Brief Announcement: Weakening the Online Adversary Just Enough to get Optimal Conflict-free Colorings for Intervals

Amotz Bar-Noy* Brooklyn College
Panagiotis Cheilaris† CUNY and NTUA
Svetlana Olonetsky‡ Tel-Aviv University
Shakhar Smorodinsky§ Courant Institute, NYU

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

General Terms: Algorithms, Theory.

Keywords: Online algorithms, cellular networks, frequency assignment, conflict free, coloring.

A *hypergraph* $H = (V, \mathcal{E})$ is a generalization of a graph for which *hyperedges* (elements of $\mathcal{E}$) can be arbitrary-sized non-empty subsets of the vertex set V. A vertex coloring $C\colon V \to \mathbb{N}^+$ of hypergraph H is called *conflict-free* if in every hyperedge e there is a vertex whose color is unique among all other colors in the hyperedge. The above problem models a frequency assignment problem for cellular networks. A cellular network consists of fixed-position base stations and moving agents. The study of conflict-free colorings was originated in the work of Even et al. [4] and Smorodinsky [6]. 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.

Chen et al. [3] 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 $\mathcal{E}$ consists of all subsets of V defined by intervals intersecting at least one vertex. 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. Also, conflict-free coloring for intervals plays a role in the study of conflict-free coloring for more complicated range spaces (see [4]). The *static* version of the problem, where the n points are to be colored simultaneously, is solved optimally in [4] with $\lfloor \lg n \rfloor + 1$ colors.

*E-mail: *amotz@sci.brooklyn.cuny.edu*. Supported by the CUNY Collaborative Incentive Research Grants Program Round 11 (2004–2006).

†E-mail: *philaris@sci.brooklyn.cuny.edu*. Supported by the European Social Fund (75%) and National Resources (25%) under the program ΕΠΕΑΕΚ II, 'Heraclitus'.

‡E-mail: *olonetsk@post.tau.ac.il*.

§E-mail: *shakhar@cims.nyu.edu*. Supported by NSF Mathematical Sciences Postdoctoral Fellowship award 0402492.

SPAA'07, June 9–11, 2007, San Diego, California, USA.
ACM 978-1-59593-667-7/07/0006.

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. This is an online setting, in which the algorithm has no knowledge of how vertices will be requested in the future. For this version of the problem, in the case of intervals, the best known deterministic algorithm is from [3] and uses $O(\log^2 n)$ colors in the worst case. That algorithm requires $\Omega(log^2 n)$ colors on some inputs. Recently, randomized algorithms that use $O(\log n)$ colors with high probability have been obtained ([3, 1]). All of these 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 without knowing the random bits that the algorithm is going to use and the *expected* number of colors is analyzed. The randomized model can be seen as a relaxation of the strict deterministic model: some power is taken from the adversary, or equivalently given to the algorithm, in order to use just a logarithmic number of colors. Another such relaxation, introduced in [2], is to give extra information to the algorithm about where each requested point will end up in the final coloring (the 'absolute positions' model). In the absolute positions model, the best algorithm in [2] uses $3\lceil \log_3 n \rceil \approx 1.89 \lg n$ colors. Other such relaxations are given in [2] (coloring with respect to rays) and [5] (online ranking of paths). In this paper we introduce yet another relaxation, the *recoloring model*, in which the algorithm is allowed to recolor some of the points. An interesting question is to come up with $O(\log n)$ algorithms that rely as little as possible on their extra power (as few random bits as possible, as few recolorings as possible). Towards that goal, in a unified framework, we provide the best known results: a randomized algorithms that expectedly uses a logarithmic number of random bits, and a recoloring algorithm that recolors at most one point per request.

Related to applications, the recoloring model is very motivated: The frequency spectrum is quite expensive, so a solution which strictly uses a logarithmic number of colors is desirable. On the other hand excessive recoloring is not desirable, because if a base station is given another color there is a disruption of service for all agents connected to it. Our algorithm uses at most $\log_{3/2} t + 1$ colors for the first t points, for every t with $1 \leq t \leq n$, and for every new request of a point it recolors at most one of the old points.

An online coloring framework.

First, some basic definitions are needed. Let $H = (V, \mathcal{E})$ be a hypergraph. For a subset $V' \subset V$ let $H(V')$ be the hypergraph $(V', \mathcal{E}')$ where $\mathcal{E}' = \{e \cap V' | e \in \mathcal{E}\}$. $H(V')$ is called the *induced hypergraph* on V'. For a hypergraph $H = (V, \mathcal{E})$, the *Delaunay graph* $G(H)$ is the simple graph $G = (V, E)$ where the edge set E is defined as $E = \{(x, y) \mid \{x, y\} \in \mathcal{E}\}$ (i.e., G is the graph on the vertex set V whose edges consist of all hyperedges in H of cardinality two).

Let $H = (V, \mathcal{E})$ be a hypergraph induced by intervals. Our goal is to define a framework that colors the vertices V in an online fashion. That is, the vertices of V are revealed by an adversary one at a time. At each time step t, the algorithm must assign a color to the newly revealed vertex v_t. The coloring has to be conflict-free for all the induced hypergraphs $H(V_t)$ $t = 1, \ldots, n$, where $V_t \subseteq V$ is the set of vertices revealed by time t. Let $A = \{a, b, c\}$ be a set of auxiliary colors (not to be confused with the set of 'real' colors used for the CF-coloring: $\{1, 2, \ldots\}$). Let $f : \mathbb{N} \to A$ be some fixed function. We now define the framework that depends on the choice of the function f. A table (to be updated online) is maintained where each entry ℓ at time t is associated with a subset $V_t^\ell \subset V_t$ in addition to an auxiliary proper coloring of $G(H(V_t^\ell))$ (using colors in A). We say that $f(\ell)$ is the color that represents entry ℓ in the table. At the beginning all entries of the table are empty. Suppose all entries of the table are updated until time $t-1$ and let v_t be the vertex revealed by the adversary at time t. The framework first assigns an auxiliary color to v_t such that the auxiliary coloring of V_{t-1}^1 together with the color of v_t is a proper coloring of $G(H(V_{t-1}^1 \cup \{v_t\}))$. This is always possible, because the Delaunay graph $G(H(V_t^\ell))$ is 3-colorable (easy proof). Any (proper) coloring procedure can be used by the framework. For example a first-fit greedy procedure in which all colors in the order a, b, c are checked until one is found. If this color is the same as $f(1)$ (the auxiliary color that is associated with entry 1), then the final color in the online CF-coloring of v_t is 1 and the updating process for the t-th vertex stops. Otherwise, if an auxiliary color cannot be found or if the assigned auxiliary color is not the same as the color associated with this entry, the updating process continues to the next entry. The updating process stops at the first entry ℓ for which v_t is both added to V_t^ℓ and the auxiliary color assigned to v_t is the same as $f(\ell)$. The color of v_t in the final conflict-free coloring is then set to ℓ.

The 3-colorability of the Delaunay graph $G(H(V_t^\ell))$ implies that $G(H(V_t^\ell))$ has an independent set of size at least $\lceil |V_t^\ell|/3 \rceil$. Intuitively, an algorithm that chooses a large independent set in each entry uses few colors in total. The above framework was used in an offline setting in [2]. In that case, the algorithm chooses f after it sees the whole input. In the following, we show how the above framework can be adapted to a randomized and a recoloring setting.

A randomized $O(\log n)$ algorithm.

If the choice of $f(\ell)$ is uniformly random for every ℓ, at each level, the expected value of the size of the independent set with color $f(\ell)$ is $|V_t^\ell|/3$ and it is not difficult to prove that this algorithm uses with high probability at most $\log_{3/2} n + 1$ colors. In contrast with the algorithm in [3] which uses a linear number of random bits, our algorithm uses a logarithmic number of bits with high probability. In [3] the expected number of colors is bounded by $\log_{8/7} n + 1 \approx 5.19 \log_2 n$, three times our bound of $\log_{3/2} n + 1 \approx 1.71 \log_2 n$.

An $O(\log n)$ algorithm with recoloring.

We describe a deterministic online CF-coloring algorithm for intervals that is only allowed to recolor a single old point during each request of a new point. The algorithm uses two auxiliary colors: a and d (points colored with d correspond to ones colored with b, c above; we use only d in the presentation for simplicity). All points that get to the entry ℓ and are assigned a get real color at entry ℓ (similar to $f(\ell) = a$). Points that are assigned d get their real color at a higher entry. In order to have logarithmic number of entries (and thus total colors) the size of the independent set colored with a has to be large. To achieve this goal, our algorithm maintains the following invariant in every level: For every point colored with d, there exists an adjacent point colored with a. Therefore, at least a third of the points that get to each entry get color a, and two thirds are deferred for coloring in a higher entry. Again, the colors used in total are at most $\log_{3/2} n + 1$; the number of recolorings is at most $n - (\lfloor \lg n \rfloor + 1)$ and this is tight. When a new point p arrives, it is colored according to the following algorithm, starting from entry 1:

- We color point p with a if this does not generate two consecutive a's. In this case, the point gets a real color at this entry.
- We color point p with d if it does not generate a point colored with d that will not have an adjacent point colored with a. Note that here p is deferred for real coloring at a higher entry and thus the same algorithm is recursively applied at the next entry.
- It remains to handle the case where the new point p has a point colored with a on one side and a point, say q, colored with d on the other side, such that q has no adjacent point colored with a. We color p with d in the current entry and the real color of q, and we recolor q with a, and thus we recolor q with the real color of the current entry.

References.

[1] Amotz Bar-Noy, Panagiotis Cheilaris, Svetlana Olonetsky, and Shakhar Smorodinsky. Randomized online conflict-free coloring for hypergraphs. Manuscript, 2006.

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

[3] Ke Chen, Amos Fiat, Haim Kaplan, Meital Levy, Jiří Matoušek, Elchanan Mossel, János Pach, Micha Sharir, Shakhar Smorodinsky, Uli Wagner, and Emo Welzl. Online conflict-free coloring for intervals. *SIAM Journal on Computing*, 36(5):956–973, 2006.

[4] Guy Even, Zvi Lotker, Dana Ron, and Shakhar Smorodinsky. Conflict-free colorings of simple geometric regions with applications to frequency assignment in cellular networks. *SIAM Journal on Computing*, 33:94–136, 2003.

[5] Ingo Schiermeyer, Zsolt Tuza, and Margit Voigt. On-line rankings of graphs. *Discrete Mathematics*, 212(1–2):141–147, 2000.

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