Dynamic Coloring of Unit Interval Graphs with Limited Recourse Budget

Bartłomiej Bosek
Theoretical Computer Science Department, Faculty of Mathematics and Computer Science, Jagiellonian University in Kraków, Poland

Anna Zych-Pawlewicz[1]
University of Warsaw, Poland

Abstract

In this paper we study the problem of coloring a unit interval graph which changes dynamically. In our model the unit intervals are added or removed one at the time, and have to be colored immediately, so that no two overlapping intervals share the same color. After each update only a limited number of intervals are allowed to be recolored. The limit on the number of recolorings per update is called the recourse budget. In this paper we show, that if the graph remains k-colorable at all times, the updates consist of insertions only, and the final instance consists of n intervals, then we can achieve an amortized recourse budget of $\mathcal{O}(k^7 \log n)$ while maintaining a proper coloring with k colors. This is an exponential improvement over the result in [10] in terms of both k and n. We complement this result by showing the lower bound of $\Omega(n)$ on the amortized recourse budget in the fully dynamic setting. Our incremental algorithm can be efficiently implemented.

As an additional application of our techniques we include a new combinatorial result on coloring unit circular arc graphs. Let L be the maximum number of arcs intersecting in one point for some set of unit circular arcs $\mathcal{A}$. We show that if there is a set $\mathcal{A}'$ of non-intersecting unit arcs of size $L^2 - 1$ such that $\mathcal{A} \cup \mathcal{A}'$ does not contain $L + 1$ arcs intersecting in one point, then it is possible to color $\mathcal{A}$ with L colors. This complements the work on circular arc coloring [4, 30, 31], which specifies sufficient conditions needed to color $\mathcal{A}$ with $L + 1$ colors or more.

2012 ACM Subject Classification Theory of computation → Design and analysis of algorithms

Keywords and phrases dynamic algorithms, unit interval graphs, coloring, recourse budget, parametrized dynamic algorithms

Digital Object Identifier 10.4230/LIPIcs.ESA.2022.25

Related Version *Full Version*: `https://arxiv.org/abs/2202.08006`

Funding National Science Centre, Poland, Grant 2017/26/D/ST6/00264.

Acknowledgements The authors want to thank an anonymous reviewer for very useful comments.

1 Introduction

In this paper we study dynamic algorithms for the graph coloring problem. The setting we focus on is as follows. We are given a graph G, which is modified over time by vertex insertions or vertex deletions, where vertices are inserted (or deleted) together with all the adjacent edges connecting them to the vertices that are already present in G. For a positive integer k, a *proper k-coloring* of a graph is an assignment of colors in $\{1, \ldots, k\}$ to the vertices of the graph in such a way that no two adjacent vertices share a color. We say that a graph admitting such an assignment is k-colorable. In the dynamic setting, the ultimate goal is to design an algorithm, that (for some given $l \geqslant k$) efficiently maintains the proper l-coloring on a dynamically changing k-colorable graph G.

[1] corresponding author

Dynamic graph coloring is a fundamental problem in computer science and as such it has received a lot of attention in the literature. Chronologically, dynamic graph coloring was first considered in a more restricted online setting, where the updates consist of vertex insertions only, and one is not allowed to change the colors of the previously added vertices. Many pessimistic lower bounds for online graph coloring have been revealed. In particular, for general k-colorable graphs, one cannot color the graph online with less then $l = \frac{n}{\log^2 n}k$ colors [18], where n is the number of vertices added to G. This lower bound holds even for randomized algorithms. Even for trees, which are 2-colorable, one needs $l = \Omega(\log n)$ colors [3] in the online model. The situation improves if we consider k-colorable interval graphs, for which $l = 3k - 2$ colors are necessary and sufficient in the online model [20]. The case of unit interval graphs has also been extensively studied, revealing that $2k - 1$ colors suffice and $3k/2$ colors are necessary to color a k-colorable unit interval graph online [6, 15].

To go beyond the pessimistic lower bounds imposed by the online model, several settings were proposed where an algorithm is given more power. The one that received a lot of attention is the limited recourse budget framework. In this setting the algorithm is allowed to change a number of past decisions, but there is a limit on the number of such changes, referred to as the recourse budget. This model has been well established and successfully applied to a variety of optimization problems, often implying efficient dynamic algorithms [5, 7, 8, 9, 19, 23, 21, 2, 29, 10]. For the coloring problem that we study, the recourse budget is the number of vertices that change their color after an addition or removal of a vertex. Barba et al. apply the recourse budget model to coloring general graphs [2]. They devise two complementary algorithms. For any $d > 0$, the first (resp. second) algorithm maintains a $k(d+1)$-coloring (resp. $k(d+1)n^{1/d}$-coloring) of a k-colorable graph and recolors at most $(d+1)n^{1/d}$ (resp. d) vertices per update, which is either addition or removal of a vertex. While the second trade-off was improved in [29], the authors in [2] show that the first trade-off is tight, and the bad example is a forest. Thus, if one insists on using few colors, one has to incur a polynomial in n recourse budget on every class of graphs that contains forests. This pushed the researchers to apply the limited recourse budget model to coloring interval and unit interval graphs.

For unit interval graphs a very positive result has been obtained. The recoloring budget of $\mathcal{O}(k^2)$ (worst case) is sufficient for maintaining a $(k+1)$-coloring of a k-colorable graph [10]. It is left open what budget is needed for maintaining an optimal k-coloring for unit interval graphs, even if we only allow vertex insertions. The lower bound given in [10] is $\Omega(\log n)$ (even when updates are only insertions), while the upper bound (which only works for vertex insertions) is $\mathcal{O}(k!\sqrt{n})$ (the same as for general interval graphs). Such a tremendous gap for such elementary graph class calls for further investigation. The main result of this paper is that we close this gap up to the factors polynomial in k. To be more precise, we show that an amortized recourse budget of $\mathcal{O}(k^7 \log n)$ is sufficient to maintain k-coloring under vertex insertions. This is an exponential improvement over [10] in terms of both n and k. It is fairly easy to see that our algorithm can be efficiently implemented. We complement this result by showing that in the fully dynamic setting one must spent an amortized recourse budget of $\Omega(n)$ per update.

It is worth emphasizing, that our results show a fine line between $(k+1)$-coloring and optimal k-coloring of unit interval graphs in the limited recourse budget model. While $(k+1)$-coloring admits an algorithm with the worst case recourse budget of $\mathcal{O}(k^2)$ in the fully dynamic setting, we cannot hope (in this setting) for any reasonable recourse budget for optimal k-coloring, even if we allow amortization. If we restrict to only adding intervals, we get the amortized recourse budget of $\mathcal{O}(k^7 \log n)$ and $\Omega(\log n)$ is the lower bound. The incremental setting is interesting by itself, as it generalizes the online model (where the recourse budget is zero and only insertions are allowed). In other words, we show that if in the online model we allow a modest number of $\mathcal{O}(k^7 \log n)$ recolorings per update, we can get down from $3k/2$ to the optimum number k of colors. Our result is not only motivated by the online model, but also fits nicely in the recent line of research related to parameterized dynamic algorithms [1, 14, 12], with k being the parameter.

The techniques we develop to obtain our main result seem applicable to a wider range of problems. In particular, we apply one of our techniques to the problem of coloring unit circular arc graphs. We obtain a combinatorial result that nicely fits into the related line of research. Imagine that L is the maximum number of arcs intersecting in one point for some set of unit circular arcs $\mathcal{A}$. We show that if there is a set $\mathcal{A}'$ of non-intersecting unit arcs of size $L^2 - 1$ such that $\mathcal{A} \cup \mathcal{A}'$ does not contain $L + 1$ arcs intersecting in one point, then it is possible to color $\mathcal{A}$ with L colors. This complements the work on circular arc coloring [4, 30, 31], which specifies sufficient conditions needed to color $\mathcal{A}$ with $L + 1$ colors or more.

The remainder of the paper is organized as follows. In Section 2 we provide basic definitions related to interval graphs and unit interval graphs, together with some elementary algorithms that solve the coloring problem for static instances of those graphs. In Section 3 we provide an overview of our results and techniques. Due to space limitations, we only sketch the proofs in Section 3, skipping most of technical details. The full proofs can be found in the full version of the paper [11]. We conclude with Section 4 in which we discuss some properties and extensions of our main result. In particular, we discuss the implementation of our incremental algorithm and its actual running time (which is amortized $\mathcal{O}(k^7 \log n)$ per interval insertion). We also discuss extending our incremental algorithm to changing values of k, and finally we briefly discuss why our main result does not easily extend to general interval graphs. We end Section 4 with the problems that we leave open.

2 Preliminaries

We consider in this paper closed-open intervals $I = [x, y)$ for some $x, y \in \mathbb{R}, x < y$. Note that this causes no loss in generality, as closed-open intervals induce the same class of graphs as open-closed, closed-closed and open-open intervals (see [13, 26]). For an interval I we define operators $\mathsf{x}(I) \overset{\text{def}}{=} x$ and $\mathsf{y}(I) \overset{\text{def}}{=} y$ for accessing the begin and the end coordinate. A multiset of intervals can be interpreted as a graph: the intervals are interpreted as vertices, which are adjacent if and only if the corresponding intervals intersect. Graphs obtained in this way are called interval graphs. The coloring related definitions stated in the first paragraph of Section 1 directly translate to multisets of intervals interpreted as graphs. For a multiset of intervals $\mathcal{I}$, the function $c : \mathcal{I} \mapsto \{1, \ldots, k\}$ is a proper k-coloring if for any $I, J \in \mathcal{I}$ it holds that $c(I) \neq c(J)$ if I and J intersect. The chromatic number $\chi(\mathcal{I})$ is the minimum number k for which $\mathcal{I}$ admits a proper k-coloring. Similarly we can adapt the definition of a clique: a multiset of intervals $\mathcal{J} = \{J_1, J_2, \ldots, J_m\}$ is a clique if and only if $\bigcap \mathcal{J} \overset{\text{def}}{=} J_1 \cap \ldots \cap J_m \neq \emptyset$, or equivalently $\max_{i=1}^{m} \mathsf{x}(J_i) < \min_{i=1}^{m} \mathsf{y}(J_i)$. In such case it holds that $\bigcap \mathcal{J} = [\max_{i=1}^{m} \mathsf{x}(J_i), \min_{i=1}^{m} \mathsf{y}(J_i))$. We refer to this intersection as span of $\mathcal{J}$, and denote it as $\mathsf{span}(\mathcal{J}) \overset{\text{def}}{=} \bigcap \mathcal{J}$. To emphasize the size of $\mathcal{J}$ we often refer to it as an m-clique. The clique number $\omega(\mathcal{I})$ of a multiset of intervals $\mathcal{I}$ is the maximum number m such that $\mathcal{I}$ contains an m-clique. It is well known that interval graphs are perfect, that is:

▶ **Lemma 1** (Golumbic [17]). *Let $\mathcal{I}$ be a multiset of intervals. Then $\omega(\mathcal{I}) = \chi(\mathcal{I})$.*

We introduce two orders $\sqsubset$ and $<$ on a multiset of intervals $\mathcal{I}$. For any $I, J \in \mathcal{I}$ we let $I \sqsubset J \quad \text{if} \quad \mathsf{x}(I) < \mathsf{x}(J)$. If $I = J$, we solve the tie arbitrarily. Hence, $\sqsubset$ is a linear order on $\mathcal{I}$. We say that $I < J$ if and only if $\mathsf{y}(I) \leqslant \mathsf{x}(J)$. Hence, $<$ is a linear order only on independent sets of intervals (i.e., multisets of intervals that are pairwise non-intersecting). Orders $\sqsubset$ and $<$ extend to multisets of intervals in a natural manner. For two multisets of intervals $\mathcal{J}$ and $\mathcal{K}$, we say that $\mathcal{J} \sqsubset \mathcal{K}$ (respectively $\mathcal{J} < \mathcal{K}$) if for all $J \in \mathcal{J}, K \in \mathcal{K}$ it holds that $J \sqsubset K$ (respectively $J < K$). For a multiset of intervals $\mathcal{I}$ by $\mathcal{I} = \{I_1 \sqsubset \ldots \sqsubset I_m\}$ we denote that $\mathcal{I} = \{I_1, \ldots, I_m\}$ and $I_1 \sqsubset \ldots \sqsubset I_m$. For an ordered multiset of intervals $\mathcal{I} = \{I_1 \sqsubset \ldots \sqsubset I_m\}$ we now define a prefix, a suffix and an infix of $\mathcal{I}$. For $I_i, I_l \in \mathcal{I}$, $l > i$ we set $\mathsf{prefix}^{\sqsubset}_{\mathcal{I}}(I_i) = \{I_1, \ldots, I_i\}$, $\mathsf{suffix}^{\sqsubset}_{\mathcal{I}}(I_i) = \{I_i, \ldots, I_m\}$ and $\mathsf{infix}^{\sqsubset}_{\mathcal{I}}(I_i, I_l) = \{I_i, \ldots, I_l\}$.

We now move on to some basic observations that hold for multisets of unit intervals. An interval I is unit if and only if $\mathsf{y}(I) = \mathsf{x}(I) + 1$.

▶ **Observation 2.** *Let $\mathcal{I} = \{I_1 \sqsubset \ldots \sqsubset I_m\}$ be a multiset of unit intervals. If $\omega(\mathcal{I}) < m$, then the extremal intervals are disjoint, i.e., $I_1 < I_m$.*

Proof. Suppose contrary that $I_1 \cap I_m \neq \emptyset$, i.e., $\mathsf{x}(I_m) < \mathsf{y}(I_1) = \mathsf{x}(I_1) + 1$. Then $\max_{i=1}^{m} \mathsf{x}(I_i) < \min_{i=1}^{m} \mathsf{y}(I_i)$ and as a consequence $\bigcap \mathcal{I} \neq \emptyset$ contradicting $\omega(\mathcal{I}) < m$. ◀

For interval graphs there is a simple greedy algorithm, that can be applied to complete a coloring given on the prefix of the representation ordered by $\sqsubset$ [24]. In this paper we need a more specific but equally simple algorithm which is restricted to unit interval graphs. The same idea was used for instance to color proper circular arc graphs [25] or to schedule round-robin tournaments [27]. Since we use it in a different context, we introduce it here from scratch and we refer to it as the MODULO COLOR COMPLETION algorithm. Informally, given some coloring on the prefix of a k-colorable instance, this algorithm looks at the first k consecutive uncolored intervals (in $\sqsubset$ order), and copies the coloring given on the last k intervals that are colored (respecting the $\sqsubset$ order). This proceeds until all intervals are colored. This simple procedure works given that the coloring that is being copied consists of all colors from 1 to k. We describe the MODULO COLOR COMPLETION algorithm more formally in the following observation, which also proves the corectness.

▶ **Observation 3.** *Let $\mathcal{I} = \{I_1 \sqsubset \ldots \sqsubset I_m\}$ be a multiset of unit intervals such that $\omega(\mathcal{I}) \leqslant k$. Let $k \leqslant l < m$ and let $c : \mathsf{prefix}^{\sqsubset}_{\mathcal{I}}(I_l) \mapsto [k]$ be a proper k-coloring for $\mathsf{prefix}^{\sqsubset}_{\mathcal{I}}(I_l)$ such that c is a bijection on $\mathsf{infix}^{\sqsubset}_{\mathcal{I}}(I_{l-k+1}, I_l)$. Let $c' : \mathcal{I} \mapsto [k]$ (referred to as MODULO COLOR COMPLETION) be defined as follows: $c'(I_i) \overset{\text{def}}{=} c(I_i)$ for $i \in [l]$ and $c'(I_{l+i}) \overset{\text{def}}{=} c(I_{l-k+(i \mod k)})$ for $i \in [n-l]$. Then c' is a proper k-coloring on $\mathcal{I}$.*

Proof. It is clear that c assigns only colors in $[k]$, it remains to prove that it is also a proper coloring. Let $I_i, I_j \in \mathcal{I}$, where $i < j$ and $j > l$. If $j < i + k$, then by definition $c(I_i) \neq c(I_j)$. Otherwise, $|\{I_i, I_{i+1}, \ldots, I_j\}| \geqslant k + 1$, so by Observation 2, $I_i \cap I_j = \emptyset$. ◀

The following observation will be useful to bring the prefix coloring to the state when we can use the MODULO COLOR COMPLETION algorithm, i.e., the coloring on the prefix ends with the bijection.

▶ **Observation 4.** *Let $\mathcal{I} = \{I_1 \sqsubset \ldots \sqsubset I_{2k}\}$ be a multiset of unit intervals such that $\omega(\mathcal{I}) \leqslant k$ and let $c' : \mathsf{prefix}^{\sqsubset}_{\mathcal{I}}(I_k) \mapsto [k]$ be a proper k-coloring on $\mathsf{prefix}^{\sqsubset}_{\mathcal{I}}(I_k)$. Then there is a proper k-coloring $c : \mathcal{I} \mapsto [k]$ such that $c(I_i) = c'(I_i)$ for $i \in [k]$ and c is a bijection on $\mathsf{suffix}^{\sqsubset}_{\mathcal{I}}(I_{k+1})$.*

Proof. To construct c, we first set $c(I_i) = c'(I_i)$ for all $i \in [k]$. Let $\mathcal{J} = \{I_l, I_{l+1}, \ldots I_k\}$ be the intervals of $\mathsf{prefix}^{\sqsubset}_{\mathcal{I}}(I_k)$ that intersect I_k. Thus, $\mathcal{J}$ is a $(k - l + 1)$-clique. We first assign the $l - 1$ colors not used by $\mathcal{J}$ to the first $l - 1$ intervals of $\mathsf{suffix}^{\sqsubset}_{\mathcal{I}}(I_{k+1})$. That is we assign colors $[k] \setminus c(\mathcal{J})$ to intervals $\mathsf{infix}^{\sqsubseteq}_{\mathcal{I}}(I_{k+1}, I_{k+l-1})$ in an arbitrary order. We get a proper coloring since the intervals of $\mathsf{infix}^{\sqsubseteq}_{\mathcal{I}}(I_{k+1}, I_{k+l-1})$ do no intersect intervals of $\mathsf{prefix}^{\sqsubseteq}_{\mathcal{I}}(I_k)$ other than $\mathcal{J}$. Finally, we set $c(I_i) = c(I_{i-k})$ for $i \in \{k+l, \ldots, 2k\}$. By Observation 2 intervals I_i and I_{i-k} do not intersect. This completes the proof. ◀

Observe that Observation 3 and Observation 2 give an alternative to [24] algorithm that completes the prefix coloring on unit interval graphs. We refer to this algorithm as Greedy Color Completion (even though we can use the algorithm of [24] instead).

3 Overview of our results and techniques

Our final result is based on new techniques that might be of independent interest. This section outlines our techniques and gives an overview on how they are combined into obtaining the final result. Our techniques are encapsulated in Section 3.1 and Section 3.2 as completely independent results, as we see the potential of applying them to a wider range of problems. In Section 3.1 we introduce our new technique of *color sorting*, which allows to solve the Unit Precoloring Extension problem efficiently under a specific condition that might naturally appear in a number of applications. In Section 3.2 we introduce the Frogs game technique, which is a natural generalization of the folklore technique applied for merging sets in Find-Union like algorithms. We expect the Frogs game technique to be also applicable to a wider range of problems. Our main application of the color sorting technique and the Frogs game technique is presented in Section 3.3, where we introduce the Incremental Unit Interval Recoloring problem and sketch the solution to this problem.

3.1 Color sorting applied to the Unit Precoloring Extension problem

Our first contribution is the color sorting technique, that we apply to the Unit Precoloring Extension problem. In this problem we are given a k-colorable multiset of unit intervals $\mathcal{I} = \{I_1 \sqsubset \ldots \sqsubset I_m\}$, $m > 2k$. We are also given a proper k-coloring c' on $\mathsf{prefix}^{\sqsubseteq}_{\mathcal{I}}(I_k)$ (the first k intervals) and a proper k-coloring c'' on $\mathsf{suffix}^{\sqsubseteq}_{\mathcal{I}}(I_{m-k+1})$ (the last k intervals). The problem is to extend c' and c'' to a proper l-coloring of $\mathcal{I}$ minimizing l.

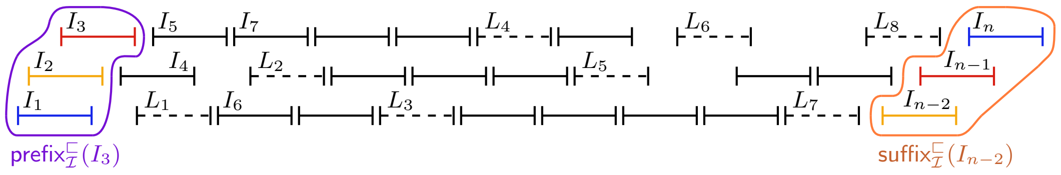
The Unit Precoloring Extension problem was proven NP-hard [22], moreover, it is even NP-hard to decide whether one can extend the coloring using $l = k$ colors. In our application, however, we are only interested in the instances when this is possible, i.e., the Unit Precoloring Extension ceases to be NP-hard. We specify a condition on $\mathcal{I}$ under which one can, using the color sorting technique, extend the precoloring to a proper k-coloring. Our condition essentially requires that there is some slack space between the colored prefix and the colored suffix, which allows color sorting. By the slack we mean that we can fit between the prefix and the suffix additional $k^2 - 1$ mutually disjoint unit intervals without increasing the chromatic number of $\mathcal{I}$. This slack is used to gradually sort the color permutation given on the prefix to finally obtain the color permutation given on the suffix, in an insertion sort fashion. The precise result is stated in the following lemma.

▶ **Lemma 5** (Precoloring Extension Lemma). *Let k be an integer and let $\mathcal{I} = \{I_1 \sqsubset \ldots \sqsubset I_m\}$ be a k-colorable multiset of $m \geqslant 2k$ unit intervals. Let $c' : \mathsf{prefix}^{\sqsubseteq}_{\mathcal{I}}(I_k) \mapsto [k]$ and $c'' : \mathsf{suffix}^{\sqsubseteq}_{\mathcal{I}}(I_{m-k+1}) \mapsto [k]$ be bijections. Let there exist a set of $k^2 - 1$ pairwise non-intersecting unit intervals $\mathcal{L}$, such that $\mathsf{prefix}^{\sqsubseteq}_{\mathcal{I}}(I_k) \sqsubset \mathcal{L} \sqsubset \mathsf{suffix}^{\sqsubseteq}_{\mathcal{I}}(I_{m-k+1})$ and $\mathcal{I} \cup \mathcal{L}$ is k-colorable. Then there is a proper k-coloring $c : \mathcal{I} \mapsto [k]$ such that c extends both c' and c'', i.e., $c(I_i) = c'(I_i)$ for $i \in [k]$ and $c(I_i) = c''(I_i)$ for $i \in \{m-k+1, \ldots, m\}$.*

The full proof of Lemma 5 can be found in the full version of the paper [11], Appendix A.

Sketch of proof of Lemma 5. The proof introduces the color sorting technique, which uses the slack intervals of $\mathcal{L}$ to gradually transform the coloring c' into the coloring c'', in an insertion sort like fashion. In order to accomplish that, we modify the instance (by inserting dummy intervals from $\mathcal{L}$) to allow the color sorting technique to work. We describe this process first.

We start by assuming that the multiset of unit intervals $\mathcal{I}$ is connected as a graph (otherwise Lemma 5 follows from the Modulo Color Completion algorithm, see Observation 3). An example instance for which our assumptions hold is pictured in Figure 1.



I_3
I_2
I_1
I_5
I_7
I_4
L_1
L_2
I_6
L_3
L_4
L_5
L_6
L_7
L_8
I_n
I_{n-1}
I_{n-2}
$\text{prefix}^{\sqsubseteq}_{\mathcal{I}}(I_3)$
$\text{suffix}^{\sqsubseteq}_{\mathcal{I}}(I_{n-2})$

Figure 1 An example illustrating the assumptions of Lemma 5 for $k = 3$.

We now partition $\mathcal{L} = \{L_1 < \ldots < L_{k^2-1}\}$ into $2k - 1$ sets of unit intervals as follows: $\mathcal{L} = \mathcal{L}_1 \cup \{M_1\} \cup \mathcal{L}_2 \cup \ldots \cup \{M_{k-1}\} \cup \mathcal{L}_k$, where $\mathcal{L}_i = \{L_{(i-1)k+1}, \ldots, L_{ik-1}\}$ for $i \in [k]$ and $M_i = L_{ik}$ for $i \in [k-1]$. Observe that $|\mathcal{L}_i| = k - 1$. Thus, the intervals of the partition are ordered as follows: $\text{prefix}^{\sqsubseteq}_{\mathcal{I}}(I_k) \sqsubset \mathcal{L}_1 < \{M_1\} < \mathcal{L}_2 < \ldots < \{M_{k-1}\} < \mathcal{L}_k \sqsubset \text{suffix}^{\sqsubseteq}_{\mathcal{I}}(I_{n-k+1})$. Next we partition $\mathcal{I}$ into k parts: $\mathcal{I} = \mathcal{I}_1 \cup \mathcal{I}_2 \cup \ldots \cup \mathcal{I}_k$, in the way that the following holds: $\text{prefix}^{\sqsubseteq}_{\mathcal{I}}(I_k) \subseteq \mathcal{I}_1 \sqsubset \{M_1\} \sqsubset \mathcal{I}_2 \sqsubset \ldots \sqsubset \{M_{k-1}\} \sqsubset \mathcal{I}_k \supseteq \text{suffix}^{\sqsubseteq}_{\mathcal{I}}(I_{n-k+1})$. This partition is pictured in Figure 2. We now enlarge $\mathcal{I}$ by adding to each $\mathcal{I}_i$ some dummy intervals. More

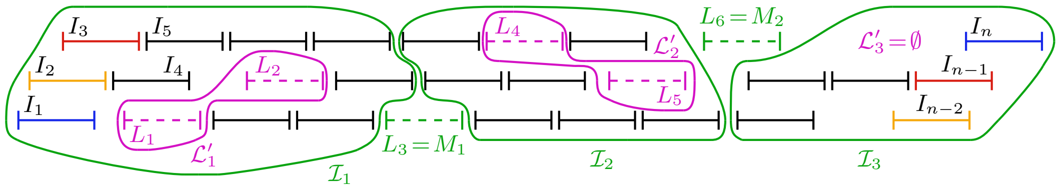


Figure 2 An example illustrating partitioning the intervals, $k = 3$.

precisely, to each $\mathcal{I}_i \subseteq \mathcal{I}$ we add a subset $\mathcal{L}'_i \subseteq \mathcal{L}_i$ as to make the number of intervals in the extension $\mathcal{J}_i \overset{\text{def}}{=} \mathcal{I}_i \cup \mathcal{L}'_i$ a multiple of k (see Figure 2). As a consequence, $\mathcal{J} \overset{\text{def}}{=} \bigcup_{i=1}^{k} \mathcal{J}_i$ and $\mathcal{J}_1, \mathcal{J}_2, \ldots, \mathcal{J}_k$ satisfy the following (note that since $\mathcal{I}$ is connected as a graph, each $\mathcal{I}_i$ is nonempty and thus each $\mathcal{J}_i$ is also nonempty):

1. $\omega(\mathcal{J} \cup \{M_1, \ldots, M_{k-1}\}) \leqslant k$,
2. for each $i \in [k]$ we have $|\mathcal{J}_i| = kp_i$ for some $p_i \in \mathbb{N} \setminus \{0\}$,
3. $\text{prefix}^{\sqsubseteq}_{\mathcal{I}}(I_k) \subseteq \mathcal{J}_1 \sqsubset \{M_1\} \sqsubset \mathcal{J}_2 \sqsubset \ldots \sqsubset \{M_{k-1}\} \sqsubset \mathcal{J}_k \supseteq \text{suffix}^{\sqsubseteq}_{\mathcal{I}}(I_{n-k+1})$.

Now, rather than the proper coloring for $\mathcal{I}$, we construct the proper coloring c for $\mathcal{J}$ (which is obviously also proper for $\mathcal{I}$). The construction of c starts by copying colors of c', so that c coincides with c' on $\text{prefix}^{\sqsubseteq}_{\mathcal{I}}(I_k)$. Let us now consider a block $\mathcal{J}_{j-1} = \{J_0^{j-1} \sqsubset \ldots \sqsubset J_{s_{j-1}-1}^{j-1}\}$, where $(j-1) \in [k]$, $s_{j-1} = |\mathcal{J}_{j-1}|$. Suppose that c is already defined on $\text{prefix}^{\sqsubseteq}_{\mathcal{J}_{j-1}}(J_{k-1}^{j-1})$ (the first k intervals of $\mathcal{J}_{j-1}$). Note that coloring c on $\text{prefix}^{\sqsubseteq}_{\mathcal{J}_{j-1}}(J_{k-1}^{j-1})$ defines a permutation of colors. We use the Modulo Color Completion algorithm to copy this permutation to $\text{suffix}^{\sqsubseteq}_{\mathcal{J}_{j-1}}(J_{s_{j-1}-k}^{j-1})$ (the last k intervals of $\mathcal{J}_{j-1}$). This works because $k | s_{j-1}$. Suppose that this permutation already coincides with the permutation given by c'' on the last $j-2$ positions. We now use the slack given by M_{j-1} to define c on $\mathsf{prefix}^{\sqsubset}_{\mathcal{J}_j}(J^j_{k-1})$ in such a way that the corresponding permutation agrees with the permutation given by c'' on the last $j-1$ positions. To be more precise, we apply one step of the insertion sort on the permutation to bring one more color to the appropriate position, as shown in Figure 3. This is possible because of the slack given by M_{j-1}. The intuition is that if we apply Modulo Color Completion on $\mathsf{prefix}^{\sqsubset}_{\mathcal{J}_j}(J^j_{k-1})$, then we obtain a proper k-coloring on $\mathsf{prefix}^{\sqsubset}_{\mathcal{J}_j}(J^j_{k-1})$ that precisely copies the color permutation from $\mathsf{suffix}^{\sqsubset}_{\mathcal{J}_{j-1}}(J^{j-1}_{s_{j-1}-k})$. On the other hand, if we apply Modulo Color Completion to $\{M_{j-1}\} \cup \mathsf{prefix}^{\sqsubset}_{\mathcal{J}_j}(J^j_{k-1})$, then on $\mathsf{prefix}^{\sqsubset}_{\mathcal{J}_j}(J^j_{k-1})$ we get the same permutation with all colors shifted down by one, and this is also a proper coloring. The insertion sort step that we apply (see Figure 3) gives a permutation that alternates at most twice between the same permutation and the shifted permutation. Also, the insertion sort step moves one special color (purple in Figure 3) further away than required by the Modulo Color Completion algorithm. Hence the obtained coloring is proper. ◀

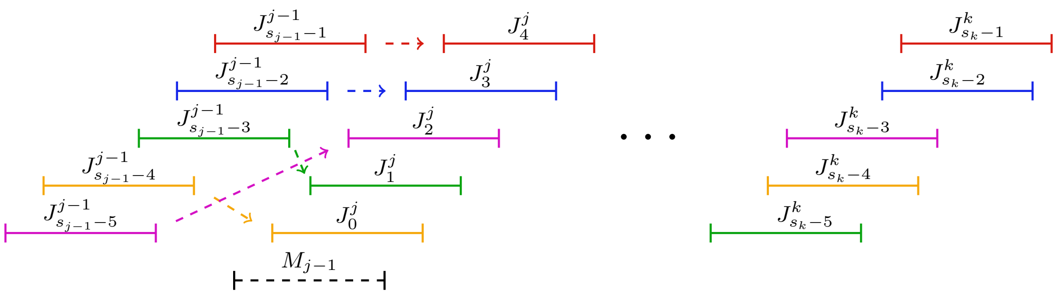


Figure 3 An insertion sort step, where coloring c is constructed on $\mathsf{prefix}^{\sqsubset}_{\mathcal{J}_j}(J^j_{k-1})$ for $j = 4$, $k = 5$.

3.2 The Frogs game

Our second contribution is a technique that generalizes a folklore trick typically used in the analysis of Union-Find data structure. The Set Union problem, where the Union-Find data structure finds its application, can be thought of as a game. In this game we are initially given n pairwise disjoint sets of size δ, and the adversary keeps merging consecutive pairs of sets until there is only one set left. Each time the adversary merges a pair of sets, we incur the cost equal to the size of the smaller set among the ones being merged (as if we move all elements of the smaller set to the larger set, one by one). It is clear that the maximum total cost the adversary can generate is $\delta n \log n$, as each of δn elements can contribute to the total cost at most $\log n$ times.

The generalization we propose is that the cost we incur with each merging is the sum of the sizes of κ consecutive sets rather then just one. We sum κ consecutive sizes either to the left or to the right of the merged pair, whatever turns out cheaper. We show that the total cost an adversary can generate here is $\mathcal{O}(\delta\kappa n \log n)$. We refer to this generalization as the Frogs game, and we now introduce its formal definition.

▶ **Definition 6.** *We define an instance of a* Frogs *game as a tuple $\mathcal{F} = (N, \kappa, \delta, J)$. There, N,κ, and δ are integers, referred to as the size of the game, the jump number, and the initial rank value respectively. It also holds that $\kappa \leqslant N$. Moreover J is a sequence of $N-1$ integers $J = (j_1, j_2, \ldots, j_{N-1})$, where $1 \leqslant j_\tau \leqslant N - \tau$ and J is referred to as the jump sequence.*

We now define the cost of the FROGS game.

▶ **Definition 7.** *Let $\mathcal{F} = (N, \kappa, \delta, J)$ be an instance of the FROGS game, where $J = (j_1, j_2, \ldots, j_{N-1})$. Let $R_1 = (\delta, \ldots, \delta)$ be a sequence of length N (R_1 is called the initial rank sequence). Let $R_{\tau+1} \overset{\text{def}}{=} (r_{\tau,1},\ \ldots,\ r_{\tau,j_\tau-1},\ r_{\tau,j_\tau} + r_{\tau,j_\tau+1},\ r_{\tau,j_\tau+2},\ \ldots,\ r_{\tau,N-\tau+1})$ (rank sequence $R_{\tau+1}$ in time $\tau+1$ is obtained by adding two neighboring ranks in R_τ placed at positions j_τ and $j_\tau+1$). The FROGS cost incurred in time τ is $\varsigma_\tau \overset{\text{def}}{=} \min(r_{\tau,j_{\tau-\kappa+1}} + \ldots + r_{\tau,j_\tau},\ r_{\tau,j_{\tau+1}} + \ldots + r_{\tau,j_{\tau+\kappa}})$, where for $i < 1$ and $i > N - \tau + 1$ we set $r_{\tau,i} \overset{\text{def}}{=} \delta$. The total cost of the FROGS game $\mathcal{F}$ is $\varsigma(\mathcal{F}) \overset{\text{def}}{=} \varsigma_1 + \ldots + \varsigma_{N-1}$.*

We bound the cost of any FROGS game $\mathcal{F}$ using the following theorem.

▶ **Theorem 8** (Frogs theorem). *For an instance $\mathcal{F} = (N, \kappa, \delta, J)$ of the FROGS game we have*

$$\varsigma(\mathcal{F}) \leqslant \delta(2\kappa - 1)(N + 2\kappa - 2) \log_2 \frac{N + 2\kappa - 2}{2\kappa - 1}.$$

The FROGS theorem is proved in the full version of the paper [11] in Appendix B. The technique given by FROGS theorem might be of independent interest. It seems applicable as a building block to more problems than the one studied further in this paper.

3.3 Incremental Unit Interval Recoloring problem

The main result of this paper is an algorithm for the INCREMENTAL UNIT INTERVAL RECOLORING problem. We are given a parameter k and a sequence of n unit intervals: $I_1, I_2, \ldots, I_n$ such that $\{I_1, \ldots, I_n\}$ is k-colorable. This sequence defines $n+1$ multisets of unit intervals: $\mathcal{I}_0 = \emptyset$ and $\mathcal{I}_j \overset{\text{def}}{=} \{I_1, \ldots, I_j\}$ for $j > 0$. Instance $\mathcal{I}_j$ differs from $\mathcal{I}_{j-1}$ by one interval I_j. The goal is to maintain a proper k-coloring on the dynamic instance. To be more precise, after the interval I_j is presented, the algorithm needs to compute a proper k-coloring c_j for $\mathcal{I}_j$. Our objective is to minimize the recourse budget, which is the number of intervals with different colors in c_j and c_{j-1}. We obtain the following result.

▶ **Theorem 9.** *There is an algorithm for the INCREMENTAL UNIT INTERVAL RECOLORING problem with a total recourse budget of $\mathcal{O}(k^7 n \log n)$.*

Theorem 9 is formally proved in the full version of the paper [11], Appendix C. Here we outline how the Precoloring Extension lemma (Lemma 5) and the Frogs theorem (Theorem 8) are combined to obtain the main result, skipping the technical details that are deferred to the full proof.

Sketch of the proof of Theorem 9. Let us imagine that a new interval I_j is presented and we need to provide the coloring c_j for $\mathcal{I}_j$. Let us order $\mathcal{I}_j$ according to the $\sqsubset$ order. Let I be the direct predecessor (in $\sqsubset$ order) of I_j in $\mathcal{I}_j$. Let interval $J^{(R)}$ be the interval of $\mathcal{I}_j$ succeeding I (in $\sqsubset$ order) such that between I and $J^{(R)}$ there is room to insert the set $\mathcal{L}$ of $k^2 - 1$ mutually disjoint unit intervals, as required by the Precoloring Extension lemma (Lemma 5). Let $J^{(L)}$ be an analogous interval preceding I. For simplicity let us assume that both $J^{(L)}$ and $J^{(R)}$ exist. Based on the Precoloring Extension lemma (Lemma 5), it is (almost) sufficient to recolor the infix of $\mathcal{I}_j$ between I and $J^{(R)}$, or the infix of $\mathcal{I}_j$ between $J^{(L)}$ and I, and it is up to the algorithm which of the two infixes it chooses to recolor. Our recoloring algorithm basically chooses the infix that is smaller in size and recolors it (although there are small technical details that make the algorithm a bit more complicated in the end).

The whole weight of the proof lies now in the analysis, which shows that the size of the smaller infix is $\mathcal{O}(k^7 \log n)$ in the amortized sense. To prove that we use the Frogs theorem (Theorem 8), so we need to define the appropriate size N, jump number κ, initial rank value δ, and the jump sequence J, and all these numbers must strongly relate to what the recoloring algorithm does. We sketch here the main idea of how we do it. We look at all k-cliques of the final instance $\mathcal{I}_{n'}$, $n' \in \mathcal{O}(kn)$ (the final instance is not $\mathcal{I}_n$ because, for a technical reason mentioned further, we may be forced to add some dummy intervals after the whole instance $\mathcal{I}_n$ was presented). Every k-clique is a set of k overlapping intervals, whose intersection is some (not necessarily unit) interval. So we represent the k-cliques of the final instance $\mathcal{I}_{n'}$ as a set $\mathcal{S}_{n'}$ of intervals, which are mutually disjoint, and we order them by $<$ order. This is shown in Figure 4, where at the bottom the intervals of $\mathcal{I}_{n'}$ are drawn (either solid black or dashed blue), and above them the corresponding cliques of $\mathcal{S}_{n'}$ are pictured (also either solid black or dashed blue).

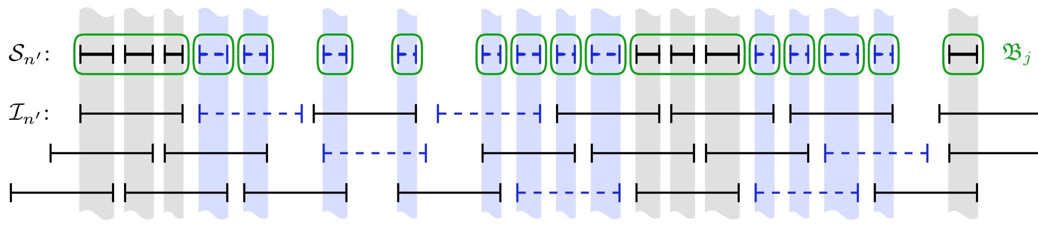


Figure 4 Example span partition : ⊢—⊣ – intervals of $\mathcal{I}_j$, ⊢- - -⊣ – intervals of $\mathcal{I}_{n'} \setminus \mathcal{I}_j$ (future intervals), ⊢—⊣ – spans of $\mathcal{S}_j$, ⊢- - -⊣ – spans of $\mathcal{S}_{n'} \setminus \mathcal{S}_j$ (future spans), ▭ – blocks of partition $\mathfrak{B}_j$, $k = 3$.

We observe that $m \overset{\text{def}}{=} |\mathcal{S}_{n'}| = \mathcal{O}(kn)$. We assume that $\mathcal{S}_{n'}$ is tightly packed, i.e., two consecutive intervals in $\mathcal{S}_{n'}$ are at distance less than one (for this assumption to hold we add dummy intervals to $\mathcal{I}_n$ and obtain $\mathcal{I}_{n'}$-whenever two consecutive cliques are at distance at least one, we insert a dummy interval between them without increasing the chromatic number). The intuition now is that at the beginning each clique in $\mathcal{S}_{n'}$ is empty, but successively the cliques in $\mathcal{S}_{n'}$ are filled with the intervals, until each of them is filled up to the maximum level k. A new interval I_j adds 1 to the level of all the cliques it intersects. Let us denote by $\mathsf{lvl}_j(S)$ the level of clique $S \in \mathcal{S}_{n'}$ in step j. In Figure 4, the intervals of $\mathcal{I}_j$ are marked solid black, while the future intervals (not presented until step j) are marked dashed blue. Consequently, the fully filled cliques of $\mathcal{S}_{n'}$ are marked solid black, while the cliques that are not filled to the maximum level in step j are marked dashed blue.

In each step j, we partition the cliques $S \in \mathcal{S}_{n'}$ into blocks depending on their level $\mathsf{lvl}_j(S)$. We refer to the corresponding partition as $\mathfrak{B}_j$. The rule is that the consecutive cliques who are entirely filled (meaning that $\mathsf{lvl}_j(S) = k$) belong to the same block of the partition $\mathfrak{B}_j$, while the cliques that are not filled are placed in a separate one-element block. An example partition $\mathfrak{B}_j$ is pictured in Figure 4. As a result, the blocks of $\mathfrak{B}_j$ are merged over the time, but never split. The grey blocks of $\mathfrak{B}_j$ consisting of entirelly filled cliques are called passive, while the remaining blue blocks (the one-element blocks containing the cliques that are not filled) are called active.

Now for each insertion step j there is a corresponding rank sequence R_τ in the Frogs game. Each block $\mathcal{B} \in \mathfrak{B}_j$ is assigned at least one and at most $(k+1)$ consecutive ranks $r^{(\mathcal{B})}_{\tau,i}$ in R_τ, in the way that every rank $r^{(\mathcal{B})}_{\tau,i}$ assigned to $\mathcal{B}$ bounds the sum of the levels in $\mathcal{B}$, i.e., $r^{(\mathcal{B})}_{\tau,i} \geqslant \sum_{S \in \mathcal{B}} \mathsf{lvl}_j(S)$. Each active block $\mathcal{B}$ is assigned at least $k - \mathsf{lvl}_j(S) + 1$ consecutive ranks $r^{(\mathcal{B})}_{\tau,i}$, while each passive block is assigned precisely one rank.

When the interval I_j arrives, some cliques $S \in \mathcal{S}_{n'}$ increase their level. Such a clique S that increases its level is necessarily in an active (singleton) block $\mathcal{B} = \{S\}$, as I_j certifies that S is not completely filled up yet. To account for the insertion of I_j, we merge any two consecutive ranks assigned to $\mathcal{B} = \{S\}$ (there is at least $k - \mathsf{lvl}_j(S) + 1 \geqslant 2$ ranks available). We observe that the infix recolored by the algorithm is entirely contained within $\mathcal{O}(k^3)$ consecutive blocks to the left or to the right of $\mathcal{B}$, and in what follows we argue why this holds. First of all, the recolored infix can intersect at most $\mathcal{O}(k^3)$ active blocks: each active block certifies that a future unit interval fits in, but k active blocks may be witnesses for the same future unit interval that fits in. Note that the infix recolored by the algorithm contains at most $(k^2 - 1)$ pairwise disjoint future intervals. Second of all, the recolored infix intersects no more passive blocks than active blocks, since each passive block has a neighboring active block.

Since every block is assigned at most $(k+1)$ consecutive ranks, if we set the jump number $\kappa = \mathcal{O}(k^4)$, the cost incurred in the Frogs game covers the ranks assigned to the $\mathcal{O}(k^3)$ consecutive blocks, which in turn bound the recoloring cost in the recoloring algorithm for step j. Of course, upon the arrival of I_j some cliques need to be merged into one passive block, and even some passive blocks get merged together, meaning that we need to merge some extra ranks in the rank sequence, but this also can be handled by the Frogs game. ◀

3.4 Fully Dynamic Unit Interval Recoloring problem

In this section we show that for the Fully Dynamic Unit Interval Recoloring problem one cannot hope for an algorithm with a reasonably limited recourse budget. In the Fully Dynamic Unit Interval Recoloring problem, we are initially given an empty multiset of unit intervals $\mathcal{I}_0 = \emptyset$. The instance $\mathcal{I}_{j+1}$ is obtained from $\mathcal{I}_j$ by adding a new unit interval to $\mathcal{I}_j$ or removing the existing chosen interval from $\mathcal{I}_j$. Every instance $\mathcal{I}_j$ presented to the algorithm is k-colorable. The goal is again to maintain a proper k-coloring on the dynamic instance. After each interval insertion and removal, the algorithm needs to compute a proper k-coloring c_j for $\mathcal{I}_j$. Similarly as before, our objective is to minimize the recourse budget, which is the number of intervals with different colors in c_j and c_{j-1}. For this problem we get the following negative result.

▶ **Observation 10.** *There is a sequence of m updates for the Fully Dynamic Unit Interval Recoloring problem that forces the total number of recolorings of $\Omega(m^2)$.*

Proof. We first construct an instance $\mathcal{M} = \mathcal{I} \cup \mathcal{J} \cup \mathcal{K} \cup \mathcal{L}$, where $\mathcal{I} = \{I_1, \ldots, I_n\}$, $\mathcal{J} = \{J_1, \ldots, J_n\}$, $\mathcal{K} = \{K_1, \ldots, K_n\}$ and $\mathcal{L} = \{L_1, \ldots, L_n\}$. All four sets contain pairwise disjoint intervals. Both $\mathcal{K} \cup \mathcal{L}$ and $\mathcal{I} \cup \mathcal{J}$ are paths when interpreted as graphs. Additionally, set $\mathcal{K} \cup \mathcal{L}$ is placed to the left of $\mathcal{I} \cup \mathcal{J}$. This is shown in Figure 5.

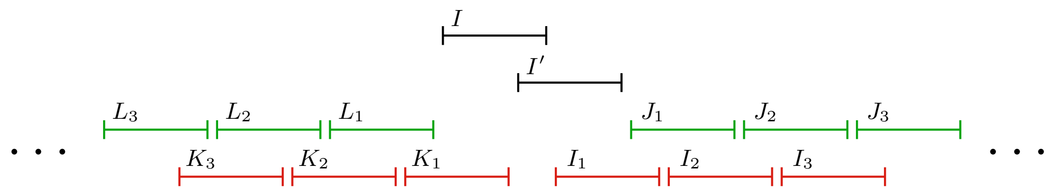


Figure 5 The update when K_1 has the same color as I_1.

Observe that since $\mathcal{M}$ is 2-colorable, all sets $\mathcal{K}$, $\mathcal{L}$, $\mathcal{I}$ and $\mathcal{J}$ are necessarily monochromatic, moreover set $\mathcal{K}$ has different color than $\mathcal{L}$, and $\mathcal{I}$ has different color than $\mathcal{J}$. We construct the instance so, that K_1 is the rightmost interval of $\mathcal{K} \cup \mathcal{L}$ and interval I_1 is the leftmost interval of $\mathcal{I} \cup \mathcal{J}$. Moreover, the distance between the end coordinate of K_1 and the begin coordinate of I_1 is less than one (see Figure 5). Consider the case when K_1 has the same color as I_1. Then we insert to $\mathcal{M}$ two intersecting intervals I and I', such that I intersects K_1 and I' intersects I_1 (see Figure 5). This results in $\mathcal{M} \cup \{I, I'\}$ being a path when interpreted as a graph. The instance is still 2-colorable, but now $\mathcal{I}$ needs to have the same color as $\mathcal{L}$ and $\mathcal{J}$ needs to have the same color as $\mathcal{K}$. This requires recoloring $2n$ intervals, since either $\mathcal{I} \cup \mathcal{J}$ or $\mathcal{K} \cup \mathcal{L}$ has to be recolored. Next, we remove I and I'. Consider now the case when K_1 has a different color than I_1 (see Figure 6). In that case we insert an interval I intersecting

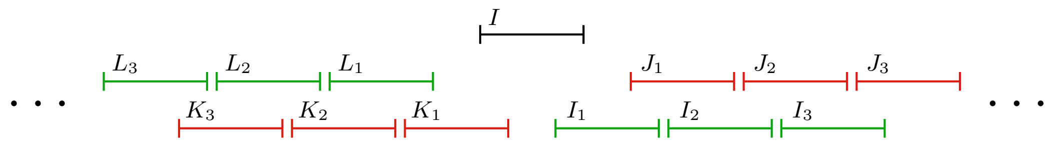


Figure 6 The update when K_1 has different color than I_1.

both K_1 and I_1. This causes that K_1 and I_1 have to now be assigned different colors, and again either $\mathcal{I} \cup \mathcal{J}$ or $\mathcal{K} \cup \mathcal{L}$ has to be recolored. ◀

3.5 Coloring Unit Circular Arc Graphs

In this section we present a different application of the Precoloring Extension Lemma (Lemma 5). As a result, we offer a new combinatorial result for coloring unit circular arc graphs.

Unit circular arc graphs, as a subclass of proper circular arc graphs, admit an $\mathcal{O}(n^{1.5})$ algorithm that statically finds an optimum proper coloring [28]. On the other hand, the problem of coloring circular arc graphs is NP-hard [16], and a lot of research has been devoted to find positive results regarding this problem. This line of research exposes two important parameters describing an instance $\mathcal{A}$ of a circular arc graph: the load $\text{load}(\mathcal{A})$ and the cover number $\text{cn}(\mathcal{A})$. The load $\text{load}(\mathcal{A})$ stands for the maximum number of arcs intersecting in one point, whereas the cover number $\text{cn}(\mathcal{A})$ stands for the minimum number of arcs covering the whole circle. The research focus is on the conditions under which a circular arc graph admits a proper coloring using close to $\text{load}(\mathcal{A})$ colors. Tucker [30] shows, that if $\text{cn}(\mathcal{A}) \geqslant 4$, then $\lfloor 3\,\text{load}(\mathcal{A})/2 \rfloor$ colors suffice. Valencia-Pabon [31] shows that if $\text{cn}(\mathcal{A}) \geqslant 5$, then $\left\lceil \frac{\text{cn}(\mathcal{A})-1}{\text{cn}(\mathcal{A})-2}\,\text{load}(\mathcal{A}) \right\rceil$ colors is enough. For $\text{cn}(\mathcal{A}) \geqslant \text{load}(\mathcal{A}) + 2$ the bound becomes $\text{load}(\mathcal{A}) + 1$. Belkale and Chandran [4] prove the Hadwiger's conjecture for proper circular arc graphs. Neither of these results exposes a condition sufficient to color the instance with precisely $\text{load}(\mathcal{A})$ colors. We use Lemma 5 to show, that if one can add $\text{load}(\mathcal{A})^2 - 1$ non-intersecting unit arcs to an instance of a unit circular arc graph in a way that the load does not increase, then $\text{load}(\mathcal{A})$ colors is sufficient to properly color the instance. This is formalized by the following lemma, proved in the full version [11] in Appendix D.

▶ **Theorem 11.** *Let $\mathcal{A}$ be a set of unit circular-arcs on the circle with a circumference at least 2, such that $\mathcal{A}$ can be extended with $r = \text{load}(\mathcal{A})^2 - 1$ not intersecting unit circular-arcs $B_1, \ldots, B_r$ which do not increase the load, i.e., $\text{load}(\mathcal{A}) = \text{load}(\mathcal{A} \cup \{B_1, \ldots, B_r\})$. Then $\chi(\mathcal{A}) \leqslant \text{load}(\mathcal{A})$.*

Note that our condition of Theorem 11 can be easily checked in linear time and it might be a very natural assumption for some applications.

4 Concluding remarks

In this section we discuss several interesting extensions and aspects of our main result, which is the recoloring algorithm for the Incremental Unit Interval Recoloring problem. We conclude with future research directions and open problems.

The first interesting property of the Recolor algorithm is its resistance to malicious coloring. By that we mean, that the Recolor algorithm on its input coloring does not assume anything other than being a proper k-coloring. In other words, some evil adversary could potentially repaint the whole instance before inserting the new interval, and as long as this is a proper k-coloring, the Recolor algorithm works. This might be of interest for some applications, one of which is described next.

Throughout the paper we make a simplifying assumption that the chromatic number k of the final instance is given apriori to the Recolor algorithm (as a parameter). Using the fact that our algorithm works against malicious coloring, it is easy to get rid of this assumption by running the recoloring algorithm with changing values of k. To be more precise, let us consider each step j when the chromatic number increases: $l-1 = \chi(\mathcal{I}_{j-1}) < \chi(\mathcal{I}_j) = l$ (we can easlily detect all such steps j). Before I_j arrives, we have the $(l-1)$-coloring c_{j-1} of $\mathcal{I}_{j-1}$ at our disposal. Coloring c is obviously also an l-coloring for $\mathcal{I}_{j-1}$, so the recoloring algorithm for $\mathcal{I}_j$ gets the correct input. From now on, until the next step when the chromatic number increases, the algorithm runs with parameter $k = l$. It is easy to see that the algorithm modified in this way returns the proper coloring. Let K be the chromatic number of the final instance $\mathcal{I}_n$ (not known to the algorithm). Since for every k used by the modified algoritm we have $k \leqslant K$, the analysis for parameter K bounds from above the total recoloring budget of the modified algorithm. Thus, using $\mathcal{O}(K^7 \log n)$ amortized recoloring budget, we can maintain an optimal coloring for each instance $\mathcal{I}_j$.

Our incremental algorithm can be implemented in total time $\mathcal{O}(k^7 n \log n)$, where k is the final chromatic number. It suffices to maintain a sorted list of intervals $\mathcal{I}_j$ in an AVL tree. Such a list allows inserting a new interval in time $\mathcal{O}(\log n)$. It allows finding the predecessor and the successor of the newly inserted interval, and efficient iteration to the left and to the right. This allows detecting the infix that we want to recolor in the time proportional to the size of the infix. The coloring step can then be performed in linear time. We refer to the full version [11] of the paper for the details and the pseudocode of the algorithm.

Finally, let us shortly discuss why our approach does not seem to extend to general interval graphs, or even the intervals whose lengths vary from 1 to $(1+\epsilon)$. The main reason for that is that the Modulo Color Completion algorithm spectacularly fails on such graphs. In particular, Observation 2 ceases to hold, and the Modulo Color Completion algorithm is based on this observation. It would be interesting to see if the color sorting technique could work with some algorithms other than Modulo Color Completion , effective on any superclass of unit interval graphs. We leave this as the main open question. Note that due to our lower bound in Section 3.4, which carries over to any superclass of unit interval graphs, we cannot hope on positive results regarding the fully dynamic setting.

References

1 Josh Alman, Matthias Mnich, and Virginia Vassilevska Williams. Dynamic parameterized problems and algorithms. In *Proceedings of the 44th International Colloquium on Automata, Languages, and Programming, ICALP 2017*, volume 80 of *LIPIcs*, pages 41:1–41:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2017. `doi:10.4230/LIPIcs.ICALP.2017.41`.

2 Luis Barba, Jean Cardinal, Matias Korman, Stefan Langerman, André van Renssen, Marcel Roeloffzen, and Sander Verdonschot. Dynamic graph coloring. In *Algorithms and data structures*, volume 10389 of *Lecture Notes in Comput. Sci.*, pages 97–108. Springer, Cham, 2017. doi:10.1007/978-3-319-62127-2_9.

3 Dwight R. Bean. Effective coloration. *The Journal of Symbolic Logic*, 41(2):469–480, 1976. URL: http://www.jstor.org/stable/2272247.

4 Naveen Belkale and L. Sunil Chandran. Hadwiger's conjecture for proper circular arc graphs. *European J. Combin.*, 30(4):946–956, 2009. doi:10.1016/j.ejc.2008.07.024.

5 Aaron Bernstein, Jacob Holm, and Eva Rotenberg. Online bipartite matching with amortized $O(\log^2 n)$ replacements. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 947–959. SIAM, Philadelphia, PA, 2018. doi:10.1137/1.9781611975031.61.

6 Bartłomiej Bosek, Stefan Felsner, Kamil Kloch, Tomasz Krawczyk, Grzegorz Matecki, and Piotr Micek. On-line chain partitions of orders: a survey. *Order*, 29(1):49–73, 2012. doi:10.1007/s11083-011-9197-1.

7 Bartłomiej Bosek, Dariusz Leniowski, Piotr Sankowski, and Anna Zych. Online bipartite matching in offline time. In *55th Annual IEEE Symposium on Foundations of Computer Science – FOCS 2014*, pages 384–393. IEEE Computer Soc., Los Alamitos, CA, 2014. doi:10.1109/FOCS.2014.48.

8 Bartłomiej Bosek, Dariusz Leniowski, Piotr Sankowski, and Anna Zych-Pawlewicz. Shortest augmenting paths for online matchings on trees. *Theory Comput. Syst.*, 62(2):337–348, 2018. doi:10.1007/s00224-017-9838-x.

9 Bartłomiej Bosek, Dariusz Leniowski, Piotr Sankowski, and Anna Zych-Pawlewicz. A tight bound for shortest augmenting paths on trees. In *LATIN 2018: Theoretical informatics*, volume 10807 of *Lecture Notes in Comput. Sci.*, pages 201–216. Springer, Cham, 2018. doi:10.1007/978-3-319-77404-6_1.

10 Bartłomiej Bosek, Yann Disser, Andreas Emil Feldmann, Jakub Pawlewicz, and Anna Zych-Pawlewicz. Recoloring Interval Graphs with Limited Recourse Budget. In Susanne Albers, editor, *17th Scandinavian Symposium and Workshops on Algorithm Theory (SWAT 2020)*, volume 162 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 17:1–17:23, Dagstuhl, Germany, 2020. Schloss Dagstuhl–Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.SWAT.2020.17.

11 Bartłomiej Bosek and Anna Zych-Pawlewicz. Recoloring unit interval graphs with logarithmic recourse budget, 2022. doi:10.48550/ARXIV.2202.08006.

12 Jiehua Chen, Wojciech Czerwiński, Yann Disser, Andreas Emil Feldmann, Danny Hermelin, Wojciech Nadara, Marcin Pilipczuk, Michał Pilipczuk, Manuel Sorge, Bartłomiej Wróblewski, and Anna Zych-Pawlewicz. *Efficient fully dynamic elimination forests with applications to detecting long paths and cycles*, pages 796–809. SIAM, 2021. doi:10.1137/1.9781611976465.50.

13 Mitre Costa Dourado, Van Bang Le, Fábio Protti, Dieter Rautenbach, and Jayme Luiz Szwarcfiter. Mixed unit interval graphs. *Discret. Math.*, 312:3357–3363, 2012.

14 Zdeněk Dvořák, Martin Kupec, and Vojtěch Tůma. A dynamic data structure for MSO properties in graphs with bounded tree-depth. In *Proceedings of the 22nd Annual European Symposium on Algorithms, ESA 2014*, volume 8737 of *Lecture Notes in Computer Science*, pages 334–345. Springer, 2014. doi:10.1007/978-3-662-44777-2_28.

15 Leah Epstein and Meital Levy. Online interval coloring and variants. In *Proceedings of the 32nd International Conference on Automata, Languages and Programming*, ICALP'05, pages 602–613, Berlin, Heidelberg, 2005. Springer-Verlag. doi:10.1007/11523468_49.

16 M. R. Garey, D. S. Johnson, G. L. Miller, and C. H. Papadimitriou. The complexity of coloring circular arcs and chords. *SIAM Journal on Algebraic Discrete Methods*, 1(2):216–227, 1980. doi:10.1137/0601025.

17 Martin Charles Golumbic. Chapter 8 – interval graphs. In Martin Charles Golumbic, editor, *Algorithmic Graph Theory and Perfect Graphs*, volume 57 of *Annals of Discrete Mathematics*, pages 171–202. Elsevier, 2004. doi:10.1016/S0167-5060(04)80056-6.

18 Magnús M. Halldórsson and Mario Szegedy. Lower bounds for on-line graph coloring, 1994.

19 Makoto Imase and Bernard M. Waxman. Dynamic Steiner tree problem. *SIAM J. Discrete Math.*, 4(3):369–384, 1991. doi:10.1137/0404033.

20 Henry A. Kierstead and William T. Trotter, Jr. An extremal problem in recursive combinatorics. *Congr. Numer.*, 33:143–153, 1981.

21 Jakub Łącki, Jakub Oćwieja, Marcin Pilipczuk, Piotr Sankowski, and Anna Zych. The power of dynamic distance oracles: efficient dynamic algorithms for the Steiner tree. In *STOC'15 – Proceedings of the 2015 ACM Symposium on Theory of Computing*, pages 11–20. ACM, New York, 2015.

22 Dániel Marx. Precoloring extension on unit interval graphs. *Discrete Applied Mathematics*, 154(6):995–1002, 2006. doi:10.1016/j.dam.2005.10.008.

23 Nicole Megow, Martin Skutella, José Verschae, and Andreas Wiese. The power of recourse for online MST and TSP. In *Automata, languages, and programming. Part I*, volume 7391 of *Lecture Notes in Comput. Sci.*, pages 689–700. Springer, Heidelberg, 2012. doi:10.1007/978-3-642-31594-7_58.

24 Stephan Olariu. An optimal greedy heuristic to color interval graphs. *Inform. Process. Lett.*, 37(1):21–25, 1991. doi:10.1016/0020-0190(91)90245-D.

25 James B. Orlin, Maurizio A. Bonuccelli, and Daniel P. Bovet. An $O(n^2)$ algorithm for coloring proper circular arc graphs. *SIAM Journal on Algebraic Discrete Methods*, 2(2):88–93, 1981. doi:10.1137/0602012.

26 H. Maehara P. Frankl. Open-interval graphs versus closed-interval graphs. *Discret. Math.*, 63:97–100, 1987.

27 Rasmus V. Rasmussen and Michael A. Trick. Round robin scheduling – a survey. *European Journal of Operational Research*, 188(3):617–636, 2008. doi:10.1016/j.ejor.2007.05.046.

28 Wei-Kuan Shih and Wen-Lian Hsu. An o(n1.5) algorithm to color proper circular arcs. *Discrete Applied Mathematics*, 25(3):321–323, 1989. doi:10.1016/0166-218X(89)90011-5.

29 Shay Solomon and Nicole Wein. Improved dynamic graph coloring. In *26th European Symposium on Algorithms*, volume 112 of *LIPIcs. Leibniz Int. Proc. Inform.*, pages Art. No. 72, 16. Schloss Dagstuhl. Leibniz-Zent. Inform., Wadern, 2018.

30 Alan Tucker. Coloring a family of circular arcs. *SIAM Journal on Applied Mathematics*, 29(3):493–502, 1975. doi:10.1137/0129040.

31 Mario Valencia-Pabon. Revisiting Tucker's algorithm to color circular arc graphs. *SIAM Journal on Computing*, 32(4):1067–1072, 2003. doi:10.1137/S0097539700382157.