Distributed Recoloring of Interval and Chordal Graphs

Nicolas Bousquet, Laurent Feuilloley, Marc Heinrich, Mikaël Rabie

Distributed Recoloring of Interval and Chordal Graphs

Nicolas Bousquet
Univ. Lyon, Université Lyon 1, LIRIS UMR CNRS 5205, F-69621, Lyon, France

Laurent Feuilloley
Univ. Lyon, Université Lyon 1, LIRIS UMR CNRS 5205, F-69621, Lyon, France

Marc Heinrich
University of Leeds, UK

Mikaël Rabie
Université de Paris, CNRS, IRIF, F-75013, Paris, France

Abstract

One of the fundamental and most-studied algorithmic problems in distributed computing on networks is graph coloring, both in bounded-degree and in general graphs. Recently, the study of this problem has been extended in two directions. First, the problem of recoloring, that is computing an efficient transformation between two given colorings (instead of computing a new coloring), has been considered, both to model radio network updates, and as a useful subroutine for coloring. Second, as it appears that general graphs and bounded-degree graphs do not model real networks very well (with, respectively, pathological worst-case topologies and too strong assumptions), coloring has been studied in more specific graph classes. In this paper, we study the intersection of these two directions: distributed recoloring in two relevant graph classes, interval and chordal graphs.

More formally, the question of recoloring a graph is as follows: we are given a network, an input coloring α and a target coloring β, and we want to find a schedule of colorings to reach β starting from α. In a distributed setting, the schedule needs to be found within the LOCAL model, where nodes communicate with their direct neighbors synchronously. The question we want to answer is: how many rounds of communication are needed to produce a schedule, and what is the length of this schedule?

In the case of interval and chordal graphs, we prove that, if we have less than 2ω colors, ω being the size of the largest clique, extra colors will be needed in the intermediate colorings. For interval graphs, we produce a schedule after $O(\text{poly}(\Delta)\log^* n)$ rounds of communication, and for chordal graphs, we need $O(\omega^2\Delta^2\log n)$ rounds to get one.

Our techniques also improve classic coloring algorithms. Namely, we get $\omega+1$-colorings of interval graphs in $O(\omega\log^* n)$ rounds and of chordal graphs in $O(\omega\log n)$ rounds, which improves on previous known algorithms that use $\omega+2$ colors for the same running times.

2012 ACM Subject Classification Theory of computation → Distributed algorithms

Keywords and phrases Distributed coloring, distributed recoloring, interval graphs, chordal graphs, intersection graphs

Digital Object Identifier 10.4230/LIPIcs.OPODIS.2021.19

Related Version *Full Version*: `https://arxiv.org/abs/2109.06021` [11]

Funding This work was supported by ANR project GrR (ANR-18-CE40-0032).

Acknowledgements We thank the reviewers for their useful comments, and Marthe Bonamy for starting this project with us. The second author thanks Fabian Kuhn and Václav Rozhoň for their kind and expert answers to his questions on network decomposition.

1 Introduction

Finding a proper coloring of the network is one of the most studied problems in the LOCAL model of distributed computing. It is a typical symmetry-breaking problem, and it can be used as a building block to solve many other problems [4]. It also has direct applications, and a popular one since [31] is the assignment of frequencies (or time slots) in wireless networks. Consider a network of stations using radio communication. A simple model consists in considering that either two stations are close enough to communicate, but then they should use different emission frequencies to avoid collisions, or they are far apart, cannot communicate, and can safely use the same frequency. An assignment of frequencies that avoids conflict is equivalent to a proper coloring. (Note that this is the simplest model, one could also consider that communication and collision happen at different distances.)

Simply computing a coloring is not enough in some contexts. Indeed, suppose that we have an algorithm for the classic coloring task: starting from a coloring with a high number of colors (for example, unique identifiers), we get a coloring with fewer colors. Now, how do we update the frequencies of the network? If we do it simultaneously without stopping all communications, there might be conflicts during the transition period, between the nodes still using the old frequencies and the ones using the new frequencies. Thus, we need to stop all communications during the transition period, which is inconvenient if the network is performing other tasks in parallel. Therefore, instead of just a new coloring, we would like to have a series of colorings, such that at any step the color change is safe, that is old and new colors do not conflict. In this paper, we aim at finding such schedules in a distributed way. More precisely, we tackle the following more general recoloring question: how to go from an input coloring to a target coloring, such that at each step, the coloring is proper and the vertices whose colors are changing form an independent set of the network. To do so, it will sometimes be necessary to assume that we are allowed to use a few additional colors, that are not present in the input and target colorings.

For this problem, three questions naturally arise: Do we need extra colors to be able to produce a schedule, and if yes, how many? Can we find a new schedule locally? And how long the recoloring schedule needs to be? We study the problem from the point of view of the LOCAL model, thus we want our algorithms to use a sublinear number of communication rounds. For this setting, it is known that in general graphs we sometimes need to use many additional colors (*e.g.* $k-1$ extra colors to go from a k-coloring to another). This is bad news, as using extra colors can be considered as costly (*e.g.* because we need to reserve frequencies to make the transitions). Fortunately, radio networks have topologies that are more constrained than general graphs, and we want to exploit these properties to design schedules that use less extra colors.

In a first approximation, radio networks can be modeled as intersection graphs of (unit) disks in the plane (see Figure 1). In a unit disk graphs, we associate to each node a location on the plane, and two nodes are in conflict (i.e., linked by an edge) if their (unit) disks intersect. Unit disks can always be colored with 3ω colors [30], where ω is the size of the maximum clique of the graph. In contrast, there exists triangle-free graphs with arbitrarily large chromatic number. However, deciding if there exists a k-coloring remains NP-complete in the centralized setting, even in unit disks graphs. In general, we do not have a good understanding of the coloring of unit-disk graphs, and we know very little about recoloring, even in the centralized setting.

One can then wonder what happens in other simpler geometric graph classes. For instance, if we assume that all the centers are on the same line in the plane, the obtained graphs are called *interval graphs*. An interval graph is an intersection graph of intervals of the real line

(see Figure 1). In other words, an interval of the real line is associated to every node of the graph and there is an edge between two nodes if their corresponding intervals intersect. Interval graphs can be colored with ω colors in polynomial time in the centralized setting.

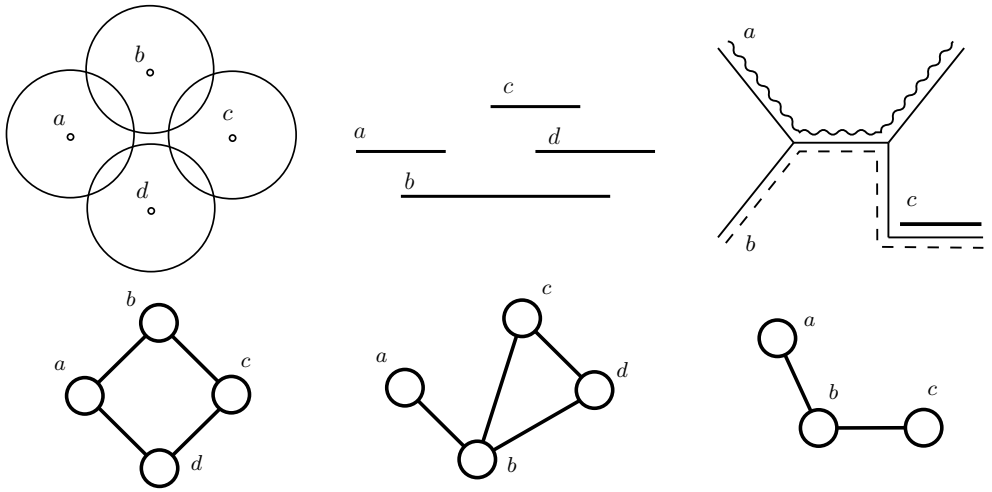

Figure 1 Illustrations of the different intersection graph classes mentioned. On each column, the geometric intersection model is on top, and the corresponding graph on the bottom. The left-most column is for unit-disk graph, the middle one for interval graphs, and the right-most for chordal graphs. For the chordal intersection model, the base tree uses plain edges, a is the wavy subtree, b is the dashed one, and c the bold one. (This last intersection model is simplistic, to allow a readable drawing. In general the subtrees are not paths, and for this specific graph, a simpler geometric representation exists.)

A natural generalization of interval graphs are chordal graphs. Chordal graphs are one of the most-well studied classes in graph theory, with deep structures, fast algorithms, and many important applications [21, 6, 33]. They are intersection graphs of subtrees of a tree. In other words, given a tree T, every vertex of a chordal graph G is associated to a subtree of T and two nodes of G are adjacent if their corresponding subtrees intersect (see Figure 1).

Our main results are recoloring algorithms for interval and chordal graphs. Our techniques also yield coloring algorithms that improve on the state of the art.

1.1 Our results

Our main results are recoloring algorithms for interval and chordal graphs. Let us describe a bit more formally what a distributed recoloring algorithm is. We use the definition introduced in [9]. A *valid recoloring schedule* from a coloring α to a coloring β consists in a sequence of colorings that starts with α and ends in β such that, at every step, the coloring is proper and the vertices whose colors are modified at a given step (called the *recolored vertices*) form an independent set of the graph. A distributed recoloring algorithm is an algorithm such that every node computes its own schedule. In this paper, we compute the schedule in the LOCAL model. Each node can check the validity of the schedule by comparing its own with its neighbors: they check that at each step, the coloring is locally proper, and that if it changes its own color during a step, none of its neighbors does the same.

In this paper, we will study how many *rounds* are used in the LOCAL model to produce a schedule of some *length*. We first focus on interval graphs and then extend our result to chordal graphs. We first prove the following:

▶ **Theorem 1.** *Let G be an interval graph and α, β be two proper k-colorings of G. It is possible to find a schedule to transform α into β in the LOCAL model in $O(\text{poly}(\Delta)\log^* n)$ rounds using at most:*

- *c additional colors, with $c = \omega - k + 4$, if $k \leqslant \omega + 2$, with a schedule of length $\text{poly}(\Delta)$,*
- *1 additional color if $k \geqslant \omega + 3$, with a schedule of length $\text{poly}(\Delta)$,*
- *no additional color if $k \geqslant 2\omega$ with a schedule of exponential-in-Δ length.*
- *no additional color if $k \geqslant 4\omega$ with a schedule of length $O(\omega\Delta)$.*

We also prove in Lemma 14 that the complexity of second item of Theorem 1 is optimal, in the sense that with less than 2ω colors, the number of rounds (as well as a schedule) must be linear in the size of the graph. Basically, when the number of colors is smaller than 2ω, if we do not have additional colors, we might need to recolor vertices from the border of a graph in order to be able to recolor vertices in the middle. On the contrary, when the number is at least 2ω, we can save a color using only local modifications, and then proceed as if we had one additional unused color. Unfortunately, in order to free this additional color, we use a schedule of size exponential in Δ.[1] On the positive side, this color saving step (and actually the whole recoloring) can be performed with a short schedule when $k \geqslant 4\omega$.

For the first item, as argued above, at least one additional color is needed. The colors that we use in addition to that one come from a result of [10] that we use (almost) as a black-box (see the next section). If the number of colors could be decreased in the context of [10], then it could also be decreased in our work. We left as an open problem the question of deciding if the numbers of additional colors can be reduced to 1.

One can naturally wonder what are the dependencies in Δ and ω on our complexity and schedule lengths, as we did not give them explicitly. We are using as a black-box the recoloring result on chordal graphs of [10], which gives a centralized recoloring algorithm in $O(\omega^4 \cdot \Delta \cdot n)$ steps, where only one vertex can be recolored at each step. The schedule we produce contains as subsequence a constant number of such black-box schedules, corresponding to subgraphs containing about $O(\omega^4 \cdot \Delta^2)$ vertices. In order to keep the proofs as simple as possible, we did not try to optimize this polynomial function. In particular, the recoloring procedure of [10] can probably be adapted in the LOCAL model with a parallel schedule shorter than $O(\omega^4 \cdot \Delta \cdot n)$.

Since for interval graphs, the gap between ω and Δ can be arbitrarily large, it would be interesting to determine if the recoloring schedule (as well as the number of rounds) can be reduced to a function of ω only.

Our second main result consists in extending this recoloring result to chordal graphs. Namely, we prove:

▶ **Theorem 2.** *Let G be a chordal graph and α, β be two proper k-colorings of G. It is possible to find a schedule of length $n^{O(\log \Delta)}$ to transform α into β in $O(\omega^2\Delta^2 \log n)$ rounds in the LOCAL model using at most:*

- *c additional colors, with $c = \omega - k + 4$, if $k \leqslant \omega + 2$,*
- *1 additional color if $k \geqslant \omega + 3$.*

Note that while the recoloring schedule of Theorem 1 is polynomial, the one obtained in Theorem 2 is superpolynomial in Δ (but the number of rounds remains polynomial). We left as an open problem the existence of polynomial schedule. In terms of number of rounds,

[1] It is likely that the $O(n^{d+1})$-recoloring algorithm in the centralized setting for d-degenerate graphs can be adapted in order to provide a polynomial schedule instead, but we did not do it to keep the proof as short as possible.

we also pay a logarithmic factor in comparison to the $O(\log^* n)$ rounds used for interval graphs. It would be interesting to know if there exists a $(\omega+x)$-coloring algorithm for chordal graphs for a constant x, that only needs $O(\log^* n)$ rounds. We will explain in the subsection that follows, that these differences in schedule length and running time are inherent to our approach, and going beyond those would require new techniques.

While designing tools for recoloring, we also produce improved algorithms for the coloring problem. In a nutshell, it was known how to get $(\omega+2)$-colorings for interval and chordal graphs, and we improve this to $(\omega+1)$ colors. More precisely, it was proved in [23] that an $(\omega+2)$-coloring of interval graphs can be obtained in the LOCAL model in $O(\omega \log^* n)$ rounds.[2] Our first result consists in improving the result of Halldorsson and Konrad [23] by proving:

▶ **Theorem 3.** *Interval graphs can be colored with $(\omega+1)$-colors in $O(\omega \log^* n)$ rounds in the LOCAL model.*

Note that $(\omega+1)$ is the best we can hope for in sublinear time in paths (which are interval graphs with $\omega = 2$), since paths cannot be colored with 2 colors in less than $\Omega(n)$ rounds [26, 4]. This can actually be generalized for any fixed ω, by considering the $(\omega-1)$-th power of a path. The *k-th power of a path* is a graph with vertex set $v_1, \dots, v_n$ where two vertices v_i, v_j are adjacent if and only if $|i-j| \leqslant k$. Hence, each maximal clique corresponds to ω consecutive nodes. An ω-coloring of such a graph would require the algorithm to put a same color every ω nodes in the path, which again requires $\Omega(n)$ communication rounds.

For chordal graphs, Konrad and Zamaraev [24] proved that an $(\omega+2)$-coloring can be found in $O(\omega \log n)$ steps. Again, we can reduce the number of colors to $\omega+1$.

▶ **Theorem 4.** *Chordal graphs can be colored with $(\omega+1)$-colors in $O(\omega \log n)$ rounds in the LOCAL model.*

Again, the dependency in n is optimal for $\omega = 2$ because (unoriented) trees cannot be 3-colored in $o(\log n)$ rounds [26, 4]. We leave as an open question if $\Omega(\log n)$ communication rounds are required to produce an $(\omega+1)$-coloring for $\omega > 2$.

1.2 Our techniques

Our algorithms are using graph decompositions. Basically, we first split the graph into components of controlled diameter, and then work on the different components, taking care of not creating conflicts at the borders.

To get more into the details, let us start with coloring. For interval graphs, we use a variation of a decomposition from [23]. It consists in proving that we can find some subsets of vertices that cut the interval graphs into parts whose diameter is large enough, but not too large, in $O(\log^* n)$ rounds (with a multiplicative factor depending on the diameter of each part). See Figure 2. Our contribution is in the second part of the algorithm. We show that we can start from an arbitrary coloring of the cuts, and extend this partial coloring into a tight $(\omega+1)$-coloring of G, whereas previous algorithms needed some extra slack in the number of colors.

For chordal graph, [24] provides a recursive decomposition into interval graphs, that we adapt to our setting. The idea is to use a rake-and-compress strategy (in the spirit of [28]) to decompose the graph in $O(\log n)$ layers, each layer being a union of interval graphs. A

[2] Actually, in [23] the result is stated as a $(1+\epsilon)$-approximation of the optimal ω-coloring, with the condition that $\epsilon \geqslant 2/\omega$, which is equivalent to an $(\omega+2)$-coloring.

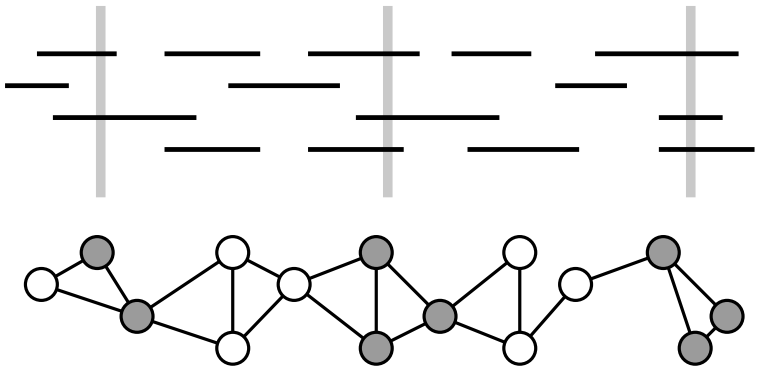

Figure 2 Illustration of the interval decomposition of [23]. On top the interval representation, and on the bottom, the graph. The decomposition consists in choosing cliques (the gray nodes) whose removal decomposes the graphs into subgraphs of controlled diameter (the white nodes). Note that the graph can then be seen as a path alternating between gray cliques and white subgraphs. What we do is slightly different, as our cuts are not cliques, but the intuition is the same.

typical phase of rake-and-compress on a tree consists in removing the leaves (the rake part), and removing or contracting the long paths (the compress part). After $O(\log n)$ such phases, the tree is empty. In our algorithm, the tree structure is the underlying tree of the chordal graph, like in Figure 1.[3] We consider here as leaves the pending interval subgraphs. These are subgraphs made by taking the intervals that are on the branches of the tree leading to a leaf (see Fig. 3). On the other hand, long paths correspond to long separating interval subgraphs, whose removal disconnects the graph, and that have large enough diameter.

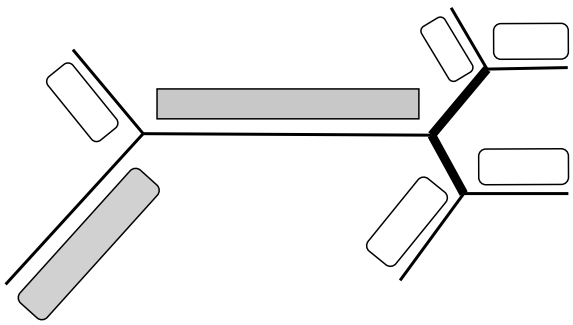

Figure 3 Illustration of one step of the decomposition of chordal graphs into interval graphs (of controlled diameter). The tree of plain edges is the underlying tree. The rectangles with rounded corners correspond to the pending interval graphs. The rectangle with spiky corners correspond to a separating interval graph. The gray rectangles are the ones that are too long and will be further decomposed. The bold edges correspond to part of the graphs that are not pending, nor long and separating. These part are the ones that are kept for the next phases, all the other vertices are removed.

Let V_i be the set of nodes of the interval graph removed at phase i. The coloring algorithm then consists in coloring recursively $V_k, \ldots, V_1$, in this order. That is, we start by coloring the vertices that have been removed last. Assume that we have a coloring of $V_{i+1}, \ldots, V_k$.

[3] Actually, there are two such trees, the one that correspond to the geometric representation, that we use in this introduction, and the clique tree, which has a more graph-theoretic definition, and that we use in the proofs. The two are essentially equivalent.

The key point is that when we consider a new interval subgraph, by construction, only its borders can be already colored, and therefore we are back to the situation we had for interval graphs. We then get the same number of colors. The logarithmic-in-n complexity comes from the $O(\log n)$ layers of the rake-and-compress, that we must process sequentially.

Now, let us describe our recoloring techniques. For interval graphs, Theorem 1 is obtained using two tools coming from the centralized setting. First, we use Kempe chains that have been used in several graph coloring proofs in the last decades. Given a graph G and a coloring c of G, an (a, b)-*component* is a connected component of G restricted to the vertices colored a and b. A *Kempe component* is an (a, b)-component for some pair of colors a, b. One can remark that, given a proper coloring and an (a, b)-component, permuting the colors a and b still leaves a proper coloring. In recoloring, this permutation can be performed by using one extra color as buffer. In a distributed setting, using directly those transformations can be perilous, as components can be as large as the diameter of the graph. We adapt Kempe components for distributed algorithms by using an additional color to cut these Kempe components and then permute locally the colors on these components[4]. This will allow us to color independently some parts of the graph with the target coloring.

Our second main tool is a recoloring technique introduced in [10] for recoloring interval graphs in the centralized setting. We show how to adapt their technique to the distributed setting. The result of [10] essentially ensures that if a large enough (that is, of size $\text{poly}(\Delta)$) number of consecutive vertices X are colored in a desirable way, then we can recolor all the vertices around X with the target coloring by simply recoloring vertices locally. The idea of the proof consists in sliding little by little the set of vertices X colored with the desirable coloring from left to right in such a way that when a vertex leaves the set, it has its target color.

For chordal graphs (Theorem 2), we use Theorem 1 as a black-box, as well as an adaptation of the distributed partition of chordal graphs into interval pieces from [24]. We then prove that if we have a recoloring schedule of $G[\cup_{j\geqslant i+1} V_j]$ then we can adapt it into a recoloring schedule for $G[\cup_{j\geqslant i} V_j]$ using Theorem 1 and classical recoloring tools.

Notice that the schedule for chordal graphs is large in comparison with the ones for interval graphs (order of $n^{O(\Delta)}$ versus $\text{poly}(\Delta \cdot \log^* n)$). This comes from the fact that in the extension of the recoloring schedule of $G[\cup_{j\geqslant i+1} V_j]$ to $G[\cup_{j\geqslant i} V_j]$, we need to leave a polynomial amount of recoloring steps to recolor vertices of V_i between consecutive recolorings of vertices of $G[\cup_{j\geqslant i+1} V_j]$.

The schedule having a $\log n$ factor instead of a $\log^* n$ factor originates once again from the decomposition of [24] that uses a logarithmic number of layers.

1.3 Organization of the paper

We start with the related work in Section 2, and some preliminaries in Section 3. Then in Section 4, we explain how to decompose interval graphs and how to modify colorings in such graphs. As a corollary, we get our coloring result for interval graphs. Section 5 is devoted to the recoloring of interval graphs, based on the decomposition. Finally, Section 6 tackles chordal graphs, describing our decomposition, coloring and recoloring results. Many proofs, as well as full subsections of Section 5, are deferred to the full version [11].

[4] Kempe chains have already been used in the distributed setting, see e.g. [29].

2 Related work

Distributed coloring

Distributed coloring in bounded-degree and general graphs has been extensively studied. We refer to the monograph [4] for a general overview of the domain. In the LOCAL model, the classic coloring problem is $(\Delta+1)$-coloring, where Δ is the maximum degree of the graph. In this paper, we are interested in coloring and recoloring with a nearly optimal number of colors (which can be much lower than Δ). There are ways to cope with such small color palette while preserving some locality: studying (multiplicative) approximation of the optimal coloring [2, 5] and/or restricting to special graph classes, for example planar graphs [20, 1, 15] or bounded arboricity graphs [3, 19].

The line of work that is the closest to ours is about coloring interval and chordal graphs. It started with [22] who proved an 8-approximation of the optimal coloring, in time $O(\log^* n)$ in interval graphs. This result was then improved to $(1+\epsilon)$-approximation in time $O\left(\frac{1}{\epsilon}\log^* n\right)$ in [23]. The ϵ of this theorem can be as small as $2/\omega$, which means that [23] obtains an $(\omega+2)$-coloring in time $O(\omega \log^* n)$. Coloring in chordal graphs was considered in [24]. The authors prove a $(1+\epsilon)$-approximation in time $O\left(\frac{1}{\epsilon}\log n\right)$. Again, the bounds on ϵ allow to derive an $(\omega+2)$-coloring, and here the running time is $O(\omega \log n)$ rounds.

Distributed Reconfiguration

The introduction of recoloring, *i.e.* reconfiguration of colorings, in the distributed setting is due to [9]. They, for instance, provide algorithms to recolor trees and subcubic graphs with one extra color, and an impossibility result for 3-colored toroidal grids with only one extra color. In particular, they find a constant size schedule with $O(\log n)$ communication rounds on trees if an extra color is allowed. Trees are chordal graphs with $\omega = 2$, hence our results relate directly to that, and we use a similar *Rake-and-Compress* approach. Then [13] considered the distributed reconfiguration of maximal independent sets (MIS). Before these papers, some results of [29] can be seen as recoloring, as they describe a way to find a Δ-coloring from a given $(\Delta+1)$-coloring by using "augmenting paths" that actually are Kempe changes.

Centralized graph recoloring

In the centralized setting, graph recoloring received a considerable attention in the last decades. In this overview, we will focus on single-vertex reconfiguration, that is, the model where exactly one vertex is recolored at each step. In that model, it is known that every k-coloring can be transformed into any other as long as k is at least the degeneracy d of G plus two [17, 14]. However, the number of recoloring steps is *a priori* exponential in n. Cereceda conjectured in [14] that, when $k \geqslant d+2$, there exists a quadratic transformation between any pair of k-colorings of any d-degenerate graph. Bousquet and Heinrich [12] proved that there always exist $O(n^{d+1})$ transformation between any pair of k-colorings when $k \geqslant d+2$.

Since chordal graphs are perfect graphs, it is well known that the degeneracy is related to the cliques number by: $d = \omega - 1$. For interval and chordal graphs, Bonamy et al. [8] proved that there exists a quadratic transformation between any pair of vertices when $k \geqslant \omega+1$ and they provide a quadratic lower bound when $k = \omega+1$. This existence of a quadratic transformation has been extended to bounded treewidth graphs [7]. Recently Bartier and Bousquet proved in [10] that, when $k \geqslant \omega+3$, there always exists a linear transformation

between any two colorings of a chordal graph. Moreover, in contrast to most of the centralized recoloring algorithms, this algorithm is "local" and will be used as one of the blocks of our proof.

Graph decompositions

The first phase of our algorithms consists in decomposing the graph into components of small diameter with some additional properties. These decompositions are close to what is known as network decompositions, which are partitions of the nodes of the graph into few classes, such that every class has all its connected components of small diameter (see [18] for an overview of this topic, and in particular the recent advances). For geometric graphs, such as interval, unit-disks graphs, it is known that a network decomposition with a constant number of classes and constant diameter can be obtained in $O(\log^* n)$ rounds [25, 32], which implies that all the classic problems such as $(\Delta+1)$-coloring or maximal matching can be solved in $O(\log^* n)$ rounds. For trees, [9] establishes a network decomposition with constant size components in time $O(\log n)$, which is enough for recoloring. For our purpose, which is to (re)color interval and chordal graphs with few colors, standard network decompositions are not powerful enough, and we need to build more constrained decompositions. Our decompositions are inspired by the ones of [23] and [24].

3 Basic properties of interval and chordal graphs

Interval graphs are intersection graphs of intervals of the real line. *Proper interval graphs* are the intersection graphs of a set of intervals of size one. Equivalently, they correspond to interval graphs where no interval is included in the other. It is easy to check that, starting from an interval graph, and keeping only the intervals that are maximal for inclusion, we get a proper interval graph.

An interval graph can also be characterized as an intersection graph of subpaths of a path, and we can extend this definition to define chordal graphs. A graph is *chordal* if it is an intersection graph of subtrees of a tree. Equivalently, it is the class of graphs for which all the cycles of length at least 4 has a chord.

Before we list some properties of interval, proper intervals and chordal graphs, let us define the notion of degeneracy.

▶ **Definition 5.** *A graph G is* d-degenerate *if there exists an ordering $v_1, \dots, v_n$ of V such that for every $i \leqslant n-1$, the number of neighbors of v_i in $\{v_{i+1}, \dots, v_n\}$ is at most d.*

Let us remind a few facts about interval graphs.

▶ **Observation 6.** *The following holds.*

1. *Any chordal graph can be colored with ω colors and is $(\omega-1)$-degenerate [27].*
2. *The max degree Δ can be as large as n, even if ω is small (consider $n-1$ disjoint small intervals contained in a big one).*
3. *In an interval graph, the set of intervals that are minimal for inclusion corresponds to a proper interval. The same holds for "maximal for inclusion" [23].*
4. *In any proper interval graph, $\Delta \leqslant 2\omega - 2$ (since every neighboring interval should cross one of the extremities of the vertex interval.)*

Clique trees, clique paths and borders

An essential tool to study chordal graphs is the notion of clique tree.

▶ **Definition 7** (See *e.g.* [6]). *Let G be a chordal graph. A* clique tree *of G is a tree T together with a function that associates to every vertex a connected subtree of T. Two vertices x and y are connected in G if and only if the subtrees of x and y intersect. For each node $u \in T$, the subset of nodes in G whose subtree contains u is called the* bag *of u. Note that each bag of a node of T forms a clique in G, so each bag contains at most ω vertices (and there always exists a clique tree on at most n nodes).*

For interval graphs, the tree T is a path, and it is called the clique path *of G (see Figure 4 for an illustration).*

We introduce two more notions. Let G be an interval graph. A set of vertices X is said to be *consecutive* if it consists in the union of the vertices contained in consecutive cliques of the clique path of G. The *border* of X is the subset of X connected to at least one vertex which is not in X. In other words, it is the set of vertices of the first and last clique of the clique path containing X that have a neighbor not in X. One can easily notice that all the vertices of the border of X can be partitioned into two cliques (the ones that belong to the first and the last clique of the set of bags).

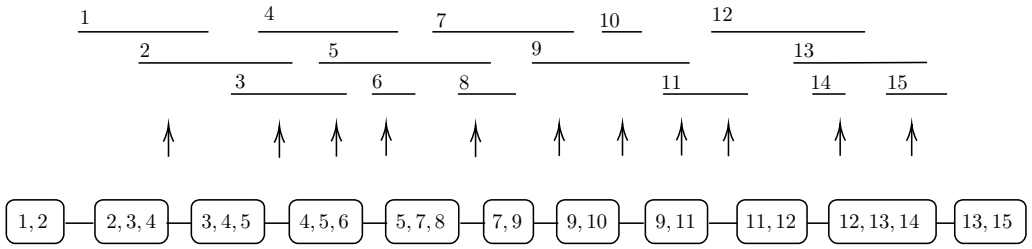

Figure 4 The first picture represents a set of intervals. The arrows correspond to points of the axis where there is a maximal clique. The second picture describes the clique path of the graph, whose nodes are the maximal cliques.

We will need the following remarks about interval graphs:

▶ Remark 8. Let G be an interval graph given with an interval representation of G. Let x be a vertex of G. Then:

- $N(x)$ contain all the vertices v whose intervals intersect the interval of x.
- The removal of $N(x) \cup \{x\}$ separates the vertices with interval at the left of x with the vertices with interval at the right of x.[5] A direct consequence is that, for each node x that is not contained in a bag of a node of one extremity of the clique path, its box separates the graph.
- For every $r \geqslant 2$, consider the set $Y = \cup_{i \leqslant r} N^i(x)$. The subset of vertices of Y that are incident to any vertex of $V \setminus Y$ is composed of at most two cliques A, B. Moreover, there is a clique path of $G[Y]$ where A is the first clique and B is the last clique.

[5] Note that there might be more components since, for instance, the "left graph" might not be connected.

4 Decomposing and coloring interval graphs

In the sequential setting, it is easy to compute an interval representation of the graph, and such a representation is often useful in the design of algorithms. For example, one can compute a maximum independent set in a greedy manner, scanning the interval by the increasing right-most endpoint. In our setting, the nodes do not have access to such a representation, and it is not possible to compute one locally. We will build a weaker local version of the interval representation to facilitate the design of coloring and recoloring algorithms. This idea originates from [23, 22].

We now introduce the notion of *boxes*. Remember that an (a, b)-ruling-set is a set of nodes S, such that for any $u, v \in S$ u and v are at distance at least a, and any node is at distance at most b from a node of S.

▶ **Definition 9.** *Given a $(4, 5)$-ruling-set S, for every node v of S, we define its* box *as its closed neighborhood in the graph.*

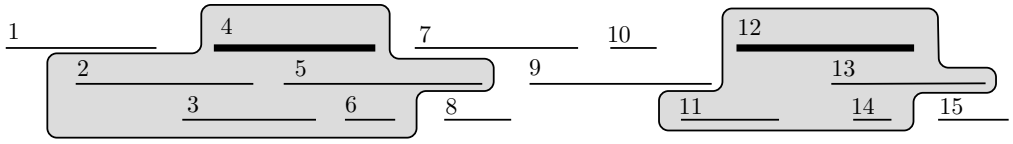

Figure 5 An example of a box decomposition. The boxes are the gray areas. The other areas are interboxes. The nodes of the ruling sets are the bold intervals. The intervals 6, 8, 10 and 14, are included into other interval thus they are not considered for the computation of the ruling set.

▶ **Proposition 10.** *For a $(4, 5)$-ruling-set S in an interval graph G, the following holds:*

1. *The boxes of the nodes in S are disjoint, and two nodes from two different boxes cannot be adjacent.*
2. *The removal of all the boxes leaves a set of connected components, that we call* interboxes*. We have a path alternating boxes and interboxes (by defining adjacency between boxes and interboxes by the existence of an edge linking the two). We call the* virtual path *the path on vertex set S whose adjacency is given by the sequence of boxes defined above.*
3. *The interboxes have diameter at most 11.*

The proof, as well as all the other missing proof of this section, can be found in the full version [11].

Now from a distributed point of view, we say that the nodes compute a box decomposition if they compute a $(4, 5)$-ruling-set S, and for every node, either it is in a box, and then it knows the two adjacent boxes, and the structure of the graph between these boxes, or it is not in a box, and knows between which two boxes it is, and the structure of the graph between these boxes.

Let S be a $(4, 5)$-ruling set of G. If we replace every box a by a single vertex and every interbox by an edge, the resulting graph is a path called the *ruling path*. Two vertices of S are said to be consecutive if they are adjacent in the ruling path and at distance at most r if they are at distance at most r in the ruling path.

▶ **Lemma 11.** *A box decomposition can be found in $O(\log^* n)$ rounds in the LOCAL model.*

Let M be a box decomposition. As in the proof of Lemma 11, there is a natural virtual path between these boxes. Let B, B' be two boxes of a box decomposition. We say that the boxes B, B' are at distance r, if they are at distance r in the virtual path of the box decomposition. The subgraph between B and B' is the subgraph of G composed of the boxes B, B', all the boxes B and B' (in the virtual path) and all the interboxes between B and B'. Note that if the boxes B and B' are at distance r, then the subgraph between B and B' can be computed in $O(r)$ rounds in the LOCAL model.

We will now provide a technical lemma that will be helpful in several places of the paper.

▶ **Lemma 12.** *Let A, B be two boxes of two nodes at distance 3 and H be the subgraph between A and B. Let α be a proper k-coloring of H, and let x and y be two arbitrary colors. Let β be the k-coloring of H made by permuting x and y in α. There exists a $(k+1)$-coloring of H, that corresponds to α on A and β on B.*

▶ **Lemma 13.** *Consider the subgraph H induced by the nodes of two boxes A and B, separated by at least $3k$ other boxes, and all the nodes in between. Consider also two arbitrary proper k-coloring α and β of H. Then there exists a $(k+1)$-coloring of H, that corresponds to α on A and β on B.*

Given this tool, we easily get the following theorem.

▶ **Theorem 3.** *Interval graphs can be colored with $(\omega+1)$-colors in $O(\omega \log^* n)$ rounds in the LOCAL model.*

Proof. One first computes the box decomposition in time $O(\log^* n)$, then, given the paths of boxes, one can iterate an MIS algorithm for paths (*e.g.* Cole-Vishkin algorithm [16]) to compute a $(3\omega, 6\omega)$-ruling set S of boxes. This uses $O(\omega \log^* n)$ rounds. Then the nodes of the boxes of S computes an ω-coloring of their own box. Finally, we use the lemma above to fill the gaps, which also takes $O(\omega)$ rounds. ◀

5 Recoloring interval graphs

The goal of this section is to prove the following theorem.

▶ **Theorem 1.** *Let G be an interval graph and α, β be two proper k-colorings of G. It is possible to find a schedule to transform α into β in the LOCAL model in $O(\text{poly}(\Delta)\log^* n)$ rounds using at most:*

- *c additional colors, with $c = \omega - k + 4$, if $k \leqslant \omega + 2$, with a schedule of length $\text{poly}(\Delta)$,*
- *1 additional color if $k \geqslant \omega + 3$, with a schedule of length $\text{poly}(\Delta)$,*
- *no additional color if $k \geqslant 2\omega$ with a schedule of exponential-in-Δ length.*
- *no additional color if $k \geqslant 4\omega$ with a schedule of length $O(\omega\Delta)$.*

The result is tight in terms of number of colors for the two last items, because of the following lemma:

▶ **Lemma 14.** *In interval graphs of clique number ω, if no additional color is allowed, then finding a recoloring from a c-coloring to a c'-coloring with $c, c' < 2\omega$ requires $\Omega(n/\omega)$ rounds in the LOCAL model and a schedule of length $\Omega(n/\omega)$.*

Proof. We will consider the ω-th power of a path. We can build an interval representation of such a graph, by representing every vertex at position i by the interval $[i + 1/4, i + k + 3/4]$. If we consider a power of a path of clique number ω, and color its i-th vertex with color $i \bmod 2\omega - 1$, all the vertices but the first and the last ω ones are *frozen* (i.e. we cannot change

their color without changing the color of some vertex in its neighborhood before). Thus, a recoloring can only happen little by little starting from the extremities of the interval graph. Thus, a recoloring schedule has length $\Omega(n)$ and the nodes in the middle of the power path cannot stop before $\Omega(n)$ rounds, as their slot in the schedule will depend on the length of the power path. Moreover, a node in the middle needs to know its distance to an extremity, which takes $\Omega(n)$ rounds in the LOCAL model. ◀

5.1 Outline of the proof

We now give an outline of the proof of Theorem 1. The full version [11] contains detailed explanations and complete proofs for each step of this outline. Let α, β be two k-colorings.

Step 1. Compute a $(4,5)$-ruling set S in $O(\log^* n)$ rounds.

Step 2. In graph recoloring, instead of transforming α into β, a classical method consists in proving that both α and β can be recolored into a so-called canonical coloring γ with desirable properties (see e.g. [8, 10]). If transformations $\mathcal{S}_1, \mathcal{S}_2$ from respectively α and β to γ exist, then α can be transformed into β via the transformation $\mathcal{S}_1\mathcal{S}_2^{-1}$.
Theorem 3 ensures that an $(\omega+1)$-coloring of G can be found in $O(\omega \log^* n)$ rounds. Let us denote by γ such a coloring. The goal of the proof will consist in proving that we can find a transformation from α to γ.

Step 3. We show that if $k \geqslant 2\omega$, then we can reduce the number of colors without any additional color. We first compute an independent set S' of S at constant distance. The main idea of this step consists in proving that all the vertices in the interboxes between two consecutive vertices of S' but the vertices of the boxes of S' can be recolored with a color smaller than 2ω without recoloring the boxes of S' (which guarantees that we can perform this recoloring simultaneously everywhere in the graph). By repeating this operation twice, we then prove that all the vertices are recolored with a color smaller than 2ω.

When the number is at least 4ω, we prove a stronger result, since we directly provide a transformation from α to β in $O(\omega\Delta)$ rounds.

Step 4. At this point, after applying Step 3 if $k \geqslant 2\omega$, we can assume that we have one additional color. Indeed, in the two first cases of Theorem 1, we are allowed to use at least one extra color, and when $k \geqslant 2\omega$, we have freed at least one color at Step 3.

The goal of Step 4 consists in coloring some boxes with their colors in γ. To do that, we will perform some Kempe changes, cut at some large enough distance, using one additional color. By repeating this process several times, we will prove that a box plus all its neighbors at distance $\Omega(\text{poly}\,\Delta)$ can be colored with the target coloring.

Using this technique, we can prove that some portions of diameter $\Omega(\text{poly}(\Delta))$ of the interval graph which are at distance $f(\Delta)$ from each other are colored with the target coloring γ.

Step 5. By Step 4, we can assume that some portions of diameter $\Omega(\text{poly}(\Delta))$ of the interval graph which are at distance $f(\Delta)$ from each other are colored with the target coloring γ. We can now use as a black-box a result of Bartier and Bousquet [10] that ensures that we can recolor all the vertices between a consecutive set of boxes with the target coloring without recoloring the border of these sets as long as a sufficiently large number of consecutive vertices

of the initial coloring are colored "nicely". Since many consecutive vertices of the initial coloring are colored with the target coloring and such a coloring is "nice", we can use the result of [10] as a blackbox.

After Step 5, all the vertices have received their final color, which concludes the proof.

6 Decomposing, coloring and recoloring chordal graphs

We now investigate how to extend the results from the previous sections to chordal graphs. The algorithms for coloring and recoloring proceed in roughly two steps. The first step computes a partition of the chordal graph, obtained by iteratively removing interval subgraphs with controlled diameter. During the computation of this decomposition, we will assume that the nodes of the graph have access to a local view of a clique tree of the graph G. These local views can be computed in a distributed fashion with a constant number of rounds using the method from [24] (see Section 6.1 for more details).

In the second step, the coloring algorithm consists in adding back step by step the vertices which have been deleted in the previous phase. Informally speaking, for each connected component, either the vertices that have been removed are attached to a single clique, and then we know that all the vertices attached to that clique form a $(\omega - 1)$-degenerate graph, and we can then give them a color between 1 and ω greedily. Either those vertices are attached to two cliques at large enough distance, and we can then use the machinery introduced in Section 4 to color them with $\omega + 1$ colors in total. Since at each step, each set has bounded diameter, one leader can decide of the coloring for all the vertices of that set. For the recoloring algorithm, the idea is almost the same. The algorithm constructs iteratively a recoloring schedule by adding back step by step the vertices which were removed in the previous phase either using degeneracy or the tools of Section 5.

6.1 Interval decomposition of chordal graphs

▶ **Definition 15.** *Let G be a chordal graph, and H a subgraph of G. We say that H is a* pending interval graph *if H is an interval graph, and the border of H is a subset of the first clique in a clique path of H. We say that H is a* separator interval graph*, if it is an interval graph, and the vertices of the border all belong to either the first or the last clique of a clique path of H.*

An interval decomposition *of width D and depth ℓ of a chordal graph G is a partition $V_1, \ldots, V_\ell$ of the vertices of G such that, $G[V_i]$ is a disjoint union of interval graphs with diameter at most $3D$, and every connected component of $G[V_i]$ is either:*

- *a pending interval in $G[V_1, ..., V_i]$;*
- *or a separator interval graph with diameter at least D, in $G[V_1, ..., V_i]$.*

The result below is adapted from the proofs of [24].

▶ **Lemma 16.** *For every $D \geqslant 0$, there exists a distributed algorithm which computes an interval decomposition of width D and depth $O(\log n)$ in $O(D \log n)$ rounds in the LOCAL model.*

The proof can be found in the full version [11].

6.2 $(\omega + 1)$-coloring chordal graphs

In this section, we show how an interval decomposition can be used to compute an $(\omega + 1)$-coloring of chordal graphs in the LOCAL model. Namely, we prove the following theorem.

▶ **Theorem 4.** *Chordal graphs can be colored with $(\omega+1)$-colors in $O(\omega \log n)$ rounds in the LOCAL model.*

The proof of Theorem 4 follows from the decomposition of Lemma 16 and the lemma below.

▶ **Lemma 17.** *There is an algorithm that, given a graph G and an interval decomposition of G of depth ℓ and width at least $15(\omega+1)$, computes a $(\omega+1)$ coloring of G in $O(\omega\ell)$ rounds in the LOCAL model.*

The proof can be found in the full version [11].

6.3 Recoloring chordal graphs

In this section, we describe how to use an interval decomposition in order to compute a recoloring schedule. We prove the following theorem:

▶ **Theorem 2.** *Let G be a chordal graph and α, β be two proper k-colorings of G. It is possible to find a schedule of length $n^{O(\log \Delta)}$ to transform α into β in $O(\omega^2\Delta^2 \log n)$ rounds in the LOCAL model using at most:*

- *c additional colors, with $c = \omega - k + 4$, if $k \leqslant \omega + 2$,*
- *1 additional color if $k \geqslant \omega + 3$.*

Theorem 2 uses our decomposition (Lemma 16) and the result below.

▶ **Lemma 18.** *Let G be an interval graph and α, β be two proper $\omega+3$-colorings of G. We suppose we are given an interval decomposition of width $D = \Omega(\omega^2\Delta^2)$ and depth ℓ of G. It is possible to find a schedule of length at most $O(\text{poly}(\Delta)^\ell)$ to transform α into β in $O(D\ell)$ rounds in the LOCAL model.*

Proof. Let D chosen polynomial in ω and Δ (its value will be specified later). Note that an interval graph of diameter $D = \text{poly}(\omega, \Delta)$ has a number of nodes that is $O(\text{poly}\,\Delta)$, because this number is upper bounded by $D\Delta$ and $\omega \leqslant \Delta$. Also let k' be the total of number of colors used (including the additional colors). Let us denote by $V_1, \ldots, V_\ell$ the interval decomposition of G, and $G_i = G[V_1 \cup \ldots V_i]$. Let α and β the two colorings given as input, and let α_i to β_i be the restrictions of α and β to G_i. The algorithm proceeds by building successively a recoloring schedule λ_i for G_i from α_i to β_i, and progressively extending this recoloring schedule for the rest of the graph. In order to simplify the proof, we will require that the schedule produced by the algorithm has an additional property, namely that all the vertices recolored at a step j of the schedule, are recolored with the color $j \mod k'$. Note that we can easily produce a schedule satisfying this property, up to multiplying its length by a factor of k'. This property will be useful later in order to extend progressively the schedule.

Since G_1 is a disjoint union of interval graphs of diameter $O(D)$, we can compute a recoloring schedule in G_1 from α_1 to β_1 of length linear in the number of nodes by [10], which is in $O(\text{poly}\,\Delta)$ in $O(D)$ rounds.

Assume that we have computed a recoloring schedule in G_{i-1}. We now describe how to extend this schedule to G_i. Before each step of λ_{i-1}, we insert t steps during which only the vertices of V_i are recolored, where $t = \text{poly}(\Delta)$ is the length of the schedule we produced for interval graphs (multiplied by k' to ensure the fact that the recolored vertices in each step go to the same color). Finally, after the last step of λ_{i-1}, we insert again t steps where only the vertices of V_i are recolored such that they can reach their target coloring.

Let us consider the step j of λ_{i-1}, and let σ be the coloring of G_i just before this step. Let $c = j \mod k'$ be the color with which vertices recolored at this step are recolored with. We know that $\sigma|_{V_i}$ uses only $\omega + 3$ colors. Our goal is to recolor the vertices of V_i so that no vertex is colored c, and the recoloring step of λ_{i-1} can be safely applied.

Since G_{i-1} is obtained from G_i by removing pending interval graphs, and separating interval graphs of diameter at least D, there is a coloring σ' of G_i which agrees with σ on G_{i-1} and which uses at most $\omega + 1$ colors, all different from c for the vertices in V_i. Taking $D = 240\omega^2\Delta^2$, the same conditions as for interval graphs are satisfied, and there is a recoloring schedule from σ to σ' of length t. Similarly, after the last step of λ_{i-1}, if σ is the current coloring, then only $\omega + 1$ colors are used in V_i, and by Theorem 1, there is a recoloring schedule from σ to β_i in G_i of length t.

This new schedule can be computed for each component of $G[V_i]$ in a centralized way, and up to multiplying the length of the schedule by k', we can assume that at step j, vertices are recolored with the color $j \mod k'$. In a distributed setting, this requires $O(D)$ steps, since each component has diameter $O(D)$. Since we need $O(D)$ steps to extend the recoloring schedule from G_{i-1} to G_i, the algorithm uses at most $O(D\ell)$ rounds in total. Moreover, the schedule has length at most $t^\ell = (\operatorname{poly}\Delta)^\ell$. ◀

Now, for Theorem 2, from Lemma 16, we get ℓ in $O(\log n)$, and we set D in $O(\omega^2\Delta^2)$, hence we have an algorithm in $O(\omega^2\Delta^2 \log n)$ rounds that produces a schedule of length $(\operatorname{poly}\Delta)^{\log n}$ that is $n^{O(\log\Delta)}$.

References

1 Pierre Aboulker, Marthe Bonamy, Nicolas Bousquet, and Louis Esperet. Distributed coloring in sparse graphs with fewer colors. *Electron. J. Comb.*, 26(4):P4.20, 2019.

2 Leonid Barenboim. On the locality of some np-complete problems. In Artur Czumaj, Kurt Mehlhorn, Andrew M. Pitts, and Roger Wattenhofer, editors, *ICALP 2012*, volume 7392, pages 403–415, 2012. doi:10.1007/978-3-642-31585-5_37.

3 Leonid Barenboim and Michael Elkin. Sublogarithmic distributed MIS algorithm for sparse graphs using nash-williams decomposition. *Distributed Comput.*, 22(5-6):363–379, 2010. doi:10.1007/s00446-009-0088-2.

4 Leonid Barenboim and Michael Elkin. *Distributed Graph Coloring: Fundamentals and Recent Developments*. Synthesis Lectures on Distributed Computing Theory. Morgan & Claypool Publishers, 2013. doi:10.2200/S00520ED1V01Y201307DCT011.

5 Leonid Barenboim, Michael Elkin, and Cyril Gavoille. A fast network-decomposition algorithm and its applications to constant-time distributed computation. *Theor. Comput. Sci.*, 751:2–23, 2018. doi:10.1016/j.tcs.2016.07.005.

6 Jean RS Blair and Barry Peyton. An introduction to chordal graphs and clique trees. In *Graph theory and sparse matrix computation*, pages 1–29. Springer, 1993.

7 Marthe Bonamy and Nicolas Bousquet. Recoloring graphs via tree decompositions. *Eur. J. Comb.*, 69:200–213, 2018. doi:10.1016/j.ejc.2017.10.010.

8 Marthe Bonamy, Matthew Johnson, Ioannis Lignos, Viresh Patel, and Daniël Paulusma. Reconfiguration graphs for vertex colourings of chordal and chordal bipartite graphs. *J. Comb. Optim.*, 27(1):132–143, 2014. doi:10.1007/s10878-012-9490-y.

9 Marthe Bonamy, Paul Ouvrard, Mikaël Rabie, Jukka Suomela, and Jara Uitto. Distributed recoloring. In *DISC 2018*, volume 121, pages 12–1, 2018.

10 Nicolas Bousquet and Valentin Bartier. Linear transformations between colorings in chordal graphs. In *ESA 2019*, volume 144 of *LIPIcs*, pages 24:1–24:15, 2019. doi:10.4230/LIPIcs.ESA.2019.24.

11 Nicolas Bousquet, Laurent Feuilloley, Marc Heinrich, and Mikaël Rabie. Distributed recoloring of interval and chordal graphs. *CoRR*, abs/2109.06021, 2021. arXiv:2109.06021.

12 Nicolas Bousquet and Marc Heinrich. A polynomial version of cereceda's conjecture. *CoRR*, abs/1903.05619, 2019.

13 Keren Censor-Hillel and Mikaël Rabie. Distributed reconfiguration of maximal independent sets. *Journal of Computer and System Sciences*, 112:85–96, 2020.

14 L. Cereceda. *Mixing Graph Colourings*. PhD thesis, London School of Economics and Political Science, 2007.

15 Shiri Chechik and Doron Mukhtar. Optimal distributed coloring algorithms for planar graphs in the LOCAL model. In *SODA 2019*, pages 787–804. SIAM, 2019. doi:10.1137/1.9781611975482.49.

16 Richard Cole and Uzi Vishkin. Deterministic coin tossing with applications to optimal parallel list ranking. *Inf. Control.*, 70(1):32–53, 1986. doi:10.1016/S0019-9958(86)80023-7.

17 M. Dyer, A. D. Flaxman, A. M Frieze, and E. Vigoda. Randomly coloring sparse random graphs with fewer colors than the maximum degree. *Random Structures & Algorithms*, 29(4):450–465, 2006.

18 Mohsen Ghaffari. Network decomposition and distributed derandomization (invited paper). In *SIROCCO 2020*, volume 12156, pages 3–18, 2020. doi:10.1007/978-3-030-54921-3_1.

19 Mohsen Ghaffari and Christiana Lymouri. Simple and near-optimal distributed coloring for sparse graphs. In *DISC 2017*, volume 91 of *LIPIcs*, pages 20:1–20:14, 2017. doi:10.4230/LIPIcs.DISC.2017.20.

20 Andrew V. Goldberg, Serge A. Plotkin, and Gregory E. Shannon. Parallel symmetry-breaking in sparse graphs. *SIAM J. Discret. Math.*, 1(4):434–446, 1988. doi:10.1137/0401044.

21 Martin Charles Golumbic. *Algorithmic graph theory and perfect graphs*. Elsevier, 1980.

22 Magnús M. Halldórsson and Christian Konrad. Distributed algorithms for coloring interval graphs. In *DISC 2014*, volume 8784, pages 454–468, 2014. doi:10.1007/978-3-662-45174-8_31.

23 Magnús M. Halldórsson and Christian Konrad. Improved distributed algorithms for coloring interval graphs with application to multicoloring trees. *Theor. Comput. Sci.*, 811:29–41, 2020. doi:10.1016/j.tcs.2018.11.028.

24 Christian Konrad and Viktor Zamaraev. Distributed minimum vertex coloring and maximum independent set in chordal graphs. In *MFCS 2019*, volume 138 of *LIPIcs*, pages 21:1–21:15, 2019. doi:10.4230/LIPIcs.MFCS.2019.21.

25 Fabian Kuhn, Thomas Moscibroda, and Roger Wattenhofer. On the locality of bounded growth. In *PODC 2005*, pages 60–68. ACM, 2005. doi:10.1145/1073814.1073826.

26 Nathan Linial. Locality in distributed graph algorithms. *SIAM J. Comput.*, 21(1):193–201, 1992. doi:10.1137/0221015.

27 Frédéric Maffray. On the coloration of perfect graphs. In *Recent Advances in Algorithms and Combinatorics*, pages 65–84. Springer, 2003.

28 Gary L. Miller and John H. Reif. Parallel tree contraction part 1: Fundamentals. *Adv. Comput. Res.*, 5:47–72, 1989.

29 Alessandro Panconesi and Aravind Srinivasan. The local nature of δ-coloring and its algorithmic applications. *Combinatorica*, 15(2):255–280, 1995.

30 Marinus Johannes Petrus Peeters. On coloring j-unit sphere graphs. Technical report, Tilburg University, School of Economics and Management, 1991.

31 Ram Ramanathan. A unified framework and algorithm for channel assignment in wireless networks. *Wirel. Networks*, 5(2):81–94, 1999. doi:10.1023/A:1019126406181.

32 Johannes Schneider and Roger Wattenhofer. An optimal maximal independent set algorithm for bounded-independence graphs. *Distributed Comput.*, 22(5-6):349–361, 2010. doi:10.1007/s00446-010-0097-1.

33 Lieven Vandenberghe and Martin S. Andersen. Chordal graphs and semidefinite optimization. *Found. Trends Optim.*, 1(4):241–433, 2015. doi:10.1561/2400000006.