Conflict-free coloring with respect to a subset of intervals

Panagiotis Cheilaris[*]   Shakhar Smorodinsky[†]

Abstract

Given a hypergraph $H = (V, \mathcal{E})$, a coloring of its vertices is said to be conflict-free if for every hyperedge $S \in \mathcal{E}$ there is at least one vertex in S whose color is distinct from the colors of all other vertices in S. The discrete interval hypergraph H_n is the hypergraph with vertex set $\{1, \ldots, n\}$ and hyperedge set the family of all subsets of consecutive integers in $\{1, \ldots, n\}$. We provide a polynomial time algorithm for conflict-free coloring any subhypergraph of H_n, we show that the algorithm has approximation ratio 2, and we prove that our analysis is tight, i.e., there is a subhypergraph for which the algorithm computes a solution which uses twice the number of colors of the optimal solution. We also show that the problem of deciding whether a given subhypergraph of H_n can be colored with at most k colors has a quasipolynomial time algorithm.

1   Introduction

A hypergraph H is a pair $(V, \mathcal{E})$, where V is a finite set and $\mathcal{E}$ is a family of non-empty subsets of V. We denote by $\mathbb{Z}^+$ the set of positive integers and by $\mathbb{N}$ the set of non-negative integers.

Definition 1.1. Let $H = (V, \mathcal{E})$ be a hypergraph and let C be a coloring $C \colon V \to \mathbb{Z}^+$: We say that C is a *conflict-free coloring* (cf-coloring in short) if for every hyperedge $e \in \mathcal{E}$ there exists a color $i \in \mathbb{Z}^+$ such that $|e \cap C^{-1}(i)| = 1$. That is, every hyperedge $e \in \mathcal{E}$ contains some vertex whose color is unique in e.

The study of cf-coloring was initiated in the work of Even et al. [10] and of Smorodinsky [17] and was extended in numerous other works (c.f., [1, 2, 3, 5, 6, 7, 8, 11, 13, 14, 15]). The study was initially motivated by its application to frequency assignment for cellular networks. A cellular network consists of two kinds of nodes: *base stations* and *mobile clients*. Base stations have fixed positions, modeled by a finite set of points in the plane, and provide the backbone of the network. Every base station emits at a fixed frequency. If a client wants to establish a link with a base station it has to tune itself to this base station's frequency. Clients, however, can be in the range of many different base stations. To avoid interference, the system must assign frequencies to base stations in the following way: For any closed disk d in the plane (representing the communication range of a client located at the center of this disk), there must be at least one base station which is contained in d and has a frequency that is not used by any other base station contained in d. Since frequencies are limited and costly, a scheme that reuses frequencies, where possible, is desirable.

Here is a more general, formal definition: Let P be a set of n points in the plane and let $\mathcal{R}$ be a family of regions in the plane (e.g., all closed discs). We denote by $H = H_\mathcal{R}(P)$ the hypergraph on the set P whose hyperedges are all subsets P' that can be cut off from P by a region in $\mathcal{R}$. That

---

[*]Department of Informatics, Università della Svizzera italiana, 6900 Lugano, Switzerland. `cheilarp@usi.ch`

[†]Mathematics department, Ben-Gurion University, Be'er Sheva 84105, Israel. `shakhar@math.bgu.ac.il`

is, all subsets P' such that there exists some region $r \in \mathcal{R}$ with $r \cap P = P'$. We refer to such a hypergraph as the hypergraph *induced by P with respect to $\mathcal{R}$.*

Now, consider the hypergraph induced by a set of n *collinear* points with respect to the family of closed disks in the plane. It is not difficult to see that this hypergraph is isomorphic to the hypergraph induced by a set of n real numbers with respect to the family of closed intervals, which is also isomorphic to the following discrete interval hypergraph.

Definition 1.2. Let $[n] = \{1, \ldots, n\}$. For $s \leq t$, $s, t \in [n]$, we define the (discrete) interval $[s,t] = \{i \in [n] \mid s \leq i \leq t\}$. The *discrete interval hypergraph H_n* has vertex set $[n]$ and hyperedge set $\mathcal{I}_n = \{[s,t] \mid s \leq t,\ s, t \in [n]\}$.

It is not difficult to prove that $\lfloor \log_2 n \rfloor + 1$ colors are necessary and sufficient in order to cf-color H_n (see, e.g., [10]). An online variation of this cf-coloring problem in which vertices appear one by one and the algorithm has to commit to a color for each point as soon as it appears, maintaining the conflict-free property of the point set at every time, was introduced in [6] and further studied in [4].

In this paper, we are interested in cf-coloring subhypergraphs of H_n of the following form: $H = ([n], I)$, where $I \subseteq \mathcal{I}_n$. Then, H is a hypergraph induced by n points on the real line with respect to a *subset* of all possible intervals. Cf-colorings of such hypergraphs were studied in the online setting in [4]. Katz et al., in [12], claim a 4-approximation polynomial time cf-coloring for any such hypergraph H (in the offline setting). Studying cf-coloring for subhypergraphs of geometric hypergraphs can be justified by applications where only a given subset of the hyperedge set is required to have the conflict-free property.

In section 2, we describe an algorithm for computing cf-colorings for general hypergraphs, based on hitting sets. In section 3, we show how the above algorithm and an appropriate choice of the hitting set can give a 2-approximation polynomial time algorithm for cf-coloring a subhypergraph of the discrete interval hypergraph, improving on the 4-approximation algorithm of Katz et al. In section 4, we show that the above analysis is tight, i.e., there are subhypergraphs of H_n for which the algorithm computes a cf-coloring with twice the optimal (minimum) number of colors. In section 5, we show that the decision problem whether a given subhypergraph of H_n can be cf-colored with at most k colors has a quasipolynomial time algorithm; this implies that this decision problem is probably not NP-complete.

2 A hitting-set algorithm for conflict-free coloring

In this section, we present an algorithm for conflict-free coloring a hypergraph. It is based on repeatedly computing a minimal hitting set in hypergraphs.

Definition 2.1. A *hitting set* of a hypergraph $H = (V, \mathcal{E})$ is a subset $S \subseteq V$ such that for every $e \in \mathcal{E}$ there exists some $v \in S$ with $v \in e$. A hitting set S is *minimal* if for every $v \in S$, $S \setminus \{v\}$ is not a hitting set.

In the literature, a conflict-free coloring is an assignment of colors (positive integers) to the vertices of the hypergraph. In this work, we introduce and consider a slight variation of conflict-free coloring, in which we allow some vertices to not be assigned colors, as long as in every hyperedge, there exists a vertex with assigned color that is uniquely occurring in the hyperedge. In other words, we allow the coloring function $C \colon V \to \mathbb{Z}^+$ in definition 1.1 to be a partial function. Alternatively, we can use a special color '0' given to vertices that are not assigned any positive color and obtain a total function $C \colon V \to \mathbb{N}$. Then, we arrive at the following variant of definition 1.1.

Definition 2.2. Let $H = (V, \mathcal{E})$ be a hypergraph and let $C \colon V \to \mathbb{N}$: We say that C is a *conflict-free coloring* if for every hyperedge $S \in \mathcal{E}$ there exists a color $i \in \mathbb{Z}^+$ such that $|S \cap C^{-1}(i)| = 1$. We denote by $\chi_{\mathrm{cf}}(H)$ the minimum integer k for which H admits a cf-coloring with colors in $\{0, \ldots, k\}$.

Remark 2.3. We claim that this variation of conflict-free coloring, with the partial coloring function or the placeholder color '0', is interesting from the point of view of applications. As mentioned in section 1, vertices model base stations in a cellular network. A vertex with no positive color assigned to it can model a situation where a base station is not activated at all, and therefore the base station does not consume energy. One can also think of a bi-criteria optimization problem where a conflict-free assignment of frequencies has to be found with small number of frequencies (in order to conserve the frequency spectrum) and few activated base stations (in order to conserve energy).

We describe algorithm 1 for conflict-free coloring any hypergraph $H = (V, \mathcal{E})$.

Algorithm 1 A hitting set algorithm for conflict-free coloring $H = (V, \mathcal{E})$

$\quad \ell \leftarrow 0; V^0 \leftarrow V; \mathcal{E}^0 \leftarrow \mathcal{E}$
$\quad$ **while** $\mathcal{E}^\ell \neq \emptyset$ **do**
$\quad\quad S^\ell \leftarrow$ a minimal hitting set for $(V^\ell, \mathcal{E}^\ell)$
$\quad\quad$ color every $v \in V^\ell \setminus S^\ell$ with color ℓ
$\quad\quad V^{\ell+1} \leftarrow S^\ell$
$\quad\quad \mathcal{E}^{\ell+1} \leftarrow \{e \cap S^\ell \mid e \in \mathcal{E}^\ell \text{ and } |e \cap S^\ell| > 1\}$
$\quad\quad \ell \leftarrow \ell + 1$
$\quad$ **end while**
$\quad$ **if** $V^\ell \neq \emptyset$ **then** color every $v \in V^\ell$ with color ℓ **end if**

Lemma 2.4. *Algorithm 1 terminates.*

Proof. At every iteration of the loop, there is some hyperedge $e \in \mathcal{E}^\ell$ for which $|e \cap S^\ell| = 1$. This follows from the minimality of S^ℓ. Thus, $|\mathcal{E}^\ell| > |\mathcal{E}^{\ell+1}|$. Therefore, the number of hyperedges decreases at every iteration of the loop, and necessarily reaches zero after a finite number of iterations of the loop. $\qquad\square$

Lemma 2.5. *Algorithm 1 produces a conflict-free coloring.*

Proof. We first show that for every hyperedge $e \in \mathcal{E}$, there is some ℓ for which $|e \cap S^\ell| = 1$. Notice that for every iteration $i > 0$, we have $S^{i-1} \supseteq S^i$. If $|e \cap S^0| > 1$, consider the maximum i for which $|e \cap S^i| > 1$. Then, hyperedge $e \cap S^i = e \cap V^{i+1}$ belongs to $\mathcal{E}^{i+1}$ and has to be hit by S^{i+1}, i.e., $(e \cap S^i) \cap S^{i+1} = e \cap S^{i+1}$ is non-empty and thus $|e \cap S^{i+1}| = 1$, because of the maximality of i.

Let v be the one element of $e \cap S^\ell$. Vertex v is colored with some color greater than ℓ by the algorithm and all other vertices of e are colored with colors which are at most of value ℓ. Thus, e has the conflict-free property. $\qquad\square$

3 A 2-approximation algorithm for a set of intervals

We use algorithm 1, described in the previous section, to conflict-free color a subhypergraph of H_n which is comprised of a given subset $I \subseteq \mathcal{I}_n$ of intervals. It is necessary to specify how to compute the minimal hitting set.

The minimal hitting set S is computed as follows (in fact, we compute a minimum cardinality hitting set, but we do not need this stronger fact):

First, we compute a special independent set of intervals $F \subseteq I$ (i.e., in F no two intervals have a common vertex). We compute this independent set F of intervals incrementally. Initially, there is nothing in the independent set. We scan vertices from 1 to n and we include in the independent set the interval $[i, j] \in I$ with minimum j such that $[i, j]$ does not intersect anything already in the independent set. After computing F, for every interval $[i, j] \in F$, we take in S the vertex j (i.e., the maximum or rightmost vertex).

Lemma 3.1. *S is a minimal hitting set.*

Proof. Set S is a hitting set because no interval is completely contained between two vertices in S, no interval ends before the first interval in F, and no interval starts after the last interval in F; otherwise such intervals would be chosen in the independent set F. Set S is minimal, because removing any element j of it, means that the interval with right endpoint j in F is not hit any more. □

Remark 3.2. The computation of the maximal (in fact maximum) independent set of intervals given above is also known as a solution to the activity selection problem. See for example [9, section 16.1].

Notice that the time complexity of the algorithm is $O(n \log n)$: We sort the intervals according to their right endpoints. Then, at every iteration of the loop we can choose the hitting set in linear time. There is at most a logarithmic number of iterations of the loop, because $\chi_{\text{cf}}(H) \leq \chi_{\text{cf}}(H_n) = \lfloor \log_2 n \rfloor + 1$.

We intend to compare colorings produced by the above algorithm with optimal colorings. We define recursively the following families of sets of intervals of $\mathbb{Z}^+$.

Definition 3.3. Family $\mathcal{J}_1$ exactly contains all singleton sets of intervals. For $k > 1$, set of intervals I is in family $\mathcal{J}_k$ if and only if it can be expressed as a union $I = L \cup R \cup \{\iota\}$, where both $L, R \in \mathcal{J}_{k-1}$, no interval from L has a common point with an interval from R, and interval ι includes every interval in L and every interval in R.

We refer to a set of intervals in family $\mathcal{J}_k$ as a $\mathcal{J}_k$ *configuration*.

Lemma 3.4. *Any conflict-free coloring uses at least k colors for a set of intervals that is a superset of a $\mathcal{J}_k$ configuration.*

Proof. We use induction on k. For $k = 1$, the statement is trivially true. Assume it is true for k, we will prove it for $k+1$. Assume, for the sake of contradiction, that there is a conflict-free coloring C with just k colors of a set of intervals I' that is a superset of a $\mathcal{J}_{k+1}$ configuration I. Then, by definition of $\mathcal{J}_{k+1}$, $I = L \cup R \cup \{\iota\}$, where both $L, R \in \mathcal{J}_k$, no interval from L has a common point with an interval from R, and interval ι includes every interval in L and every interval in R. By the inductive hypothesis, the points contained in intervals of L use k colors and also the points contained in intervals of R use k colors. The above two pointsets are disjoint and the interval ι includes both pointsets. As a result, ι is not conflict-free colored, which is a contradiction. □

We are now ready to bound the approximation ratio of the proposed algorithm.

Theorem 3.5. *The conflict-free coloring algorithm for hypergraphs with respect to a subset of intervals is a 2-approximation algorithm.*

4

Proof. It is enough to prove that if some hyperedge (or interval), say ι, reaches iteration with $\ell = k - 1$ of the loop (i.e., the algorithm uses at least k colors), then the input contains as a subset a $\mathcal{J}_{\lceil k/2 \rceil}$ configuration and moreover this configuration is entirely contained in ι.

We prove it by induction. For $k = 1, 2$, it is true, because there is at least one interval in the input, and therefore at least one non-zero color is needed in any optimal coloring. For $k > 2$, assume there is a vertex v that gets color k. Then at iteration with $\ell = k - 1$ of the loop there is an interval ι with its rightmost vertex being $v \in S^\ell$ (see figure 1).
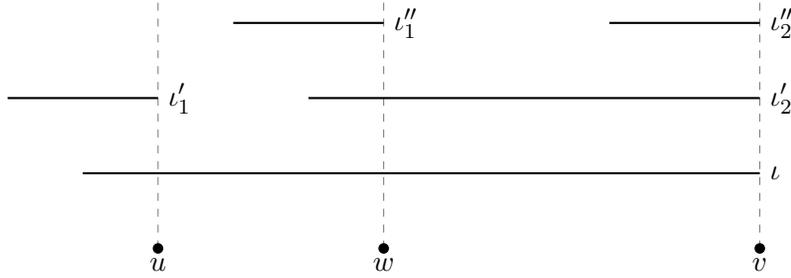

Figure 1: Intervals in an input using k colors

Since ι was not removed in the previous iteration $\ell - 1$, there were two vertices of ι in $S^{\ell-1}$, say u and v, with $u < v$. Also, since u and v are in $S^{\ell-1}$ there are two intervals with them as right endpoints in the independent set computed at iteration $\ell - 1$, say ι'_1 and ι'_2. Since ι'_2 was not removed in the iteration $\ell - 2$, there were two vertices of ι'_2 in $S^{\ell-2}$, say w and v, with $u < w < v$. Also, since u, w, and v are in $S^{\ell-2}$ there are three intervals with them as right endpoints in the independent set computed at iteration $\ell - 2$; call ι''_1 the one ending at w and ι''_2 the one ending at v. Since the three intervals are independent, ι''_1 and ι''_2 start after u, therefore they are fully contained in ι (which contains u). By the inductive hypothesis, since each of ι''_1, ι''_2 reach iteration $\ell - 2$, each of them entirely contains a $\mathcal{J}_{\lceil (k-2)/2 \rceil}$ configuration, and, since ι''_1 and ι''_2 are disjoint, together with ι they constitute a $\mathcal{J}_{\lceil k/2 \rceil}$ configuration. $\square$

4 A tight instance for the 2-approximation algorithm

For $k \geq 2$, we intend to define an input I_k that is a tight instance for the approximation algorithm, i.e., an instance that forces the algorithm to use at least twice the number of colors in an optimal coloring. Before doing that, we define some notation that will prove useful.

Definition 4.1. Given a set of intervals I and a natural number d, we define I^{+d} to be the set of intervals, where all intervals of I are shifted d to the right, i.e.,

$$I^{+d} = \{[i + d, j + d] \mid [i, j] \in I\}.$$

Definition 4.2. Given a set of intervals I, we define the *length* of I, denoted $\text{len}(I)$ to be the rightmost point occurring in any of the intervals of I minus the leftmost point occurring in any of the intervals of I plus one.

Now, we are ready to proceed with the definition of the tight instance.

Definition 4.3. For $k = 2$ the input I_2 has length equal to four and consists of three intervals.

$$I_2 = \{[1, 2], [3, 3], [2, 4]\}$$

5

For $k > 2$ the input is defined recursively as follows.

$$I_{k+1} = I_k \cup I_k^{+\text{len}(I_k)} \cup \{[\text{len}(I_k) - k + 1, 2\text{len}(I_k) + 1]\}$$

Abusing notation, we call the I_k component the *left* I_k part of I_{k+1} and the $I_k^{+\text{len}(I_k)}$ component the *right* I_k part of I_{k+1}. These left and right parts are disjoint. Input I_4 is shown in figure 2. Moreover, in the figure, under the vertices of the input we give the coloring produced by the 2-approximation algorithm and then an optimal conflict-free coloring.
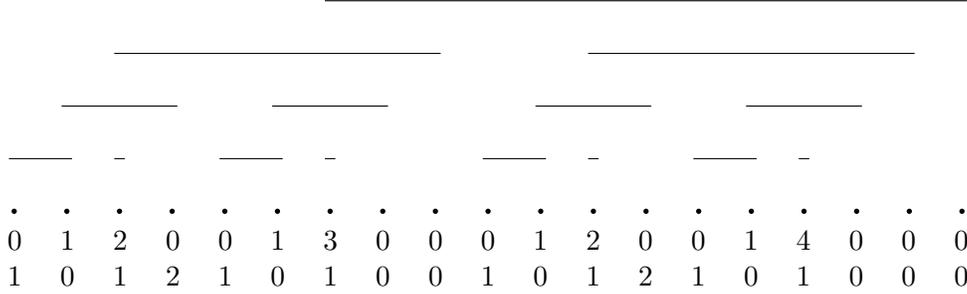

0	1	2	0	0	1	3	0	0	0	1	2	0	0	1	4	0	0	0
1	0	1	2	1	0	1	0	0	1	0	1	2	1	0	1	0	0	0

Figure 2: Input I_4, algorithm cf-coloring, and optimal cf-coloring

It is not difficult to see that the length of the instance satisfies the recurrence relation

$$\text{len}(I_{k+1}) = 2\text{len}(I_k) + 1, \tag{1}$$

which implies, since $\text{len}(I_2) = 4$, that $\text{len}(I_k) = 5 \cdot 2^{k-2} - 1$.

Another notion that will prove useful is the *level* of each interval in the above instance that we define in the following.

Definition 4.4. In input I_2, intervals $[1, 2]$ and $[3, 3]$ are of level 1 and interval $[2, 4]$ is of level 2. In the recursively defined instance

$$I_{k+1} = I_k \cup I_k^{+\text{len}(I_k)} \cup \{[\text{len}(I_k) - k + 1, 2\text{len}(I_k) + 1]\}$$

the intervals of the I_k part have the same levels as the corresponding intervals in the I_k instance, the intervals of the $I_k^{+\text{len}(I_k)}$ part have the same levels as the corresponding intervals of the I_k instance before the '$+\text{len}(I_k)$' operation, and interval $[\text{len}(I_k) - k + 1, 2\text{len}(I_k) + 1]$ has level $k + 1$.

In fact, in figure 2 the vertical coordinate of each interval signifies its level, with higher intervals having higher level.

Lemma 4.5. *For $k \geq 3$, in I_k, the leftmost point of the level k interval is the same as the rightmost level 1 interval in the left I_{k-1} part of I_k.*

Proof. We prove by induction that the rightmost level 1 interval of the left I_{k-1} part of I_k is at position $\text{len}(I_{k-1}) - (k-1) + 1$. For I_3, the rightmost level 1 interval of the left I_2 part of I_3 consists of point $4 - (3 - 1) + 1 = 3$. By the inductive hypothesis, the rightmost level 1 interval of the left I_{k-1} part of I_k is at $\text{len}(I_{k-1}) - (k-1) + 1$. Then for I_{k+1}, the rightmost level 1 interval of its left I_k part is at

$$\text{len}(I_{k-1}) - (k-1) + 1 + \text{len}(I_{k-1}) = (2\text{len}(I_{k-1}) + 1) + k - 1 = \text{len}(I_k) + k - 1.$$

The last equality is implied by equation (1). $\qquad\square$

Lemma 4.6. *Instance I_k contains a $\mathcal{J}_{\lceil k/2 \rceil}$ configuration as a subset.*

Proof. By induction. It is true for $k = 2$ and $k = 3$, because I_2 contains a $\mathcal{J}_1$ configuration and I_3 contains a $\mathcal{J}_2$ configuration. For $k > 3$, in instance I_k, the interval of level k contains completely a copy of I_{k-1}, in which two disjoint copies of I_{k-2} are contained. By the inductive hypothesis, in each copy of I_{k-2}, a $\mathcal{J}_{\lceil (k-2)/2 \rceil}$ configuration is contained. These two disjoint $\mathcal{J}_{\lceil (k-2)/2 \rceil}$ configurations, together with the level k interval constitute a $\mathcal{J}_{\lceil k/2 \rceil}$ configuration in I_k. $\square$

Lemma 4.7. *There is a conflict-free coloring of I_k with $\lceil k/2 \rceil$ colors.*

Proof. We define recursively a coloring of I_k that uses $\lceil k/2 \rceil$ colors and we prove by induction that it is conflict-free.

For $k = 2$ the coloring is 1010, which can be easily checked to be conflict-free.

If k is odd, take a coloring of I_{k-1} and in its rightmost position use color $\lceil k/2 \rceil$, concatenate a coloring of I_{k-1}, and then concatenate color '0'. By induction, the left I_{k-1} part is conflict-free because we started with a conflict-free coloring and we introduced a new color $\lceil k/2 \rceil$, the right I_{k-1} part is conflict-free because it is colored with a conflict-free coloring. The level k interval is conflict-free because of color $\lceil k/2 \rceil$ that occurs uniquely.

If k is even, with $k > 2$, take a coloring of I_{k-1}, concatenate a coloring of I_{k-1}, and then concatenate color '0'. By induction, the left I_{k-1} part is conflict-free because it is colored with a conflict-free coloring, the right I_{k-1} part is conflict-free because it is colored with a conflict-free coloring. The level k interval is conflict-free because of color $\lceil k/2 \rceil$ that occurs in the right I_{k-1} part and because its leftmost point, by lemma 4.5, is to the right of the $\lceil k/2 \rceil$ color occurring in the left I_{k-2} part of the left I_{k-1} part. $\square$

Corollary 4.8. *An optimal coloring of I_k uses $\lceil k/2 \rceil$ colors.*

We now describe a family of hypergraphs that arise after the first iteration of the while loop of the 2-approximation algorithm, if the initial input is I_k.

Definition 4.9. The instance L_0 is on one vertex, namely the vertex set is $\{1\}$, and contains no interval, i.e, $L_0 = \{\}$. The length of instance L_0 is defined to be 1. For $k > 0$, L_{k+1} is defined recursively, as follows.

$$L_{k+1} = L_k \cup L_k^{+\mathrm{len}(L_k)} \cup \{[\mathrm{len}(L_k), 2\mathrm{len}(L_k)]\}$$

It is not difficult to see that the length satisfies the recurrence relation $\mathrm{len}(L_{k+1}) = 2\mathrm{len}(L_k)$, which implies $\mathrm{len}(L_k) = 2^k$. We say that L_{k+1} consists of a left L_k part, a right L_k part, and the interval $[2^k, 2^{k+1}]$.

Proposition 4.10. *The 2-approximation algorithm colors I_k with k colors.*

Proof. Assume input I_k is given to the 2-approximation algorithm. In the iteration of the while loop where the algorithm colors points with color ℓ ($\ell = 0, 1, \dots$), the algorithm considers a hypergraph H_ℓ. We will prove that the algorithm considers the hypergraphs

$$H_0 = I_k, H_1 = L_{k-1}, \dots, H_{k-1} = L_1, H_k = L_0,$$

and then it terminates, i.e., it uses k colors. We say that H_i is *followed* by H_{i+1}, to show that two hypergraphs H_i, H_{i+1} are considered successively by the algorithm, in that order.

First, we prove that for every $k \geq 2$, I_k is followed by L_{k-1}, by induction on k. It is not difficult to see that, when I_k is considered, the independent set of intervals chosen consists of all

level 1 intervals of I_k and the hitting set that is chosen consists of the right endpoints of all level 1 intervals of I_k (a formal proof can be carried out by induction on k). For $k = 2$ it is not difficult to check that I_2 is followed by L_1. For $k > 2$, I_k consists of a left I_{k-1} part which induces a left L_{k-2} part and a right I_k part, which induces a right L_{k-2} part (we use the inductive hypothesis). From lemma 4.5, the leftmost point of the level k interval is the same as the rightmost level 1 interval in the left I_{k-1} part of I_k, and therefore the level k interval induces an interval that starts from the last point of the left L_{k-2} part of the hypergraph that follows I_k and ends at the last point of the right L_{k-2} part of the hypergraph that follows I_k. To summarize, the I_k is followed by a left L_{k-2} part, a right L_{k-2} part and interval $[2^{k-2}, 2^{k-1}]$, i.e., it is L_{k-1}.

Then, we prove that for $k > 0$, L_k is followed by L_{k-1}, by induction on k. For $k = 1$, it is not difficult to see that for L_1 the interval $[1, 2]$ is chosen and its right endpoint, i.e., 2, makes up the hitting set. Then, easily, L_1 is followed by L_0. For $k > 1$, when L_k is considered, the independent set of intervals that is chosen consists of the intervals of length two of the left L_{k-1} part

$$\{[1, 2], [3, 4], \dots, [2^{k-1} - 1, 2^{k-1}]\}$$

and the intervals of length two of the right L_{k-1} part

$$\{[2^{k-1} + 1, 2^{k-1} + 2], [2^{k-1} + 3, 2^{k-1} + 4], \dots, [2^k - 1, 2^k]\}.$$

Therefore the hitting set is

$$\{2, 4, \dots 2^{k-1}\} \cup \{2^{k-1} + 2, 2^{k-1} + 4, \dots, 2^k\} = \{i \colon \text{odd} \mid 2 \leq i \leq 2^k\}$$

and consists of 2^{k-1} elements. By induction, after removal of the points of the hitting set, the left L_{k-1} part induces a L_{k-2} part, and the right L_{k-1} part induces a L_{k-2} part. The interval $[2^{k-1}, 2^k]$ of L_k contains all points in $\{2^{k-1} + 2, 2^{k-1} + 4, \dots, 2^k\}$ of the right L_{k-1} part and just point 2^{k-1} of the left L_{k-1} part, and therefore induces $[2^{k-2}, 2^{k-1}]$ in the hypergraph that follows L_k. To summarize, the L_k is followed by a left L_{k-2} part, a right L_{k-2} part and interval $[2^{k-2}, 2^{k-1}]$, i.e., it is L_{k-1}.

Finally, we prove that when L_0 is reached, no hypergraph follows, and the algorithm terminates. This is true, because L_0 contains no interval (hyperedge). $\qquad\blacksquare$

Remark 4.11. From the above proof of proposition 4.10, it is immediate that if L_k is given as an input to the 2-approximation algorithm, the following sequence of hypergraphs

$$H_0 = L_k, H_1 = L_{k-1}, \dots, H_{k-1} = L_1, H_k = L_0$$

is considered in the iterations of the while loop. Moreover, it can also be proved, with a proof similar to those of lemmata 4.6 and 4.7, that an optimal coloring for L_k uses $\lceil k/2 \rceil$ colors. Therefore, the family of instances L_k is also a family of tight instances for the 2-approximation algorithm. However, the family of instances I_k has the additional property that no two intervals in it share a common right endpoint.

5 A quasipolynomial time algorithm

Consider the decision problem CFSUBSETINTERVALS:

> "Given a subhypergraph $H = ([n], I)$ of the discrete interval hypergraph H_n and a natural number k, is it true that $\chi_{\mathrm{cf}}(H) \leq k$?"

Notice that the above problem is non-trivial only when $k < \lfloor \log_2 n \rfloor + 1$; if $k \geq \lfloor \log_2 n \rfloor + 1$ the answer is always yes, since $\chi_{\text{cf}}(H_n) = \lfloor \log_2 n \rfloor + 1$.

Algorithm 2 is a non-deterministic algorithm for CFSUBSETINTERVALS.

The algorithm scans points from 1 to n, tries non-deterministically every color in $\{0, \ldots, k\}$ at the current point and checks if all intervals in I ending at the current point have the conflict-free property. If some interval in I has not the conflict-free property under a non-deterministic assignment, the algorithm answers 'no'. If all intervals in I have the conflict-free property under some non-deterministic assignment, the algorithm answers 'yes'.

We check if an interval in I that ends at the current point, say t, has the conflict-free property in the following space-efficient way. For every color c in $\{0, \ldots, k\}$, we keep track of:

(a) the closest point to t colored with c in variable p_c and

(b) the second closest point to t colored with c in variable s_c.

Then, color c is occurring exactly one time in $[j, t] \in I$ if and only if $s_c < j \leq p_c$.

Algorithm 2 A non-deterministic algorithm deciding whether $\chi_{\text{cf}}(H) \leq k$ for $H = ([n], I)$
_____
  for $c \leftarrow 0$ to k **do**
    $s_c \leftarrow 0$
    $p_c \leftarrow 0$
  end for
  for $t \leftarrow 1$ to n **do**
    choose c non-deterministically from $\{0, \ldots, k\}$
    $s_c \leftarrow p_c$
    $p_c \leftarrow t$
    for $j \in \{j \mid [j, t] \in I\}$ **do**
      IntervalConflict $\leftarrow$ True
      for $c \leftarrow 1$ to k **do**
        if $s_c < j \leq p_c$ **then**
          IntervalConflict $\leftarrow$ False
        end if
      end for
      if IntervalConflict **then**
        return NO
      end if
    end for
  end for
  return YES
_____

Lemma 5.1. *The space complexity of algorithm 2 is $O(\log^2 n)$.*

Proof. Since $k = O(\log n)$ and each point position can be encoded with $O(\log n)$ bits, the arrays p and s (indexed by color) take space $O(\log^2 n)$. All other variables in the algorithm can be implemented in $O(\log n)$ space. Therefore the above non-deterministic algorithm has space complexity $O(\log^2 n)$. □

Corollary 5.2. CFSUBSETINTERVALS *has a quasipolynomial time deterministic algorithm.*

Proof. By standard computational complexity theory arguments (see, e.g., [16]), we can transform algorithm 2 to a deterministic algorithm solving the same problem with time complexity $2^{O(\log^2 n)}$, i.e., CFSUBSETINTERVALS has a quasipolynomial time deterministic algorithm. $\square$

6 Discussion and open problems

The exact complexity of computing an optimal cf-coloring for a subhypergraph of the discrete interval hypergraph remains an open problem. We have provided a 2-approximation algorithm. One might try to improve the approximation ratio, find a polynomial time approximation scheme, or even find a polynomial time exact algorithm. The last possibility is supported by the fact that the decision version of the problem, CFSUBSETINTERVALS, is unlikely to be NP-complete, unless NP-complete problems have quasipolynomial time algorithms.

It would also be interesting to study the complexity of computing optimal conflict-free colorings for subhypergraphs of other geometric hypergraphs, like the hypergraph induced by a set of n points in the plane with respect to a given set of closed disks in the plane.

Finally, we introduced a slightly different cf-coloring function $C\colon V \to \mathbb{N}$, for which vertices colored with '0' can not act as uniquely-colored vertices in a hyperedge. Naturally, one could try to study the bi-criteria optimization problem, in which there two minimization goals: (a) the number of colors used, $\max_{v \in V} C(v)$ (minimization of frequency spectrum use) and (b) the number of vertices with positive colors, $|\{v \in V \mid C(v) > 0\}|$ (minimization of activated base stations).

Acknowledgments.

References

[1] D. Ajwani, K. Elbassioni, S. Govindarajan, and S. Ray. Conflict-free coloring for rectangle ranges using $\tilde{O}(n^{.382+\epsilon})$ colors. In *Proc. 19th ACM Symp. on Parallelism in Algorithms and Architectures (SPAA)*, pages 181–187, 2007.

[2] N. Alon and S. Smorodinsky. Conflict-free colorings of shallow discs. *Internat. J. Comput. Geom. Appl.*, 18(6):599–604, 2008.

[3] A. Bar-Noy, P. Cheilaris, S. Olonetsky, and S. Smorodinsky. Online conflict-free colouring for hypergraphs. *Combin. Probab. Comput.*, 19:493–516, 2010.

[4] A. Bar-Noy, P. Cheilaris, and S. Smorodinsky. Deterministic conflict-free coloring for intervals: from offline to online. *ACM Transactions on Algorithms*, 4(4):44.1–44.18, 2008.

[5] P. Cheilaris. *Conflict-free coloring.* PhD thesis, City University of New York, 2009.

[6] K. Chen, 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. *SIAM J. Comput.*, 36:545–554, 2006.

[7] K. Chen, H. Kaplan, and M. Sharir. Online conflict free coloring for halfplanes, congruent disks, and axis-parallel rectangles. *ACM Transactions on Algorithms*, 5(2):16.1–16.24, 2009.

[8] X. Chen, J. Pach, M. Szegedy, and G. Tardos. Delaunay graphs of point sets in the plane with respect to axis-parallel rectangles. *Random Struct. Algorithms*, 34(1):11–23, 2009.

[9] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. MIT Press, 2nd edition, 2001.

[10] 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.*, 33:94–136, 2003.

[11] S. Har-Peled and S. Smorodinsky. Conflict-free coloring of points and simple regions in the plane. *Discrete Comput. Geom.*, 34(1):47–70, 2005.

[12] M.J. Katz, N. Lev-Tov, and G. Morgenstern. Conflict-free coloring of points on a line with respect to a set of intervals. In *CCCG '07: Proc. 19th Canadian Conference on Computational Geometry*, 2007.

[13] N. Lev-Tov and D. Peleg. Conflict-free coloring of unit disks. *Discrete Appl. Math.*, 157(7):1521–1532, 2009.

[14] J. Pach and G. Tardos. Conflict-free colourings of graphs and hypergraphs. *Combin. Probab. Comput.*, 18(5):819–834, 2009.

[15] J. Pach and G. Tóth. Conflict free colorings. *Discrete & Computational Geometry, The Goodman-Pollack Festschrift*, pages 665–671, 2003.

[16] C. Papadimitriou. *Computational Complexity*. Addison Wesley, 1993.

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