Fully-Dynamic and Kinetic Conflict-Free Coloring of Intervals with Respect to Points*

Mark de Berg[1], Tim Leijsen[2], Aleksandar Markovic[3], André van Renssen[†4], Marcel Roeloffzen[‡5], and Gerhard Woeginger[6]

1 **TU Eindhoven, Eindhoven, The Netherlands**
mdberg@win.tue.nl
2 **TU Eindhoven, Eindhoven, The Netherlands**
3 **TU Eindhoven, Eindhoven, The Netherlands**
a.markovic@tue.nl
4 **National Institute of Informatics, Tokyo and JST, ERATO, Kawarabayashi Large Graph Project, Japan**
andre@nii.ac.jp
5 **National Institute of Informatics, Tokyo and JST, ERATO, Kawarabayashi Large Graph Project, Japan**
marcel@nii.ac.jp
6 **TU Eindhoven, Eindhoven, The Netherlands**
g.woeginger@tue.nl

Abstract

We introduce the fully-dynamic conflict-free coloring problem for a set S of intervals in $\mathbb{R}^1$ with respect to points, where the goal is to maintain a conflict-free coloring for S under insertions and deletions. A coloring is conflict-free if for each point p contained in some interval, p is contained in an interval whose color is not shared with any other interval containing p. We investigate trade-offs between the number of colors used and the number of intervals that are recolored upon insertion or deletion of an interval. Our results include:

- a lower bound on the number of recolorings as a function of the number of colors, which implies that with $O(1)$ recolorings per update the worst-case number of colors is $\Omega(\log n/\log\log n)$, and that any strategy using $O(1/\varepsilon)$ colors needs $\Omega(\varepsilon n^{\varepsilon})$ recolorings;
- a coloring strategy that uses $O(\log n)$ colors at the cost of $O(\log n)$ recolorings, and another strategy that uses $O(1/\varepsilon)$ colors at the cost of $O(n^{\varepsilon}/\varepsilon)$ recolorings;
- stronger upper and lower bounds for special cases.

We also consider the kinetic setting where the intervals move continuously (but there are no insertions or deletions); here we show how to maintain a coloring with only four colors at the cost of three recolorings per event and show this is tight.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases Conflict-free colorings, Dynamic data structures, Kinetic data structures

Digital Object Identifier 10.4230/LIPIcs.ISAAC.2017.26

* A full version of the paper is available at https://arxiv.org/abs/1701.03388.
† A. v. R. and was supported by JST ERATO Grant Number JPMJER1201, Japan.
‡ M. R. was supported by JST ERATO Grant Number JPMJER1201, Japan.

1 Introduction

Consider a set S of fixed base stations that can be used for communication by mobile clients. Each base station has a transmission range, and a client can potentially communicate via that base station when it lies within the transmission range. However, when a client is within reach of several base stations that use the same frequency, the signals will interfere. Hence, the frequencies of the base stations should be assigned in such a way that this problem does not arise. Moreover, the number of used frequencies should not be too large. Even *et al.* [10] and Smorodinsky [14] introduced conflict-free colorings to model this problem, as follows. Let S be a set of disks in the plane, and for a point $q \in \mathbb{R}^2$ let $S(q) \subseteq S$ denote the set of disks containing the point q. A coloring of the disks in S is *conflict-free* if, for any point $q \in \mathbb{R}^2$ with non-empty $S(q)$, the set $S(q)$ has at least one disk with a color that is unique among the disks in $S(q)$. Even *et al.* [10] proved that any set of n disks in the plane admits a conflict-free coloring with $O(\log n)$ colors, and this bound is tight in the worst case.

The concept of conflict-free colorings can be generalized and extended in several ways, giving rise to a host of challenging problems. Below we mention some of them; for lack of space we only discuss the papers most directly related to our work. A more extensive overview is given by Smorodinsky [15]. One obvious generalization is to work with types of regions other than disks. For instance, Even *et al.* [10] showed how to find a coloring with $O(\log n)$ colors for a set of translations of any single centrally symmetric polygon. Har-Peled and Smorodinsky [12] extended this result to regions with near-linear union complexity. One can also consider the dual setting, where one wants to color a given set P of n points in the plane, such that any disk – or rectangle, or other range from a given family – contains at least one point with a unique color (if it contains any point at all). This too was studied by Even *et al.* [10] and they show that this can be done with $O(\log n)$ colors when the ranges are disks or scaled translations of a single centrally symmetric convex polygon.

The results mentioned above deal with the static setting, in which the set of objects to be colored is known in advance. This may not always be the case, leading Fiat *et al.* [11] to introduce the *online* version of the conflict-free coloring problem. Here the objects to be colored arrive one at a time, and each object must be colored upon arrival. Fiat *et al.* show that when coloring points in the plane with respect to disks, n colors may be needed in the online version. Hence, they turn their attention to the 1-dimensional problem of online coloring points with respect to intervals. They prove that this can be done deterministically with $O(\log^2 n)$ colors and randomized with $O(\log n \log\log n)$ colors with high probability. Later Chen [8] gave a randomized algorithm that uses $O(\log n)$ colors with high probability. In the same paper, similar results were obtained for conflict-free colorings of points with respect to halfplanes, unit disks and axis-aligned rectangles of almost the same size. In these cases the colorings use $O(\text{polylog } n)$ colors with high probability. Bar-Noy, Cheilaris, and Smorodinsky [3] discussed several versions of the deterministic one-dimensional variant. Furthermore, Abam *et al.* [1] studied the dual version of coloring intervals on a line with respect to points. Later, Bar-Noy *et al.* [2] considered the case where recolorings are allowed for each insertion. They prove that for coloring points in the plane with respect to halfplanes, one can obtain a coloring with $O(\log n)$ colors in an online setting at the cost of $O(n)$ recolorings in total. More recent variants include strong conflict-free colorings [7, 13], where we require several unique colors, and conflict-free multicolorings [4], which allow assigning multiple colors to a point. Even more variants of online conflict-free colorings can be found in the survey [15].

Our contributions. We introduce a variant of the conflict-free coloring problem where the objects to be colored arrive and disappear over time. This *fully-dynamic conflict-free coloring problem* models a scenario where new base stations may be deployed (to deal with increased capacity demands, for example) and existing base stations may break down or be taken out of service (either permanently or temporarily). We also define the *semi-dynamic conflict-free coloring problem* as the online variant where recolorings are allowed (or the fully-dynamic variant without deletions). Note that when we talk about the *dynamic* variant, we mean *fully-dynamic*. These natural variants have, to the best of our knowledge, not been considered so far. It is easy to see that, unless one maintains a coloring in which any two intersecting objects have distinct colors, there is always a sequence of deletions that invalidates a given conflict-free coloring. Hence, recolorings are needed to ensure that the new coloring is conflict-free. This leads to the question: how many recolorings are needed to maintain a coloring with a certain number of colors? We initiate the study of fully-dynamic conflict-free colorings by considering the problem of coloring intervals with respect to points. In this variant, we are given a (dynamic) set S of intervals in $\mathbb{R}^1$, which we want to color such that for any point $q \in \mathbb{R}^1$ the set $S(q)$ of intervals containing q contains an interval with a unique color. This version of the problem can be used to model the case where the base stations are located along a highway, for instance, and 1-dimensional range and frequency assignment problems have already been studied in various settings [2, 7, 11]. Moreover, the lower bounds that we prove hold for the 2-dimensional problem as well. In the static setting, coloring intervals is rather easy: a simple procedure yields a conflict-free coloring with three colors. The dynamic version turns out to be much more challenging.

In Section 2 we prove lower bounds on the possible tradeoffs between the number of colors used and the worst-case number of recolorings per update: for any algorithm that maintains a conflict-free coloring on a sequence of n insertions of intervals with at most $c(n)$ colors and at most $r(n)$ recolorings per insertion, we must have $r(n) > n^{1/(c(n)+1)}/(8c(n))$. This implies that for $O(1/\varepsilon)$ colors we need $\Omega(\varepsilon n^{\varepsilon})$ recolorings per updated, and with only $O(1)$ recolorings per update we must use $\Omega(\log n/\log\log n)$ colors.

In Section 3 we then present several algorithms that achieve bounds close to our lower bound. All bounds are worst-case, unless specifically stated otherwise. First, we present two algorithms for the case where the interval endpoints come from a universe of size U. One algorithm uses $O(\log U)$ colors and two recolorings per update; the other uses $O(\log_t U)$ colors and $O(t)$ recolorings per update in the worst case, where $2 \leqslant t \leqslant U$ is a parameter. We then extend the second algorithm to an unbounded universe, leading to two results: we can use $O(\log_t n)$ colors and perform at most $O(t \log_t n)$ recolorings per update for any fixed $t \geqslant 2$, or we can use $O(1/\varepsilon)$ colors and $O(n^{\varepsilon}/\varepsilon)$ recolorings, amortized, for any fixed $\varepsilon > 0$.

Finally, in Section 4 we turn our attention to *kinetic conflict-free colorings*. Here the intervals do not appear or disappear, but their endpoints move continuously on the real line. At each *event* where two endpoints of different intervals cross each other, the coloring may need to be adapted so that it stays conflict-free. One way to handle this is to delete the two intervals involved in the event, and re-insert them with the new endpoint order. We show that a specialized approach is much more efficient: we show how to maintain a conflict-free coloring with four colors at the cost of three recolorings per event. We also show that on average $\Theta(1)$ recolorings per event are needed in the worst case when using only four colors.

Due to space constraints some proofs have been deferred to the full version [6].

2 Lower bounds for semi-dynamic conflict-free colorings

In this section we present lower bounds on the semi-dynamic (insertion only) conflict-free coloring problem for intervals. More precisely, we present lower bounds on the number of recolorings necessary to guarantee a given upper bound on the number of colors. We prove a general lower bound and a stronger bound for so-called local algorithms. The general lower bound uses a construction where the choice of segments to be added depends on the colors of the segments already inserted. This adaptive construction is also valid for randomized algorithms, but it does not give a lower bound on the expected behavior.

▶ **Theorem 1.** *Let* ALG *be a deterministic algorithm for the semi-dynamic conflict-free coloring of intervals. Suppose that on any sequence of $n > 0$ insertions,* ALG *uses at most $c(n)$ colors and $r(n)$ recolorings per insertion, where $r(n) > 0$. Then $r(n) > n^{1/(c(n)+1)}/(8c(n))$.*

Proof. We first fix a value for n and define $c := c(n)$ and $r := r(n)$. Our construction will proceed in rounds. In the i-th round we insert a set R_i of n_i disjoint intervals – which intervals we insert depends on the current coloring provided by ALG. After R_i has been inserted (and colored by ALG), we choose one of the colors used by ALG for R_i to be the *designated color* for the i-th round. We denote this designated color by c_i. We will argue that in each round we can pick a different designated color, so that the number of rounds, ρ, is a lower bound on the number of colors used by ALG. We then prove a lower bound on ρ in terms of n, c, and r, and derive the theorem from the inequality $\rho \leqslant c$.

To describe our construction more precisely, we need to introduce some notation and terminology. Let $R_i := \{I_1, \ldots, I_{n_i}\}$, where the intervals are numbered from left to right. (Recall that the intervals in R_i are disjoint.) To each interval $I = I_j$ we associate the set $I^e := (a, b)$, where a is the right endpoint of I, and b is the left endpoint of I_{j+1} if $j < n_i$ and $+\infty$ if $j = n_i$, that is, I^e represents the empty space to the right of I. We call (I, I^e) an i-*brick*. We define the color of a brick (I, I^e) to be the color of I, and we say a point or an interval is contained in this brick if it is contained in $I \cup I^e$. Recall that each round R_i has a designated color c_i. We say that an i-brick $B := (I, I^e)$ is *living* if:

- I has the designated color c_i;
- if $i > 1$ then both I and I^e contain living $(i-1)$-bricks.

A brick that is not alive is called *dead* and an event such as a recoloring that causes a brick to become dead is said to *kill* the brick. By recoloring an interval I, ALG can kill the brick $B = (I, I^e)$ and the death of B may cause some bricks containing B to be killed as well.

We can now describe how we generate the set R_i of intervals we insert in the i-th round and how we pick the designated colors. (Note that the designated color of a round is fixed once it is picked; it is not updated when recolorings occur.) We denote by R_i^* the subset of intervals $I \in R_i$ such that (I, I^e) is a living i-brick. Note that R_i^* can be defined only after the i-th round, when we have picked the designated color c_i.

1. The set R_1 contains the $\frac{n}{2}$ intervals $[0, 1], [2, 3], \ldots, [n-2, n-1]$, and the designated color c_1 of the first round is the color used most often in the coloring produced by ALG after insertion of the last interval in R_1.
2. To generate R_i for $i > 1$, we proceed as follows. Partition R_{i-1}^* into groups of $4r$ consecutive intervals. (If $|R_{i-1}^*|$ is not a multiple of $4r$, the final group will be smaller than $4r$. This group will be ignored.) For each group $G := I_1, \ldots, I_{4r}$ we put an interval I_G into R_i, which starts at the left endpoint of I_1 and ends slightly before the left endpoint of I_{2r+1}; see Fig. 1 for an illustration.

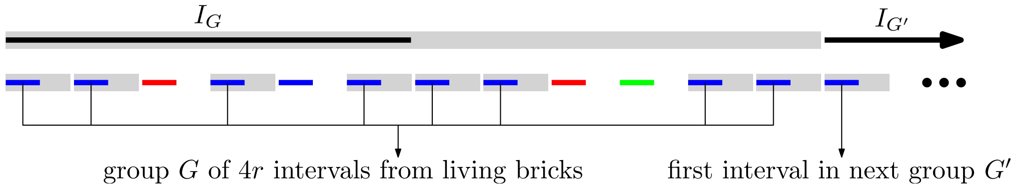

Figure 1 Example of how the intervals are created when $r = 2$. The designated color c_{i-1} is blue, and the grey rectangles around them indicate living $(i-1)$-bricks. The grey rectangle around I_G indicates the brick (I_G, I^e_G). Note that $I_{G'}$ extends further to the right.

The designated color c_i is picked as follows. Consider the coloring after the last interval of R_i has been inserted, and let $C(i)$ be the set of colors assigned by ALG to intervals in R_i and that are not a designated color from a previous round – we argue below that $C(i) \neq \emptyset$. Then we pick c_i as the color from $C(i)$ that maximizes the number of living i-bricks.

We continue generating sets R_i in this manner until $|R^*_i| < 4r$, at which point the construction finishes. Below we prove that in each round ALG must introduce a new designated color, and we prove a lower bound on the number of rounds in the construction.

▶ **Claim.** *Let $B = (I, I^e)$ be a living i-brick. Then for any $j \in \{1, \dots, i\}$ there is a point $q_j \in I \cup I^e$ that is contained in a single interval of color c_j and in no other interval from $\bigcup_{\ell=1}^{i-1} R_\ell$. Moreover, there is a point $q_j \in I \cup I^e$ not contained in any interval from $\bigcup_{\ell=1}^{i-1} R_\ell$.*

Proof of claim. We prove this by induction on i. For $i = 1$ the statement is trivially true, so suppose $i > 1$. By definition, both I and I^e contain living $(i-1)$-bricks, $\overline{B}$ and $\overline{B}^e$. Using the induction hypothesis we can now select a point q_j with the desired properties: for $j = i$ we use that $\overline{B}$ contains a point that is not contained in any interval from $\bigcup_{\ell=1}^{i-1} R_\ell$, for $j < i$ we use that $\overline{B}^e$ contains a point in an interval of color c_j and in no other interval from $\bigcup_{\ell=1}^{i-1} R_\ell$, and to find a point not contained in any interval from $\bigcup_{\ell=1}^{i-1} R_\ell$ we can also use $\overline{B}^e$. ◀

Now consider the situation after the i-th round, but before we have chosen the designated color c_i. We say that a color c is *eligible* (to become c_i) if $c \neq c_1, \dots, c_{i-1}$, and we say that an i-brick (I, I^e) is eligible if its color is eligible and (I, I^e) would be living if we were to choose its color as the designated color c_i. Note that due to some recolorings, some of the newly inserted intervals might not contain any living brick and hence can never be living no matter the designated color; the next claim shows that at most half intervals inserted this round are eligible.

▶ **Claim.** *Immediately after the i-th round, at least half of the i-bricks are eligible.*

Proof of claim. Consider an i-brick (I, I^e). At the beginning of the i-th round, before we have actually inserted the intervals from R_i, both the interval I and its empty space I^e contain $2r$ living $(i-1)$-bricks. As the intervals from R_i are inserted, ALG may recolor certain intervals from $R_1 \cup \ldots \cup R_{i-1}$, thereby killing some of these $(i-1)$-bricks. Now suppose that ALG recolored at most $2r - 1$ of the intervals from $R_1 \cup \ldots \cup R_{i-1}$ that are contained in $I \cup I^e$. Then both I and I^e still contain a living $(i-1)$-brick. By the previous claim and the definition of a conflict-free coloring, this implies ALG cannot use any of the colors c_j with $j < i$ for I. Hence, the color of I is eligible and the i-brick (I, I^e) is eligible as well.

It remains to observe that ALG can do at most rn_i recolorings during the i-th round. We just argued that to prevent an i-brick from becoming eligible, ALG must do at least $2r$ recolorings inside that brick. Hence, ALG can prevent at most half of the i-bricks from becoming eligible. ◀

Recall that after the i-th round we pick the designated color c_i that maximizes the number of living i-bricks. In other words, c_i is chosen to maximize $|R_i^*|$. Next we prove a lower bound on this number. Recall that ρ denotes the number of rounds.

▶ **Claim.** *For all $1 \leqslant i \leqslant \rho$ we have $|R_i^*| \geqslant n_1/(8rc)^i - 1$.*

Proof of claim. Since ALG can use at most c colors, we have $|R_1^*| \geqslant n_1/c$. Moreover, for $i > 1$ the number of intervals we insert is $\lfloor |R_{i-1}^*|/4r \rfloor$. By the previous claim at least half of these are eligible. The eligible intervals have at most c different colors, so if we choose c_i to be the most common color among them we see that $|R_i^*| \geqslant \lfloor |R_{i-1}^*|/4r \rfloor /2c$. We thus obtain the following recurrence:

$$|R_i^*| \geqslant \begin{cases} \dfrac{\lfloor |R_{i-1}^*|/4r \rfloor}{2c} & \text{if } i > 1, \\ \dfrac{n_1}{c} & \text{if } i = 1. \end{cases} \tag{1}$$

We can now prove the result using induction.

$$|R_i^*| \quad \geqslant \quad \frac{\lfloor |R_{i-1}^*|/4r \rfloor}{2c} \quad \geqslant \quad \frac{1}{2c} \cdot \left(\left(\frac{n_1}{(8rc)^{i-1}} - 1 \right) /4r - 1 \right) \quad > \quad \frac{n_1}{(8rc)^i} - 1. \qquad ◀$$

Finally we can derive the desired relation between $n, c,$ and r. Since $n_1 = n/2$ and $n_{i+1} < n_i/2$ for all $i = 1, \ldots, \rho - 1$, the total number of insertions is less than n. The construction finishes when $|R_i^*| < 4r$. Hence, ρ, the total number of rounds, must be such that $n/(2(8rc)^\rho) - 1 \leqslant |R_\rho^*| < 4r$, which implies $\rho > \log_{8rc}(n/(8r+2)) > \log_{8rc} n - 1$. The number of colors used by ALG is at least ρ, since every round has a different designated color. Thus $c > \log_{8rc} n - 1$ and so $n \leqslant (8rc)^{c+1}$, from which the theorem follows. ◀

Two interesting special cases of the theorem are the following: with $r = O(1)$ we will have $c = \Omega(\log n / \log\log n)$, and for $c = O(1/\varepsilon)$ (for some small fixed $\varepsilon > 0$) we need $r = \Omega(\varepsilon n^\varepsilon)$. Note that the theorem requires $r > 0$. Obviously the $\Omega(\log n / \log\log n)$ lower bound on c that we get for $r = 1$ also holds for $r = 0$. For the special case of $r = 0$ – this is the standard online version of the problem – we can prove a stronger result, however: here we need at least $\lfloor \log n \rfloor + 1$ colors. This bound even holds for a nested set of intervals, that is, a set S such that $I \subset I'$, $I' \subset I$, or $I \cap I' = \emptyset$ for any two intervals $I, I' \in S$. We also show in the full paper [6] that a greedy algorithm achieves this bound for nested intervals.

Local algorithms. We now prove a stronger lower bound for so-called local algorithms. Intuitively, these are deterministic algorithms where the color assigned to a newly inserted interval I only depends on the structure and the coloring of the connected component where I is inserted – hence the name *local*. More precisely, local algorithms are defined as follows.

Suppose we insert an interval I into a set S of intervals that have already been colored. The union of the set $S \cup \{I\}$ consists of one or more connected components. We define $S(I) \subseteq S$ to be the set of intervals from S that are in the same connected component as I. (In other words, if we consider the interval graph induced by $S \cup \{I\}$ then the intervals in $S(I)$ form a connected component with I.) Order the intervals in $S(I) \cup \{I\}$ from left

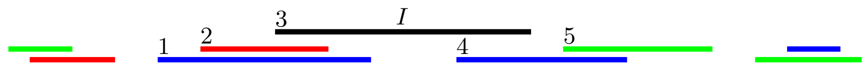

Figure 2 Example of a signature. The set $S(I)$ contains the segments labeled 1,2,4,5. The signature of I is $\langle 2, 1, 3, 4, 5, \text{red}, \text{blue}, \textsc{nil}, \text{blue}, \text{green}\rangle$.

to right according to their left endpoint, and then assign to every interval its rank in this ordering as its label. (Here we assume that all endpoints of the intervals in $S(I) \subseteq S$ are distinct. It suffices to prove our lower bound for this restricted case.) Based on this labeling we define a signature for $S(I) \cup \{I\}$ as follows. Let $\lambda_1, \ldots, \lambda_k$, where $k := |S(I)| + 1$, be the sequence of labels obtained by ordering the intervals from left to right according to their right endpoint. Furthermore, let c_i be the color of the interval labeled i, where $c_i = \textsc{nil}$ if the interval labeled i has not yet been colored. Then we define the *signature* of $S(I) \cup I$ to be the sequence $\text{sig}(I) := \langle \lambda_1, \ldots, \lambda_k, c_1, \ldots, c_k\rangle$; see Fig. 2.

We now define a semi-dynamic algorithm ALG to be *local* if upon insertion of an interval I the following holds: (i) ALG only performs recoloring in $S(I)$, and (ii) the color assigned to I and the recolorings in $S(I)$ are uniquely determined by $\text{sig}(I)$, that is, the algorithm is deterministic with respect to $\text{sig}(I)$. Note that randomized algorithms are not local.

To strengthen Theorem 1 for the case of local algorithms, it suffices to observe that the intervals inserted in the same round must all receive the same color. Hence, the factors c in the denominators of Inequality (1) disappear, leading to the theorem below. Note that for $r(n) = O(1)$, we now get the lower bound $c(n) = \Omega(\log n)$.

▶ **Theorem 2.** *Let* ALG *be a local algorithm for the semi-dynamic conflict-free coloring of intervals. Suppose that on any sequence of $n > 0$ insertions,* ALG *uses at most $c(n)$ colors and $r(n)$ recolorings per insertion, where $r(n) > 0$. Then $r(n) \geqslant n^{1/(c(n)+2)} - 2$.*

3 Upper bounds for fully-dynamic conflict-free colorings

Next we present algorithms to maintain a conflict-free coloring for a set S of intervals under insertions and deletions. The algorithms use the same structure, which we describe first. From now on, we use n to denote the current number of intervals in S.

Let P be the set of $2n$ endpoints of the intervals in S. (To simplify the presentation we assume that all endpoints are distinct, but the solution is easily adapted to the general case.) We will maintain a B-tree on the set P. A B-tree of minimum degree t on a set of points in $\mathbb{R}^1$ is a multi-way search tree in which each internal node has between t and $2t$ children (except the root, which may have fewer children) and all leaves are at the same level; see the book by Cormen *et al.* [9, Chapter 18] for details. Thus each internal or leaf node stores between $t-1$ and $2t-1$ points from P (again, the root may store fewer points). We denote the set of points stored in a node v by $P(v) := \{p_1(v), \ldots, p_{n_v}(v)\}$, where $n_v := |P(v)|$ and the points are numbered from left to right. For an internal node v we denote the i-th subtree of v, where $1 \leqslant i \leqslant n_v + 1$, by $\mathcal{T}_i(v)$. Note that the search-tree property guarantees that all points in $\mathcal{T}_i(v)$ lie in the range $(p_{i-1}(v), p_i(v))$, where $p_0 = -\infty$ and $p_{n_v+1} = \infty$.

We now associate each interval $I \in S$ to the highest node v such that I contains at least one of the points in $P(v)$, either in its interior or as one of its endpoints. Thus our structure is essentially an interval tree [5, Chapter 10] but with a B-tree as underlying tree structure. We denote the set of intervals associated to a node v by $S(v)$. Note that if $\text{level}(v) = \text{level}(w) = \ell$, for some nodes $v \neq w$, and $I \in S(v)$ and $I' \in S(w)$, then I and I' are separated by a point $p_i(z)$ of some node z at level $m < \ell$. Hence, $I \cap I' = \emptyset$.

We partition $S(v)$ into subsets $S_1(v), \ldots, S_{n_v}(v)$ such that $S_i(v)$ contains all intervals $I \in S(v)$ for which $p_i(v)$ is the leftmost point from $P(v)$ contained in I. From each subset $S_i(v)$ we pick at most two *extreme intervals*: the *left-extreme interval* $I_{i,\text{left}}(v)$ is the interval from $S_i(v)$ with the leftmost left endpoint, and the *right-extreme interval* $I_{i,\text{right}}(v)$ is the interval from $S_i(v)$ with the rightmost right endpoint. Since all intervals from $S_i(v)$ contain the point $p_i(v)$, every interval from $S_i(v)$ is contained in $I_{i,\text{left}}(v) \cup I_{i,\text{right}}(v)$. Note that it may happen that $I_{i,\text{left}}(v) = I_{i,\text{right}}(v)$. Finally, we define $S_{\text{extr}}(v) := \bigcup_{i=1}^{n_v} \{I_{i,\text{left}}(v), I_{i,\text{right}}(v)\}$ to be the set of all extreme intervals at v.

Our two coloring algorithms both maintain a coloring with the following properties.

(A.1) For each level ℓ of the tree $\mathcal{T}$, there is a set $C(\ell)$ of colors such that the color sets of different levels are disjoint.

(A.2) For each node v at level ℓ in $\mathcal{T}$, the intervals from $S_{\text{extr}}(v)$ are colored locally conflict-free using colors from $C(\ell)$. Here *locally conflict-free* means that the coloring of $S_{\text{extr}}(v)$ is conflict-free if we ignore all other intervals.

(A.3) All non-extreme intervals receive a universal *dummy color*, which is distinct from any of the other colors used, that is, the dummy color is not in $C(\ell)$ for any level ℓ.

The two coloring algorithms that we present differ in the size of the sets $C(\ell)$ and in which local coloring algorithm is used for the sets $S_{\text{extr}}(v)$. It is not hard to show that the properties above guarantee a conflict-free coloring.

▶ **Lemma 3.** *Any coloring with properties (A.1)–(A.3) is conflict free and uses at most* $1 + \sum_\ell |C(\ell)|$ *colors.*

Next we describe two algorithms based on this general framework: one for the easy case where the interval endpoints come from a finite universe, and one for the general case.

Solutions for a polynomially-bounded universe. The framework above uses a B-tree on the interval endpoints. If the interval endpoints come from a universe of size U – for concreteness, assume the endpoints are integers in the range $0, \ldots, U-1$ – then we can use a B-tree on the set $\{0, \ldots, U-1\}$. Thus the B-tree structure never has to be changed.

▶ **Theorem 4.** *Let S be a dynamic set of intervals whose endpoints are integers in the range* $0, \ldots, U-1$.

(i) *We can maintain a conflict-free coloring on S that uses $O(\log U)$ colors and that performs at most two recolorings per insertion and deletion.*

(ii) *For any t with $2 \leqslant t \leqslant U$, we can maintain a conflict-free coloring on S that uses $O(\log_t U)$ colors and performs $O(t)$ recolorings per insertion and deletion.*

When U is polynomially bounded in n – that is, $U = O(n^k)$ for some constant k – this gives very efficient coloring schemes. In particular, we can then get $O(\log n)$ colors with at most two recolorings using method (i), and we can get $O(1/\varepsilon)$ colors with $O(n^\varepsilon)$ recolorings (for any fixed $\varepsilon > 0$) by setting $t = U^{\varepsilon/k}$ in method (ii).

Note finally that we do not need to explicitly store the whole tree as it is enough to compute the location of any node when needed, yielding a linear space complexity.

A general solution. If the interval endpoints do not come from a bounded universe then we cannot use a fixed tree structure. Next we explain how to deal with this when we apply the method from Theorem 4(ii), which colors the sets $S_{\text{extr}}(v)$ using the so-called *chain method*: we take the interval with the leftmost left endpoint, and color it blue. Then, among all intervals whose left endpoint lies in the blue interval, we pick the one with the rightmost right endpoint and color it red. We then repeat this process, alternating between blue and red, until we reach the rightmost interval. Finally, we color all uncolored intervals grey.

Suppose we want to insert a new interval I into the set S. We first insert the two endpoints of I into the B-tree $\mathcal{T}$. Inserting an endpoint p can be done in a standard manner. The basic operations for an insertion are (i) to split a full node and (ii) to insert a point into a non-full leaf node.

Splitting a full node v (that is, a node with $2t-1$ points) is done by moving the median point into the parent of v, creating a node containing the $t-1$ points to the left of the median and another node containing the $t-1$ points to the right of the median. Note that this means that some intervals from $S(v)$ may have to be moved to $S(\mathit{parent}(v))$. Thus splitting a node v involves recoloring intervals in $S(v)$ and $S(\mathit{parent}(v))$. Observe that an interval only needs to be recolored if it was extreme before or after the change. Hence, we recolor $O(t)$ intervals when we split a node v.

Since an insertion splits only nodes on a root-to-leaf path and the depth of $\mathcal{T}$ is $O(\log_t n)$, the total number of recolorings due to node splitting is $O(t \log_t n)$. Moreover, inserting a point into a non-full leaf node only takes $O(t)$ recolorings. We conclude that an insertion performs $O(t \log_t n)$ recolorings in total. For deletions the argument is similar. Since recoloring at a single node induces $O(t)$ recolorings, the total number of recolorings is $O(t \log_t n)$.

▶ **Theorem 5.** *Let S be a dynamic set of intervals.*

(i) *For any fixed $t \geqslant 2$ we can maintain a conflict-free coloring on S that uses $O(\log_t n)$ colors and that performs $O(t \log_t n)$ recolorings per insertion and deletion, where n is the current number of intervals in S. In particular, we can maintain a conflict-free coloring with $O(\log n)$ colors using $O(\log n)$ recolorings per update.*

(ii) *For any fixed $\varepsilon > 0$ we can maintain a conflict-free coloring on S that uses $O(1/\varepsilon)$ colors and that performs $O(n^\varepsilon/\varepsilon)$ recolorings per insertion or deletion. The bound on the number of recolorings is amortized.*

The idea behind part (ii) is to use a t with $n^\varepsilon/2 \leqslant t \leqslant 2n^\varepsilon$. This causes the bound in (ii) to be amortized, since now we need to change t when n has halved or doubled.

We have not been able to efficiently generalize the first method of Theorem 4 to an unbounded universe. The problem is that splitting a node v may require many intervals in $S_{\text{extr}}(v)$ to be recolored, since many intervals may be moved to $\mathit{parent}(v)$. Hence, the method would use the same number of recolorings as the chain method, but more colors.

Bounded-length intervals. Next we present a simple method that allows us to improve the bounds when the intervals have length between 1 and L for some constant $L > 1$.

▶ **Theorem 6.** *Let S be a dynamic set of intervals with lengths in the range $[1, L)$ for some fixed $L > 1$. Suppose we have a dynamic conflict-free coloring algorithm for a general set of intervals that uses at most $c(n)$ colors and at most $r(n)$ recolorings for any insertion or deletion. Then we can obtain a dynamic conflict-free coloring algorithm on S that uses at most $2 \cdot c(2L) + 1$ colors and at most $2 \cdot r(2L) + 1$ recolorings for any insertion or deletion.*

For instance, by applying Theorem 5(i) we can maintain a coloring with $O(\log L)$ colors and $O(\log L)$ recolorings. We can also plug in a trivial dynamic algorithm with $c(n) = n$ and $r(n) = 0$ to obtain $4L + 1$ colors with only 1 recoloring per update; when L is sufficiently small this gives a better result.

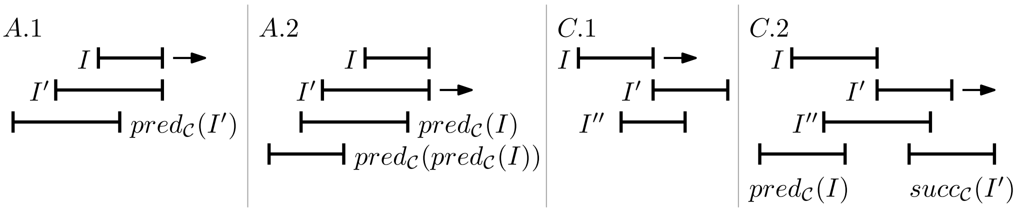

Figure 3 Illustration of the different events in the KDS.

4 Kinetic conflict-free colorings

In this section we consider conflict-free colorings of a set of intervals in $\mathbb{R}^1$ whose endpoints move continuously. Note that we allow the endpoints of an interval to move independently, that is, we allow the intervals to expand or shrink over time. We show that by using only three recolorings per event – an event is when two endpoints cross each other – we can maintain a conflict-free coloring consisting of only four colors. Our recoloring strategy is based on the chain method discussed in the proof of Theorem 4(ii). This method uses three colors: two colors for the chain and one dummy color. To be able to maintain the coloring in the kinetic setting without using many recolorings, we relax the chain properties and we allow ourselves three colors for the chain. Next we describe the invariants we maintain on the chain and its coloring, and we explain how to re-establish the invariants when two endpoints cross each other. In the remainder we assume that the endpoints of the chains are in general position except at events, and that events do not coincide (that is, we never have three coinciding endpoints and we never have two events at the same time). These conditions can be removed by using consistent tie-breaking.

The chain invariants. Let S be the set of intervals to be colored, where all interval endpoints are distinct. (Recall that we assumed this to be the case except at event times.) Consider a subset $\mathcal{C} \subseteq S$ and order the intervals in $\mathcal{C}$ according to their left endpoint. We denote the predecessor of an interval $I \in \mathcal{C}$ in this order by $pred_{\mathcal{C}}(I)$, and we denote its successor by $succ_{\mathcal{C}}(I)$. A *chain* (for S) is defined as a subset $\mathcal{C}$ with the following three properties.

(C1) Any interval $I \in \mathcal{C}$ can intersect only two other intervals in $\mathcal{C}$, namely $pred_{\mathcal{C}}(I)$ and $succ_{\mathcal{C}}(I)$.

(C2) Any interval $I \in S \setminus \mathcal{C}$ is completely covered by the intervals in $\mathcal{C}$.

(C3) No interval $I \in \mathcal{C}$ is fully contained in any other interval $I' \in S$.

Now consider a set S and a chain $\mathcal{C}$ for S. We maintain the following *color invariant*: each interval $I \in \mathcal{C}$ has a non-dummy color and this color is different from the color of $succ(I)$, and each interval in $S \setminus \mathcal{C}$ has the dummy color.

▶ **Lemma 7.** *Let S be a set of intervals and $\mathcal{C}$ be a chain for S. Then any coloring of S satisfying the color invariant is conflict-free.*

Handling events. Our kinetic coloring algorithm maintains a chain $\mathcal{C}$ for I and a coloring with three colors (excluding the dummy color) satisfying the color invariant. Later we show how to re-establish the color invariant at each event, but first we show how to update the chain by adding at most one interval to the chain and removing at most two. We distinguish several cases.

- *Case A: The right endpoints of two intervals I and I' cross.*
  Assume without loss of generality that I is shorter than I'. We have two subcases.
  - *Subcase A.1: Interval I is contained in I' before the event.* In this case I was not a chain interval before the event. If after the event I is still fully covered by the chain intervals, then there is nothing to do: we can keep the same chain. Otherwise, property (C2) is violated after the event. We now proceed as follows. First we add I to the chain. If I intersects $pred_{\mathcal{C}}(I')$ – note that I' must be a chain interval if (C2) is violated – then we remove I' from the chain.
  - *Subcase A.2: Interval I is contained in I' after the event.* If I was not a chain interval, there is nothing to do. Otherwise property (C3) no longer holds after the event, and we have to remove I from the chain. If I' is also a chain interval then this suffices. Otherwise we add I' to the chain, and remove $pred_{\mathcal{C}}(I)$ if $pred_{\mathcal{C}}(pred_{\mathcal{C}}(I))$ intersects I'.
- *Case B: The left endpoints of two intervals I and I' cross.*
  This case is symmetric to Case A.
- *Case C: The right endpoint of an interval I crosses the left endpoint of an interval I'.*
  Again we have two subcases.
  - *Subcase C.1: Intervals I and I' start intersecting.* Note that properties (C2) and (C3) still hold after the event. The only possible violation is in property (C1), namely when both I and I' are chain intervals and there is a chain interval I'' with $pred_{\mathcal{C}}(I'') = I$ and $succ_{\mathcal{C}}(I'') = I'$. In this case we simply remove I'' from the chain.
  - *Subcase C.2: Intervals I and I' stop intersecting.* First note that this cannot violate properties (C1) and (C3). The only possible violation is property (C2), namely when both I and I' are chain intervals and there is at least one non-chain interval containing the common endpoint of I and I' at the event. Of all such non-chain intervals, let I'' be the interval with the leftmost left endpoint. Note that I'' is not contained in any other interval, so we can add I'' to the chain without violating (C3). After adding I'' we check if we have to remove I and/or I': if I'' intersects $pred_{\mathcal{C}}(I)$ then we remove I from the chain, and if I'' intersects $succ_{\mathcal{C}}(I')$ then we remove I' from the chain.

It is easy to check that in each of these cases the new chain that we generate has the chain properties (C1)–(C3). Next we show that each case requires at most three recolorings and summarize the result.

▶ **Lemma 8.** *In each of the above cases, the changes to the chain require at most three recolorings to re-establish the color invariant.*

▶ **Theorem 9.** *Let S be a kinetic set of intervals in $\mathbb{R}^1$. We can maintain a conflict-free coloring for S with four colors at the cost of at most three recolorings per event, where an event is when two interval endpoints cross each other.*

A lower bound. Now consider the simple scenario where the intervals are rigid – each interval keeps the same length over time – and each interval is either stationary or moves with unit speed to the right. Our coloring algorithm may perform recolorings whenever two endpoints cross, which means that we do $O(n^2)$ recolorings in total. We show that even in this simple setting, this bound is tight in the worst case if we use at most four colors.

Consider four intervals I_1, I_2, I_3, I_4 where $I_i = (a_i, b_i)$, with $a_i < b_i$ as shown in Figure 4. Here $I_2 \subset I_1$, $I_4 \subset I_3$, the right endpoints of I_1 and I_2 are contained in $I_3 \cap I_4$, and the left endpoints of I_3 and I_4 are contained in $I_1 \cap I_2$. The exact locations of the endpoints with respect to each other is not important and we focus on the different overlap sets of the

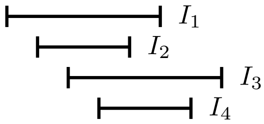

Figure 4 The gadget used to show the lower bound.

gadget. Specifically within a gadget there is a point contained in each of the following sets,

$$G_1, \ldots, G_7 := \{I_1\}, \{I_1, I_2\}, \{I_1, I_2, I_3, \}, \{I_1, I_2, I_3, I_4\}, \{I_1, I_3, I_4\}, \{I_3, I_4\}, \{I_3\}.$$

Based on these sets we can show that no coloring for crossing gadgets exists that provides a valid conflict-free coloring for each combination of intersection sets between the two gadgets. The proof relies on the following lemma.

▶ **Lemma 10.** *Let $G = \{I_1, I_2, I_3, I_4\}$ and $H = \{J_1, J_2, J_3, J_4\}$ be two gadgets, with overlap sets $G_1, \ldots, G_7$ and $H_1, \ldots, H_7$ as defined above. There is no 4-coloring for G and H such that all sets $\{G_1, \ldots, G_7\} \cup \{H_1, \ldots, H_7\} \cup \{1 \leqslant i, j \leqslant 7 \mid G_i \cup H_i\}$ are conflict-free.*

Proof. We can assume that not both I_1, I_2, I_3, I_4 and J_1, J_2, J_3, J_4 use all four colors, otherwise $G_4 \cup H_4 = \{I_1, I_2, I_3, I_4, J_1, J_2, J_3, J_4\}$ is not conflict-free. It is also not possible to use at most two colors, since each gadget by itself needs to be conflict-free. Hence, suppose that there are exactly three colors among I_1, I_2, I_3, I_4 (the other case is symmetric), say two are red, one is blue, and one is green. We define $\text{col}(G_i)$, respectively $\text{col}(H_i)$, to be the multiset of the colors used by the intervals in G_i, respectively H_i. Then $\text{col}(G_4) = \{\text{red}, \text{red}, \text{blue}, \text{green}\}$ and without loss of generality, $\text{col}(G_2) = \{\text{red}, \text{blue}\}$ and $\text{col}(G_6) = \{\text{red}, \text{green}\}$. We now have two cases.

1. One interval among J_1, J_2, J_3, J_4 uses the fourth color, say yellow. If J_1 or J_2 is yellow, then either $\text{col}(H_6) = \{\text{red}, \text{blue}\}$, implying that $G_2 \cup H_6$ is not conflict-free; or $\text{col}(H_6) = \{\text{red}, \text{green}\}$ implying that $G_6 \cup H_6$ is not conflict-free; or $\text{col}(H_6) = \{\text{blue}, \text{green}\}$ implying that $G_4 \cup H_6$ is not conflict-free. A similar argument holds when J_3 or J_4 is yellow.
2. Two intervals among J_1, J_2, J_3, J_4 use yellow. It follows that H_4 contains two yellow intervals and the remaining two intervals are colored either $\{\text{red}, \text{blue}\}$, implying that $G_2 \cup H_4$ is not conflict-free; or $\{\text{red}, \text{green}\}$, implying that $G_6 \cup H_4$ is not conflict-free; or $\{\text{blue}, \text{green}\}$, implying that $G_4 \cup H_4$ is not conflict-free. ◀

Now we place $\Omega(n)$ of these gadgets in two groups and for simplicity assume a gadget has width of 1. The gadgets in the first group are spaced with distance 2 between them, so a gadget from the second group can fit between any two consecutive gadgets. In the second group the gadgets are spaced with distance $3n$ between them, so that all gadgets of the first group fit between them. All gadgets of the first group then move at the same speed, starting somewhere to the left of the second group and moving to the right. The gadgets of the second group remain stationary. These motions ensure that each gadget of first group will cross each gadget of the second group, generating $\Omega(n^2)$ crossing events, each of which results in at least one recoloring by Lemma 10.

▶ **Theorem 11.** *For any $n > 0$, there is a set of $8n$ intervals, each of which is either stationary or moves with unit speed to the right, so that when coloring the intervals using four colors at least n^2 recolorings are required to maintain a conflict-free coloring.*

References

1 M.A. Abam, M.J. Rezaei Seraji, and M. Shadravan. Online conflict-free coloring of intervals. *Scientia Iranica*, 21(6):2138–2141, 2014. URL: http://scientiairanica.sharif.edu/article_3607.html.

2 A. Bar-Noy, P. Cheilaris, S. Olonetsky, and S. Smorodinsky. Online conflict-free colouring for hypergraphs. *Combinatorics, Probability & Computing*, 19(4):493–516, 2010. doi:10.1017/S0963548309990587.

3 A. Bar-Noy, P. Cheilaris, and S. Smorodinsky. Deterministic conflict-free coloring for intervals: From offline to online. *ACM Trans. Algorithms*, 4(4):44:1–44:18, 2008. doi:10.1145/1383369.1383375.

4 Andreas Bärtschi and Fabrizio Grandoni. On conflict-free multi-coloring. In Frank Dehne, Jörg-Rüdiger Sack, and Ulrike Stege, editors, *Algorithms and Data Structures - 14th International Symposium, WADS 2015, Victoria, BC, Canada, August 5-7, 2015. Proceedings*, volume 9214 of *Lecture Notes in Computer Science*, pages 103–114. Springer, 2015. doi:10.1007/978-3-319-21840-3_9.

5 M. de Berg, O. Cheong, M. van Kreveld, and M. Overmars. *Computational Geometry: Algorithms and Applications*. Springer-Verlag, 3rd ed. edition, 2008.

6 M. de Berg, T. Leijsen, A. Markovic, A. van Renssen, M. Roeloffzen, and G. Woeginger. Dynamic and kinetic conflict-free coloring of intervals with respect to points. *CoRR*, 2016. URL: http://arxiv.org/abs/1701.03388.

7 P. Cheilaris, L. Gargano, A.A. Rescigno, and S. Smorodinsky. Strong conflict-free coloring for intervals. *Algorithmica*, 70(4):732–749, 2014. doi:10.1007/s00453-014-9929-x.

8 K. Chen. How to play a coloring game against a color-blind adversary. In *Proc. 22nd ACM Symp. Comput. Geom.*, pages 44–51, 2006. doi:10.1145/1137856.1137865.

9 T.H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein. *Introduction to Algorithms*. The MIT Press, 3rd edition, 2009.

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 Journal on Computing*, 33(1):94–136, 2003. doi:10.1137/S0097539702431840.

11 A. Fiat, M. Levy, J. Matousek, E. Mossel, J. Pach, M. Sharir, S. Smorodinsky, U. Wagner, and E. Welzl. Online conflict-free coloring for intervals. In *Proc. 16th ACM-SIAM Symp. Discr. Alg.*, pages 545–554, 2005. URL: http://dl.acm.org/citation.cfm?id=1070432.1070506.

12 S. Har-Peled and S. Smorodinsky. Conflict-free coloring of points and simple regions in the plane. *Discrete & Computational Geometry*, 34(1):47–70, 2005. doi:10.1007/s00454-005-1162-6.

13 E. Horev, R. Krakovski, and S. Smorodinsky. Conflict-free coloring made stronger. In *Proc. 12th Scandinavian Workshop. Alg. Theory*, pages 105–117, 2010.

14 S. Smorodinsky. *Combinatorial Problems in Computational Geometry*. PhD thesis, Tel-Aviv University, 2003.

15 S. Smorodinsky. Conflict-free coloring and its applications. In *Geometry — Intuitive, Discrete, and Convex: A Tribute to László Fejes Tóth*, pages 331–389. Springer Berlin Heidelberg, 2013.