, $s_n^{\text{last}} = [0,1,2,\dots,n-1]$. In the relative positions online model, an insertion sequence is revealed from left to right, one by one, to the online algorithm. There are $n!$ possible insertion sequences of length n.

In the *absolute positions model*, initially the algorithm knows the total number of points to be inserted. Then, for each new point the absolute position of that point in the final sequence is revealed to the algorithm. The absolute position can be any number in $\{1,\dots,n\}$ which has not appeared before. Thus, the input to the algorithm is a permutation $\pi \in \mathbf{S}_n$ that is revealed one by one, from left to right.

In the dynamic *offline* setting, the input can be given in either absolute or relative positions, because the two representations are easily convertible to each other if the *whole* sequence is known. For example, the insertion sequence $\sigma = 00121$ (relative positions) corresponds to the permutation $\pi = 51342$ (absolute positions), which means the first point inserted is at the 5th absolute position (rightmost), the second point inserted is at the 1st absolute position (leftmost), and so on.

3. DYNAMIC OFFLINE MODEL

Lower bound. We exhibit insertion sequences that need asymptotically $c \lg n$ colors, where $c > 1$. First, some definitions are needed.

Definition 1. Given a string π of numbers and $x \in \mathbb{N}$, the string $(\pi + x)$ is defined by adding x to each element of π, i.e., $(\pi + x)_{(i)} = \pi_{(i)} + x$, for $i \in \{1,\dots,|\pi|\}$, where $|\pi|$ is the length of π.

We also define a sum-like operator for concatenation of strings:

$$\bigcirc_{i=1}^{p} s_i := s_1 \circ \cdots \circ s_p .$$

Permutations can be viewed as strings of numbers. The following definition proves useful:

Definition 2. Given a permutation π, with $|\pi| = n$, and $k \in \mathbb{N}$, the permutation π^k is defined recursively:

$$\pi^0 = 1 \ ,$$
$$\pi^{k+1} = \bigcirc_{i=1}^{n} \left(\pi^k + (\pi_{(i)} - 1) \cdot n^k\right) \ .$$

It is not difficult to prove that if π is a permutation, then indeed π^k is also a permutation. For example, if $\pi = 132$, then $n = |\pi| = 3$ and $\pi^2 = 132798465$. Notice that π^2 consists of three π-like components (132, 798, 465), that are inserted in the π order. In general, π^k consists of n π^{k-1}-like components that are inserted in the π order. We remark that π^k should not be confused with concatenation of k copies of the string π: $\bigcirc_{i=1}^{k} \pi$.

The idea behind our lower bound proofs is to find a permutation π such that for every $k > 0$, π^k needs y more colors than π^{k-1}, where y is fixed. In the following, we prove two lower bounds. The first is simpler, it is based on $\pi = 132$ (with $y = 2$), and gives a $2\log_3 n + 1 \approx 1.26 \lg n$ lower bound. The second is more elaborate, it is based on $\pi = 15432$ (with $y = 3$), its proof relies on the previous lower bound, and it gives a lower bound of $3\log_5 n + 1 \approx 1.29 \lg n$.

For every k, we exhibit an insertion sequence π^k of absolute positions that has length $n = 3^k$ and needs $2k + 1 = 2\log_3 n + 1$ colors to be conflict-free colored.

Proposition 3. *For $\pi = 132$, input π^k needs at least $2k + 1$ colors in the dynamic model.*

Proof. The proof is by induction. For the base case, $k = 0$, input $\pi^0 = 1$ needs one color.

For the inductive step, assume input π^{k-1} needs at least $2k - 1$ colors. Consider the colorings of the three π^{k-1} components of π^k: A, B, C, in increasing time order of appearance. Since they are inserted in the 132 order, the π^k coloring incrementally looks like: A, then AB, and finally ACB. By the inductive hypothesis each of A, B, C uses at least $2k - 1$ colors.

Assume, for the sake of contradiction, that ACB uses at most $2k$ colors. At least one color that appears in ACB must be unique. If the unique color is in A, then CB uses at most $2k - 1$ colors; if in B, then AC uses at most $2k - 1$ colors; if in C, then AB uses at most $2k - 1$ colors. However, all of CB, AC, AB appear at some point in the coloring of π^k: CB and AC appear in ACB, and AB appears just before the insertion of C. Thus, in each of CB, AC, AB, there must be a unique color. Now, take $J \in \{\mathrm{CB}, \mathrm{AC}, \mathrm{AB}\}$ that is using at most $2k-1$ colors: This J has a unique color among the colors used in J, that can only appear in one of the two π^{k-1} components of J, therefore, the other π^{k-1} component of J is using at most $2k - 2$ colors; a contradiction. □

For every k, we exhibit an insertion sequence π^k of absolute positions that has length $n = 5^k$ and needs $3k + 1 = 3\log_5 n + 1$ colors to be conflict-free colored.

Proposition 4. *For $\pi = 15432$, input π^k needs at least $3k + 1$ colors in the dynamic model.*

Proof. The proof is by induction. For the base case, $k = 0$, input $\pi^0 = 1$ needs one color.

For the inductive step, assume input π^{k-1} needs at least $3k - 2$ colors. Consider the colorings of the five π^{k-1} components of π^k: A, B, C, D, E, in increasing time order of appearance. Since they are inserted in the 15432 order, the π^k coloring incrementally looks like: A, AB, ACB, ADCB, and finally AEDCB. By the inductive hypothesis each of A, B, C, D, E uses at least $3k - 2$ colors.

Assume, for the sake of contradiction, that AEDCB uses at most $3k$ colors. At least one color that appears in AEDCB must be unique. If the unique color is in A, then EDCB uses at most $3k - 1$ colors; if in B, then AED uses at most $3k - 1$ colors; if in C, then AED uses at most $3k - 1$ colors; if in D, then ACB uses at most $3k - 1$ colors; if in E, then ACB uses at most $3k - 1$ colors. However, all of EDCB, AED, ACB appear at some point in the coloring of π^k. For AED, ACB, since the π^{k-1} components are inserted in the 132 order, an argument along the lines of the proof of proposition 3 gives a π^{k-1} component using at most $3k - 3$ colors; a contradiction. For EDCB, the π^{k-1} components are inserted in the 4321 order: One of them has a unique color, which leaves at least two consecutive π^{k-1}

```
$\ell \leftarrow 1$, $V^1 \leftarrow V$, $E^1 \leftarrow E$
while $V^\ell \neq \emptyset$ do:
    $I^\ell \leftarrow$ an independent set of the Delaunay graph of $(V^\ell, E^\ell)$
    color every $v$ in $I^\ell$ with color $\ell$
    $V^{\ell+1} \leftarrow V^\ell \setminus I^\ell$
    $E^{\ell+1} \leftarrow \{e \cap V^{\ell+1} \mid e \in E^\ell\}$
    $\ell \leftarrow \ell + 1$
```

Fig. 3. Algorithm for the dynamic offline model

components using at most $3k-2$ colors. Among these two consecutive components, one of them has a unique color, and thus the other component uses at most $3k-3$ colors; a contradiction. □

It is an open problem whether other permutations can improve on the $3\log_5 n+1$ lower bound. We remark that the above lower bound applies to all three dynamic models.

Upper bound. The dynamic offline case can be viewed as a static problem, because dynamically coloring the sequence $\{G_t\}_{t=1}^n$ is equivalent to statically coloring the hypergraph $G = (V, \bigcup_{t=1}^n E_t)$, where E_t is the hyperedge set of G_t. In [Even et al. 2003], a general framework for conflict-free coloring is presented: The authors provide an algorithm (Algorithm 1 in the paper) that colors the points in iterations. At the ℓ-th iteration, some points are colored with color ℓ and these colored points are not considered in the subsequent iterations. To simplify the presentation, we do not consider singleton hyperedges, since they are conflict-free colored for free. The notion of a Delaunay graph is useful:

Definition 5. The Delaunay graph $G(H)$ of the hypergraph $H = (V, E)$ is the graph (V, D), where the edge set $D \subseteq E$ is the subset of all hyperedges in E of cardinality two.

We set $E = \bigcup_{t=1}^n E_t$, and adapt the framework of [Even et al. 2003] to get the algorithm in figure 3.

Our algorithm in figure 3 is slightly different from the one in [Even et al. 2003]. When the algorithm in [Even et al. 2003] computes $E^{\ell+1}$ from E^ℓ, it does not consider hyperedges for which $|e \cap V^\ell| = 1$, since they are for sure conflict-free colored. On the other hand, our algorithm includes those hyperedges in the next iteration. We made the above choice because the hypergraph (V^ℓ, E^ℓ) arises by inserting the points in V^ℓ in their time order and considering all possible intervals at each time. With this observation, it is easier to argue about the Delaunay graph of (V^ℓ, E^ℓ), and the key to get an efficient conflict-free coloring is to find a big independent set in the Delaunay hypergraph. Also, it is easier to implement our algorithm. The correctness of our algorithm is an immediate consequence of the correctness of the algorithm in [Even et al. 2003], since ours just conflict-free colors more hyperedges. In order to bound the number of colors used, the following lemma is needed:

Lemma 6. *In the case of intervals, the Delaunay graph of* (V^ℓ, E^ℓ) *is* 3*-colorable.*

Proof. Consider the vertices in V^ℓ ordered according to the time of insertion. Each appearing vertex is immediately adjacent to at most two other vertices, and thus it can be colored greedily by one of three given colors. □

Corollary 7. *In a 3-coloring of the Delaunay graph of (V^ℓ, E^ℓ), the largest color class is an independent set of size at least $\lceil |V^\ell|/3 \rceil$.*

Theorem 8. *There is an algorithm that uses at most $\log_{3/2} n + 1$ colors in the dynamic offline model.*

Proof. Apply the algorithm in figure 1, where the independent set chosen at each iteration ℓ is the largest size color class of any 3-coloring of the Delaunay graph of (V^ℓ, E^ℓ). By corollary 7, at each iteration at least $\lceil |V^\ell|/3 \rceil$ are colored. Therefore, the number of iterations (and thus the number of colors) is bounded by $\log_{3/2} n + 1$. □

4. ABSOLUTE POSITIONS MODEL

In this section we present an algorithm that uses $O(\log n)$ colors in the absolute positions model. Roughly speaking, a point p with a unique color in an interval acts as a *separator*: Points to the left of p and points to the right of p can be colored independently, and colors can be freely reused. Our algorithm uses only logarithmic colors by choosing the right points as separators.

In a preliminary version of this paper (see [Bar-Noy et al. 2006]), we presented an algorithm that chooses the separators in each level and uses recursion to independently color the left and the right side of the separators. In this paper we present an algorithm that adopts the opposite approach of coloring two thirds of the points with a greedy non-recursive scheme in each level and of coloring the separators by using the recursion. This algorithm is better in terms of the number of colors it uses in the worst case.

4.1 An asymptotically $3\log_3 n$ algorithm

We provide a recursive algorithm in the absolute positions model that uses $3\lceil \log_3 n \rceil \approx 1.89 \lg n$ colors to color any input of size n. Triples of points with consecutive positions play a major role in the algorithm and this is why we call it the 'triples' algorithm.

To prove the above bound, it suffices to show a method of conflict-free coloring any input of size 3^k with $3k$ colors, because, in that case, if $3^{k-1} < n \le 3^k$ then the algorithm takes the n-sized input, attaches (in any insertion order) $3^k - n$ dummy points to the right of the n points, solves the 3^k-sized instance with the method to get a conflict-free coloring with $3k$ colors, and then it discards the colors of the dummy points to get a conflict-free coloring of the original n points.

If $n = 3^k$, points are colored in k levels that correspond to recursion call levels of the algorithm and each level uses three colors. At each level $\ell \in \{1, \ldots, k\}$, some of the points are colored and the rest are deferred for coloring at a higher level. More precisely at each level ℓ, with $\ell < k$, two thirds of the points are colored in that level and the rest (one third) are deferred. Thus, for each level $\ell < k$ of the recursion, out of the $3^{k+1-\ell}$ points that reach the level, $2 \cdot 3^{k-\ell}$ are colored in that level and $3^{k-\ell}$ are deferred for coloring in a higher level. The final level k is special because all three points that reach it are colored in that level. This situation is shown in table II, where, by convention, level ℓ uses colors $3\ell - 2$, $3\ell - 1$, and 3ℓ.

Table II. Recursion levels of the triples algorithm

recursion level	input size	points colored	points deferred	colors used
1	3^k	$2 \cdot 3^{k-1}$	3^{k-1}	1, 2, 3
...	...	...	...	...
ℓ	$3^{k+1-\ell}$	$2 \cdot 3^{k-\ell}$	$3^{k-\ell}$	$3\ell - 2$, $3\ell - 1$, 3ℓ
...	...	...	...	...
$k-1$	9	6	3	$3k-5$, $3k-4$, $3k-3$
k	3	3	0	$3k-2$, $3k-1$, $3k$

Now, we describe how the algorithm decides at each level which points to color and which to defer. At each level ℓ, with $\ell < k$, the algorithm partitions the points in triples, according to their absolute positions: The three leftmost points are in the first triple, the second three leftmost points are in the second triple, and so on, until the final triple which contains the three rightmost points. The decision at each level ℓ is made as follows:

> For every triple, the first point that is requested to be colored in the triple is deferred for coloring in a higher level, whereas the other two points are colored at level ℓ.

Also, for $\ell < k$, the input at level ℓ, which we denote by $\pi_{\langle \ell \rangle}$, induces an input $\pi_{\langle \ell+1 \rangle}$ at level $\ell + 1$ as follows: The absolute positions of triples at level ℓ give the absolute positions of points and points at level $\ell + 1$ are requested in the same order as the first points of triples at level ℓ. Initially, the input at level 1, i.e., $\pi_{\langle 1 \rangle}$, is set equal to the original input π. For example, consider the input $\pi = 923745618$, revealed to the online algorithm, one by one element, from left to right. In order to exhibit better how the algorithm runs, we take the inverse permutation of π, which maps absolute positions of points to the time they are requested:

$$\pi^{-1} = \pi^{-1}_{\langle 1 \rangle} = \; 8\mathbf{2}3\ \mathbf{5}67\ 49\mathbf{1}\ .$$

This is the input for level 1 (as denoted by the subscript) and we have also highlighted the first (i.e., earliest) point requested in every triple. Input π induces the following input for level 2: $\pi^{-1}_{\langle 2 \rangle} = 231$, or $\pi_{\langle 2 \rangle} = 312$.

The triples algorithm relies on first-fit greedy coloring, so the following lemma, which is a special case of lemma 16, proves useful:

Lemma 9. *Any conflict-free coloring of $x < 4$ points, can be extended to a conflict-free coloring of $x + 1$ points, with at most three colors, for any position of the $x + 1$-th point, by using the first-fit greedy coloring scheme.*

Proof. If the coloring of x points is using less than three colors, then the greedy scheme introduces at most one new color. If the coloring of x points is using three colors, then it must be the case that $x = 3$ and the coloring looks like abc. In that case, the first-fit greedy scheme can color any new point by reusing the minimum color among the colors of non-adjacent points to the new point. □

Now, we explain how the algorithm colors points in a specific level. If a new point p is requested that is decided to be colored in level ℓ (i.e., not deferred for coloring in a higher level), the algorithm finds the set of all points P already colored at level ℓ with the following property: p' is in P, if there is no point deferred for

$$\begin{array}{rccccccccc}
\pi_{\langle 1\rangle}^{-1} = & 8 & 2 & 3 & 5 & 6 & 7 & 4 & 9 & 1 \\
C_{\langle 1\rangle} = & 1 & * & 1 & * & 1 & 3 & 2 & 1 & * \\
\\
\pi_{\langle 2\rangle}^{-1} = & & 2 & & 3 & & & & & 1 \\
C_{\langle 2\rangle} = & & 5 & & 6 & & & & & 4 \\
\\
C = & 1 & 5 & 1 & 6 & 1 & 3 & 2 & 1 & 4
\end{array}$$

Fig. 4. A run of the triples algorithm

coloring in a higher level between p and p'. It can be proved that P contains at most three points. We defer the proof of the above statement, and include it later in the proof of the correctness of the algorithm, in proposition 10. The algorithm chooses the color of p using a greedy coloring scheme, i.e., by choosing the minimum color possible among $3\ell-2$, $3\ell-1$, and 3ℓ, so that the interval containing $P \cup \{p\}$ remains conflict-free (we proved in lemma 9 that this is always possible). The coloring at each level ℓ, corresponding to input $\pi_{\langle \ell\rangle}$, is denoted by $C_{\langle \ell\rangle}$; it is a partial coloring for $\ell < k$, because only two thirds of the points are colored.

The run of the algorithm on the example input mentioned above is shown in figure 4. Each '$*$' denotes a point that is to be colored in a higher level and C denotes the final coloring.

For example, the point colored at $t = 4$ gets color 2 (at level 1), because it is not the first point that appeared in its triple, and because there is a point requested at $t = 3$ which is colored with color 1.

The correctness of the algorithm is immediate from the following result:

PROPOSITION 10. *At any time $t \in \{1, \ldots, n\}$, in any interval I of points, there is a point in I colored with a unique color in I. Moreover, a uniquely colored point can always be found among the points that were colored in the deepest recursive level of points in I.*

PROOF. Recall that the three leftmost points are in the first triple, the second three leftmost points are in the second triple, and so on, until the final triple which contains the three rightmost points. At some time t, we say that a triple is *empty* if no point of it has yet been requested to be colored. At any time t, take any interval I and consider the points in I that were colored at the highest level ℓ. If $\ell = k$, then these are at most three points in I and by lemma 9 there is a unique color in I. If $\ell < k$, then I can not span a whole non-empty triple in level ℓ, because it will include a point colored at a level higher than ℓ. Also, I can not span parts of more than two non-empty triples, because then it would span a whole triple between the two extreme triples. Thus, I spans at most two triples and at most two points in each of them (see figure 5). By the description of the algorithm these points are colored by a greedy scheme with colors from $3\ell - 2$, $3\ell - 1$, and 3ℓ, and thus by lemma 9 there is a unique color in I. □

We have thus established the following theorem:

THEOREM 11. *There is a deterministic algorithm that uses at most $3\lceil \log_3 n\rceil$ colors in the dynamic online absolute positions model.*

$$[\,\ast \quad \underbrace{\cdot \quad \cdot\,] \overbrace{\qquad\qquad\qquad\qquad}^{\text{possible empty triples}} [\,\cdot \quad \cdot}_{\text{interval } I} \quad \ast\,]$$

Fig. 5. At most four points spanned at level $\ell < k$ by I

5. RELATIVE POSITIONS MODEL

In this section, we analyze an algorithm in the relative positions model. The *first-fit greedy* algorithm (FF) for online conflict-free coloring for intervals, mentioned in [Fiat et al. 2005], works as follows: For the next point to color, it chooses the minimum color that maintains the conflict-free coloring property. For example, the first-fit greedy algorithm colors insertion sequence $\sigma = 010322$ ($\pi = 251643$ in absolute positions) as follows: [.1....], [.1..2.], [21..2.], [21..23], [21.323], [214323]. We have the following tight result for FF:

Theorem 12. *For $n \geq 2$, the worst-case number of colors used by FF for inputs of length n is $\lceil n/2 \rceil + 1$.*

In order to establish the above result, we prove a lower bound for FF and a matching upper bound.

Lower bound for FF. There are sequences which force the FF algorithm to use $O(n)$ colors: We prove that sequence $00(20)^i1$, of length $2i+3$, uses $i+3$ colors. The following lemma is needed:

Lemma 13. *Insertion sequence $00(20)^i$ uses $i+2$ colors and the two leftmost colors in the coloring are $i+2$, $i+1$. The third and fourth leftmost colors, in case $i > 0$ are i, $i+1$.*

Proof. By induction. Base case ($i = 0$ and 1): Insertion sequence 00 gives the coloring 21 and insertion sequence 0020 gives the coloring 3212.

For the inductive step, by the inductive hypothesis, the coloring for $i > 0$ is:

$$c^i = \quad i+2 \quad i+1 \quad i \quad i+1 \quad \ldots$$

The next insertion (at relative position 2) is between $i+1$ and i. The new point can not get color i or $i+1$, because there are adjacent points colored with those colors. It has to get color $i+2$, because if it would get another smaller color, then this color could also be used as the color occurring in the leftmost point of c^i. The coloring becomes:

$$i+2 \quad i+1 \quad i+2 \quad i \quad i+1 \quad \ldots$$

The next insertion (at relative position 0) can not get a color among $i+2$, $i+1$, i. It can not get any other already used color, because then this color could also be used as the color occurring in the leftmost position of c^i. So, a new color has to be used and thus:

$$c^{i+1} = \quad i+3 \quad i+2 \quad i+1 \quad i+2 \quad \ldots \quad \square$$

Lemma 14. *Insertion sequence $00(20)^i1$ uses $i+3$ colors.*

PROOF. From lemma 13, the coloring of $00(20)^i$ is:

$$i+2 \quad i+1 \quad \ldots$$

The insertion (at relative position 1) between colors $i+2$ and $i+1$ has to get a new color, because any other color, except $i+2$ which is in any case impossible (because of adjacency), would be chosen by the FF algorithm when coloring $00(20)^i$, in the last insertion. □

This establishes the following lower bound on the number of colors used by FF:

PROPOSITION 15. *For every $n \geq 2$, there are insertions sequences of length n that force the first-fit greedy algorithm to use $\lceil n/2 \rceil + 1$ colors.*

Upper bound for FF. In order to prove an upper bound on the number of colors used by the FF algorithm, we consider *uniquely occurring colors* in a coloring.

LEMMA 16. *In any FF coloring there are at most three distinct colors with the following property: each of these colors occurs exactly once.*

PROOF. Assume the three colors x, y, z that occur uniquely in a coloring by FF:

$$\ldots x \ldots y \ldots z \ldots$$

If the new point is inserted to the left of y, then z is an eligible color. If the new point is inserted to the right of y, then x is an eligible color. In any case, FF will introduce no new color for the new point, since FF chooses the minimal eligible color. □

We remark that we can have 1, 2, or 3 uniquely occurring colors in a coloring by FF, as exhibited by the insertion sequence $\sigma = 011$ which is colored as 132 by FF.

PROPOSITION 17. *For $n \geq 2$: No insertion sequence of length n forces the first-fit greedy algorithm to use more than $\lceil n/2 \rceil + 1$ colors.*

PROOF. Consider a coloring by FF of n points. If k colors are used and u of them occur uniquely, then $k-u$ colors occur at least as duplicates, and $n \geq 2(k-u)+u$, which gives $2k \leq n+u$ and since k is integer:

$$k \leq \left\lfloor \frac{n+u}{2} \right\rfloor \leq \left\lfloor \frac{n+3}{2} \right\rfloor = \left\lceil \frac{n}{2} \right\rceil + 1 ,$$

because $u \leq 3$ (by lemma 16). □

Theorem 12 follows immediately from propositions 15 and 17.

Remark 18. The upper bound technique for FF can also be applied to the *unique max* algorithm (UM), a simple algorithm that is used as a component in other more elaborate algorithms, including the $O(\log^2 n)$ algorithm of [Fiat et al. 2005]. Our technique gives an upper bound of $\lceil n/2 \rceil + 2$ for the number of colors used by UM. However, in contrast to the tight analysis for FF, only insertion sequences that force UM to use $\Theta(\sqrt{n})$ colors have been found (see [Fiat et al. 2005]).

6. COLORING WITH RESPECT TO A SUBSET OF THE SET OF ALL INTERVALS

We relax the conflict-free coloring problem of intervals as follows. Instead of the requirement that *all* intervals need to have a uniquely colored point, it is required that the conflict-free condition holds only for intervals in a specific *subset* of the set of all intervals. Examples are coloring with respect to all intervals of a specific length, say k, or all intervals of length up to k. Recently, the problem has been also studied by Katz et al. [2007].

Another interesting case arises from the intervals that contain either of the two extreme points. Equivalently, these are intervals that are defined by halflines (infinite intervals), or *rays*. We therefore refer to the problem as conflict-free coloring with respect to *rays*. The motivation for considering this restricted subset comes from agents whose movement range is not strictly inside the line segment between the two extreme points. We also want to point out how different are the results and the gaps between models, related to the all intervals case.

For n points there are $2n-1$ ray defined intervals, of which n contain the leftmost point and are called *prefix* intervals and n contain the rightmost point and are called *suffix* intervals (the interval containing all points is both a prefix and a suffix interval).

In the static model, the coloring $133 \ldots 332$ (i.e., color the extreme points with unique colors and use the same color for all non-extreme points) suffices for all n and uses three colors. It is not hard to see that three colors are required for $n \geq 4$ (for $n = 3$ the coloring 121 with two colors is a conflict-free coloring).

To analyze the problem in the dynamic models, we consider first coloring with respect to *prefix intervals* only (the suffix case has the same bounds, because it is symmetric). In the static model for prefixes, the coloring $122 \ldots 22$ is a conflict-free coloring with 2 colors. Obviously, this coloring is optimal. In the dynamic models for prefixes, we will first prove a lower bound of $1 + \lfloor \lg n \rfloor$ already for the dynamic offline model and then provide an algorithm using $2 + \lfloor \lg(n-1) \rfloor$ colors already in the relative position model.

Proposition 19. *In the dynamic (offline) model, input $\sigma = 0^n$ needs $1 + \lfloor \lg n \rfloor$ colors to be conflict-free colored with respect to prefixes.*

Proof. The i-th point inserted is always at the left of all previously inserted points and thus contributes i new intervals. In fact, by viewing the dynamic problem as a static problem (as we did in the upper bound discussion of section 3), it can be proved that coloring 0^n with respect to prefixes is equivalent to coloring n points statically with respect to (all) intervals. Thus, at least $1 + \lfloor \lg n \rfloor$ colors are needed. $\square$

We propose the following algorithm for coloring prefixes: The algorithm colors differently

(a) points that appear to the left of all previously inserted points,

(b) points that appear to the right of at least one previously inserted point.

The first group of points contains points for which $\sigma_{(i)} = 0$, and the second group points for which $\sigma_{(i)} > 0$. Therefore, it is possible to distinguish between the two groups even in the relative positions model. Points in the first group are colored

Table III. Number of colors used in deterministic algorithms for rays ($n \geq 3$)

model	lower bound	upper bound
all dynamic	$1 + \lfloor \lg n \rfloor$	$3 + \lfloor \lg(n-2) \rfloor$
static	3	3

according to the static coloring for intervals: $\ldots 41213121$. Points in the second group are all colored with the same color, which is different from the colors used in the first group. For example, input $\sigma = 010120020$ is colored as $131{\star}2{\star}{\star}1{\star}$, where '$\star$' is the color used for points in the second group. It is not difficult to see that subsets of points which must have the conflict-free property never contain a '$\star$'-colored point as their leftmost point. Thus, every subset of points that must be conflict-free colored contains a point of the first group. Additionally, every such subset contains points with consecutive positions in the first group, and is therefore conflict-free colored. If the input is 0^n, exactly $1+\lfloor \lg n \rfloor$ colors are used. Otherwise, at least one point is '$\star$'-colored and at most $n-1$ points are in the first group, which implies that at most $2 + \lfloor \lg(n-1) \rfloor$ are used. We have just proved the following:

PROPOSITION 20. *For $n \geq 2$, there is an algorithm that conflict-free colors with respect to prefixes any insertion sequence σ (in the relative positions model) with at most $2 + \lfloor \lg(n-1) \rfloor$ colors.*

Finally, we use the upper bound for prefixes (and suffixes) to prove an upper bound for rays. We claim that for dynamically coloring with respect to rays, one more color than the prefix (or suffix) case suffices. The idea is to use a unique color for the first point p inserted, and then color independently points to the left of p from points to the right of p: color whatever is inserted to the left of p with respect to prefixes and whatever is inserted to the right of p with respect to suffixes. From the above, it is not hard to prove:

PROPOSITION 21. *For $n \geq 3$, there is an algorithm that conflict-free colors with respect to rays any insertion sequence σ (in the relative positions model) with at most $3 + \lfloor \lg(n-2) \rfloor$ colors.*

The above analysis gives a separation between static and dynamic models for coloring with respect to rays: The number of colors used is a logarithmic factor apart. All the results are shown in table III. This is in contrast with the all-intervals case in which the separation result between static and dynamic offline model is weaker, just a constant factor apart, $1 + \lg n$ and $1 + \log_{3/2} n$ colors used, respectively.

7. DISCUSSION AND OPEN PROBLEMS

We introduced a hierarchy of models for conflict-free coloring ranging from a completely static model (weakest adversary model) to a fully online model (strongest adversary model). We concentrated on deterministic conflict-free coloring with respect to intervals. For this special case, we proposed algorithms for some of the models and gave upper bounds on their worst-case performance. We also provided lower bounds on the number of colors used in some models.

There are still gaps between lower and upper bounds (see table I). For example, in the dynamic offline model the lower bound is $1 + 2\log_3 n \approx 1.26 \lg n$, whereas the upper bound is $1 + \log_{3/2} n \approx 1.71 \lg n$, a constant factor apart. The situation is similar in the absolute positions model where the upper bound is approximately $1.89 \lg n$. The most important open problem is narrowing the gap between lower and upper bound in the relative positions model: $\Omega(\log n)$, and $O(\log^2 n)$, respectively, which are a logarithmic factor apart.

So far in the literature, only the static and the fully online (relative positions) models had been considered. If there is a gap on the number of colors used between these two extreme models, the hierarchy can help pin-point exactly where the 'jump' occurs, and thus give a better understanding of the problem. In the case of all-intervals, static uses $O(\log n)$ and the best known online deterministic algorithm $O(\log^2 n)$ colors, but this logarithmic factor 'jump' is not a result of the online model, because it occurs just between the absolute positions model and the fully online (relative positions) model. However, in the rays case, a logarithmic factor jump occurs between the static and the dynamic offline model.

In the dynamic online absolute positions setting, it is natural for the algorithm to know the total number n of points that will be inserted from the start. The triples algorithm exploits that knowledge to achieve $O(\log n)$ colorings. However, in the triples algorithm, for final size of input $n = 3^k$, the adversary can request the first k points in such a way such that the algorithm uses k different colors. An open problem in the absolute positions model is to maintain an $O(\log k)$ coloring after the first k points have been inserted for all k with $0 < k \leq n$.

Finally, the hierarchy of models is not constrained to problems for points on the real line. It can be used in conflict-free coloring for hypergraphs, in general. A possible use of the hierarchy would be to understand better conflict-free coloring problems in the plane, or even higher dimensions.

ACKNOWLEDGMENTS

We would like to thank János Pach and David Peleg for helpful discussions concerning the problems studied in this paper.

REFERENCES

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

Alon, N. and Smorodinsky, S. 2006. Conflict-free colorings of shallow discs. In *Proceedings of the 22nd Annual ACM Symposium on Computational Geometry (SoCG)*. 41–43.

Bar-Noy, A., Cheilaris, P., Olonetsky, S., and Smorodinsky, S. 2007. Online conflict-free coloring for geometric hypergraphs. In *Proceedings of the 34th International Colloquium on Automata, Languages and Programming (ICALP)*. 219–230.

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

Borodin, A. and El-Yaniv, R. 1998. *Online computation and competitive analysis*. Cambridge University Press.

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

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

Chen, K., Kaplan, H., and Sharir, M. 2006. Online conflict free coloring for halfplanes, congruent disks, and axis-parallel rectangles. Manuscript.

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

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

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

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

Iyer, A. V., Ratliff, H. R., and Vijayan, G. 1988. Optimal node ranking of trees. *Information Processing Letters 28*, 225–229.

Katchalski, M., McCuaig, W., and Seager, S. 1995. Ordered colourings. *Discrete Mathematics 142*, 141–154.

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

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

Smorodinsky, S. 2003. Combinatorial problems in computational geometry. Ph.D. thesis, School of Computer Science, Tel-Aviv University.

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

Received Month Year; revised Month Year; accepted Month Year