Theoretical Computer Science Department
Faculty of Mathematics and Computer Science
Jagiellonian University

Geometric and weight constraints in Online Interval Coloring

Patryk Mikos

PhD dissertation

Supervisor:
prof. dr hab. Paweł Idziak

Co-supervisor:
dr Grzegorz Gutowski

Kraków, January 2020

Contents

Introduction

Some natural problems with time-dependent conflicts, like job scheduling, resource allocation, or network multiplexing are often modeled using intervals on the real line. The real axis naturally represents the time, and an interval can represent a job that needs to be executed in some time window. The fact that two intervals intersect means that the corresponding jobs need to be assigned to different resources (machines, workers, etc.), as the single resource cannot be assigned to different jobs for the same time. There is a natural graph representation of these problems, where each interval corresponds to a vertex, and we introduce an edge between the vertices when the intervals intersect. In general, much more complicated conflicts can be represented by different kinds of graphs. However, in this thesis we focus only on those conflicts that can be expressed by intersecting intervals.

The essence of many real life problems is to minimize the amount of resources used. Graph coloring is a natural way to express this minimization problem under restrictions represented by the given graph: each color represents a resource, and if two vertices share an edge then both cannot be assigned to the same resource. In the general case finding the proper coloring is very hard even if we know that the graph can be colored using 3 colors. On the other hand, the graph coloring problem restricted to the class of interval graphs is easily solvable: the natural greedy algorithm applied to the collection of intervals sorted by their endpoints gives an optimal coloring.

A graph coloring problem, besides its computational complexity, can be always solved since for a given graph G on n vertices, we can simply check all possible colorings of G that uses colors from the set $\{1, \ldots, n\}$. Some of these colorings uses minimum number of colors, and so this is a solution. The graph coloring problem becomes much more difficult if the whole graph is not known in advance, but each vertex is presented one by one, and the coloring algorithm has to assign a color to the presented vertex at the moment it is presented. Algorithm's decision is irrevocable and has to be made without any knowledge about the remaining part of the graph. This variant of the graph coloring problem is called the *online graph coloring*, and in a natural way models a dynamic system in which jobs have to be assigned to resources on the rolling base without any knowledge about the future jobs.

The standard performance measure, used to analyze online algorithms, is the *competitive ratio*, i.e. the worst-case guarantee on the ratio of the cost of solution given by an online algorithm to the cost of optimal offline solution (see Preliminaries for a formal definition). In general case of the online graph coloring there is no algorithm with competitive ratio bounded by a constant [16, 17]. For the class of interval graphs there

is a 3-competitive algorithm, and we know it is an optimal algorithm [26].

A model in which resources cannot be shared between jobs is quite narrow and does not cover many real life scenarios. For instance, computer's memory is a type of resource that can be easily shared between many jobs. To analyze those types of scenarios, a new model of conflicts was introduced. In this new model jobs declare what fraction of a given resource they require, while resources can be shared between many jobs unless for some resource the sum of requirements at some moment exceed the capacity of that resource. The model we talk about is called the *online graph coloring with bandwidths* (or sometimes with weights) and generalizes a well-known and widely studied problem called the *online bin-packing*, see [8, 9, 10] for surveys.

Jobs with very short or very long execution time (for instance longer than the age of the universe) are not real life jobs. Hence, it is reasonable to assume that all job-describing intervals have lengths from a fixed and relatively short range. Moreover, many memory resources are quantified in a natural way and operating systems often limit the maximum amount of memory a single job can get. Thus, it is justified to consider a model in which all jobs are allowed to request at least α-fraction and at most β-fraction of the given resource.

Sometimes the easiest solutions are the best or at least good enough in comparison with the optimal ones. One of the most natural greedy algorithms, called FirstFit, is an example of such simple and easy to implement idea that in some cases gives very satisfying results. It is quite easy to prove that in the graph coloring problem, without any restrictions on graphs, FirstFit does not achieve any constant competitive ratio, even if the underlying graphs are forests. Since in the case of interval graphs FirstFit has competitive ratio between 5 [25] and 8 [31], there is a hope that this natural algorithm might achieve a constant competitive ratio also when applied to the interval coloring problem under reasonable restrictions on the lengths of intervals and requested fractions of resources.

The presentation of our results concerning these variants of the online coloring of interval graphs is organized as follows.

The main result of Chapter 1 is a new lower bound of 4.1626 on the competitive ratio in the online interval coloring with bandwidths which improves the best previously known lower bound of $\frac{24}{7} \approx 3.428$. We also give a new lower bound of 2 in the case where all intervals have the same length improving previous bound of 1.861.

There are many different ways in which interval graphs can be presented online. Obviously, they can be presented simply as graphs, i.e. vertices and adjacent edges. However, vertices might be also presented together with their interval representation on the real line. This kind of presentation can be even more striking when there are some additional requirements those intervals have to satisfy. In Chapter 2 we investigate how the competitive ratio depends on the graph representation. In particular, we consider three variants of the online unit interval graph coloring with bandwidths. In the first variant, only the mere graph is known, while in the other two of them some interval representation is provided. Since it is well known that 'unit = proper' [6, 35], we consider the input in the last two variants to be either proper interval representation or unit interval representation. For each studied variant we provide both lower and upper bounds on the competitive ratio.

Chapter 3 is devoted to the analysis of the competitive ratio in the online interval

coloring problem in which all intervals are restricted to have lengths from some fixed range. We show that if all intervals have lengths from $[1, \sigma]$, then there is a $(1 + \sigma)$-competitive algorithm. Moreover, we prove that for every $\epsilon > 0$ and sufficiently large σ there is no $(5/2 - \epsilon)$-competitive algorithm.

In Chapter 4 we provide the detailed analysis of the FirstFit algorithm. In that chapter we analyze the competitive ratio of this algorithm in different variants of the online interval coloring with bandwidths. In particular, we show that if lengths of intervals are restricted to be in $[1, \sigma]$ and bandwidths are from $[\alpha, \beta]$, then the competitive ratio of the FirstFit algorithm is a function of $\sigma, 1/\alpha$ and $1/(1 - \beta)$.

The ability to quickly verify hypotheses is one of the key elements of conducting scientific research. Graph enumeration algorithms are helpful when we want to test our hypothesis on a quite big set of graphs, and possibly find a small counterexample. In Chapter 5 we present an algorithm that enumerates (and constructs) all non-isomorphic interval graphs on n vertices with worst-case time delay between the output of two consecutive graphs of $O(n^3 \log n)$. Our algorithm beats the best previously known algorithm with worst-case time delay $O(n^4)$.

Preliminaries

For definitions of standard graph-theoretical concepts, we refer the reader to [11]. We also use certain specific notions and notational conventions listed below.

Set notation. For every positive integer k, the set $\{1, \ldots, k\}$ is denoted by $[k]$.

Graphs. We consider only simple undirected graphs without loops and multiple edges. We use the notation $G = (V(G), E(G))$ (or $G = (V, E)$ in short), where $V(G)$ is the finite *vertex set*, and $E(G)$ is the *edge set* of a graph G. Moreover, if G is known from context, we denote the number of vertices $|V(G)|$ by n, and the number of edges $|E(G)|$ by m.

Interval graphs. A graph $G = (V, E)$ with vertex set $V = [n]$ is an *interval graph* if there is a set of closed intervals $\mathcal{I} = \{I_1, \ldots, I_n\}$ on the real line such that $(i, j) \in E$ iff $I_i \cap I_j \neq \emptyset$. The set $\mathcal{I}$ is called an *interval representation* of G. We denote l_i the left endpoint of the interval I_i and r_i the right endpoint of I_i, so $I_i = [l_i, r_i]$. Moreover, we denote the length of I_i as $length(I_i) = r_i - l_i$.

Proper interval graphs. An interval graph is a *proper interval graph* if it has an interval representation $\mathcal{I} = \{I_1, \ldots, I_n\}$ such that $\forall_{i,j} : I_i \subseteq I_j \Rightarrow I_i = I_j$. Such an interval representation is called a *proper interval representation* of G.

Unit interval graphs. An interval graph is a *unit interval graph* if it has an interval representation $\mathcal{I} = \{I_1, \ldots, I_n\}$ such that $\forall_i : r_i - l_i = 1$. Such an interval representation is called a *unit interval representation* of G.

Online graph coloring. An *online graph coloring* is a two-person game, played in rounds by *Presenter* and *Algorithm*. In this game *Presenter* shows the input graph to the *Algorithm* vertex by vertex, along with all the edges adjacent to the already presented vertices. Algorithm must assign a color to each vertex, different than any of its neighbors, immediately and irrevocably at the moment it is presented, without any knowledge of the remaining part of the graph. The goal of Algorithm is to minimize the number of different colors used during the game, while the goal of Presenter is to maximize it.

Online (unit) interval graph coloring. This is a variant of the online graph coloring problem in which the presented graph is an (unit) interval graph.

Online graph coloring with bandwidths. An *online graph coloring with bandwidths* is a variant of the online graph coloring in which every presented vertex has an associated bandwidth, which is a real number from $[0, 1]$. In this problem, two adjacent vertices can be colored with the same color, but we require that for every color γ and every induced subclique Q of the presented graph, the sum of bandwidths of all ver-

tices that belong to Q and are colored with γ is not greater than 1.

Online interval coloring. An *online interval coloring* is a two-person game, played in rounds by *Presenter* and *Algorithm*. In each round Presenter introduces a new closed interval on the real line, and Algorithm assigns a color to the incoming interval in such a way that every two intersecting intervals get different colors. The color of the new interval is assigned before Presenter introduces the next interval and the assignment is irrevocable. The goal of Algorithm is to minimize the number of different colors used during the game, while the goal of Presenter is to maximize it.

Online interval coloring with bandwidths. In this version of the online interval coloring problem Presenter assigns a bandwidth to each presented interval, which is a real number from $[0, 1]$. A coloring is feasible if for each color γ and any point p on the real line, the sum of bandwidths of intervals containing p and colored γ does not exceed 1. Note that, an *online interval coloring with bandwidths* is a simultaneous generalization of two problems - online interval coloring and *online bin packing*.

Online unit/proper interval coloring with bandwidths. Those problems are variants of the online interval coloring with bandwidths in which the presented representation is unit/proper interval representation.

Online σ-interval coloring with $[\alpha, \beta]$-bandwidths. This problem is a variant of the online interval coloring with bandwidths in which all intervals have lengths between 1 and σ, and all bandwidths are between α and β.

Absolute competitive ratio. In the context of various online coloring games, the measure of quality of a strategy for Algorithm is given by the competitive analysis. A coloring strategy for Algorithm is *ϱ-competitive* if it uses at most $\varrho \cdot c$ colors for any c-colorable input. The *absolute competitive ratio* for a problem is the infimum of all values ϱ such that there exists an ϱ-competitive strategy for Algorithm for this problem.

Asymptotic competitive ratio. Let $\chi_A(\mathcal{I})$ be the number of colors used by Algorithm A on the input $\mathcal{I}$, and $OPT(\mathcal{I})$ be the minimum number of colors required to color $\mathcal{I}$. The *asymptotic competitive ratio* for Algorithm A, denoted by $\mathcal{R}_A^\infty$, is defined as follows:

$$\mathcal{R}_A^\infty = \liminf_{k\to\infty}\{\tfrac{\chi_A(\mathcal{I})}{k} : OPT(\mathcal{I}) = k\}$$

The *asymptotic competitive ratio* for a problem is the infimum of all values $\mathcal{R}_A^\infty$ such that A is an Algorithm for this problem. Some authors define asymptotic competitive ratio to be the limit in the size of instance instead of the optimum value. For the problems discussed in this thesis, the competitive ratio defined as the limit in the size of instance equals the absolute competitive ratio (we can add many independent vertices to a 'bad' small instance to make it a 'bad' big instance). Thus, we choose to define the asymptotic competitive ratio as the limit in the optimal value.

1

Online Interval Coloring with Bandwidths

1.1 Introduction

In this chapter we give lower bounds on competitive ratios for the online interval coloring with bandwidths and for the unit version of this problem. We obtain these results by presenting explicit strategies for Presenter that force Algorithm to use many colors while the presented set of intervals is colorable with a smaller number of colors.

1.1.1 Previous work

The competitive ratio for the online interval coloring was established by Kierstead and Trotter [26]. They constructed a strategy for Algorithm that uses at most $3\omega-2$ colors on ω-colorable set of intervals. They also presented a matching lower bound – a strategy for Presenter that forces Algorithm to use at least $3\omega - 2$ colors. The unit variant of the online interval coloring was studied by Epstein and Levy [12]. Authors presented a strategy for Presenter that forces Algorithm to use at least $\left\lfloor \frac{3\omega}{2} \right\rfloor$ colors. Moreover, they showed that a natural greedy algorithm uses at most $2\omega - 1$ colors.

A variant of the online interval coloring with bandwidths in which all intervals have the same endpoints is known as the online bin packing, see [10] for a survey.

Online interval coloring with bandwidths was first posed in 2004. Adamy and Erlebach [2] showed a 195-competitive algorithm for this problem. An improved analysis by Pemmaraju et al. [34] showed that Adamy-Erlebach algorithm has competitive ratio at most 35. Narayanaswamy [32] and Azar et al. [4] presented a 10-competitive algorithm. On the other hand, Epstein and Levy [12] showed a lower bound of 3.2609 for the asymptotic competitive ratio in this problem, and then in [13] they improved it to $\frac{24}{7}$.

Online unit interval coloring with bandwidths was studied by Epstein and Levy [12]. They presented a lower bound of 2 and upper bound of $\frac{7}{2}$ for the absolute competitive ratio in this problem. For the asymptotic competitive ratio, they showed a 3.178-competitive algorithm and a lower bound of 1.831.

1.1.2 Our result

For the online interval coloring with bandwidths, we prove that the asymptotic competitive ratio is at least 4.1626 improving previous best lower bound $\frac{24}{7}$ of [13]. For the

online unit interval coloring with bandwidths, we improve the bound 1.831 by presenting an explicit strategy for Presenter that forces Algorithm to use at least $2k-1$ different colors while the presented set of intervals is k-colorable.

1.2 Presenter strategy

At first we recall a strategy proposed by Kierstead and Trotter for Presenter in the online interval coloring game. We use this strategy as a substrategy in our lower bound.

Theorem 1.1 (Kierstead, Trotter [26]). *For every $\omega \in \mathbb{N}_+$, there is a strategy for Presenter that forces Algorithm to use at least $3\omega - 2$ different colors in the online interval coloring game played on a ω-colorable set of intervals. Moreover, Presenter can play in such a way that every introduced interval is contained in a fixed interval $[L, R]$.*

Below we present a strategy for Presenter in the online interval coloring with bandwidths. We fix some $k \in \mathbb{N}_+$ and will ensure that at any point of the game, the set of intervals introduced by Presenter is k-colorable. To construct our strategy we need several parameters that are going to control the construction.

Definition 1.2. *A pair of sequences $([j_1, \ldots, j_n], [x_1, \ldots, x_n])$ such that $x_i \in \mathbb{N}_+$, $j_i | k$ and $j_1 < j_2 < \ldots < j_n \leqslant \frac{1}{3}k$ is called a* k-schema.

The j_i's are going to control the bandwidths of the intervals presented in the i-th phase, while the x_i's are supposed to describe how many new colors the Algorithm is forced to use in this i-th phase. However, not every k-schema leads to a strategy for Presenter. Later we give additional conditions that a given k-schema has to satisfy to produce a valid strategy.

1.2.1 Strategy description

The strategy for Presenter is based on a k-schema $([j_1, \ldots, j_n], [x_1, \ldots, x_n])$ and consists of 2 main phases: *the separation phase* and *the final phase*. The separation phase consists of n subphases indexed by $1, \ldots, n$. Let $\mathcal{M}$ be the set of *marked intervals*, which initially is empty, and $\mathcal{M}_i$ be the subset of $\mathcal{M}$ containing all marked intervals from the i-th subphase. For each color c used by Algorithm in the separation phase, the set $\mathcal{M}$ contains the first interval colored by Algorithm with c.

All intervals introduced by Presenter in the i-th subphase are contained in the same region $[L_i, R_i]$, have length $s_i = \frac{1}{2}(R_i - L_i)$, and bandwidth $\frac{j_i}{k}$, see Figure 1.1. At each moment in the i-th subphase we keep two values p_i^L and p_i^R. The p_i^L keeps the rightmost right endpoint of non-marked intervals already introduced in the i-th subphase, or $p_i^L = L_i + s_i$ if such an interval does not exist. Analogously, p_i^R keeps the leftmost right endpoint of marked intervals already introduced in the i-th subphase, or $p_i^R = R_i$ if such an interval does not exist. For the first subphase we set $L_1 = 0$, $R_1 = 2$ and for the i-th subphase we set $L_i = p_{i-1}^L$ and $R_i = p_{i-1}^R$ where p_{i-1}^L and p_{i-1}^R are values of those variables at the end of the $(i-1)$-subphase.

In the i-th subphase, Presenter introduces new intervals until exactly x_i new colors are introduced by Algorithm. Let $p_i = \frac{1}{2}(p_i^L + p_i^R)$. A new interval introduced by

Presenter has endpoints $I = [p_i - s_i, p_i]$. If Algorithm colors I with one of the already used colors, then p_i^L changes to p_i. Otherwise, p_i^R changes to p_i and interval I is marked.

Assume that Presenter secretly constructs for himself some optimal coloring after each subphase. Let Γ_i be the number of colors used by Presenter in that coloring on intervals in the set $\mathcal{M}_1 \cup \ldots \cup \mathcal{M}_i$, see Figure 1.2. A precise procedure to determine the Γ_i's is to be given later (after Definition 1.3). Our first restriction on the k-schema we actually use to build our strategy is the requirement that $\Gamma_n \leqslant k$.

In the final phase, Presenter uses the strategy from Theorem 1.1 in the interval $[L_{n+1}, R_{n+1}]$ simply with ω set to $k - \Gamma_n$. Note that the bandwidth of each interval introduced by Presenter in the final phase is simply 1. This completes the description of a strategy for Presenter based on a k-schema.

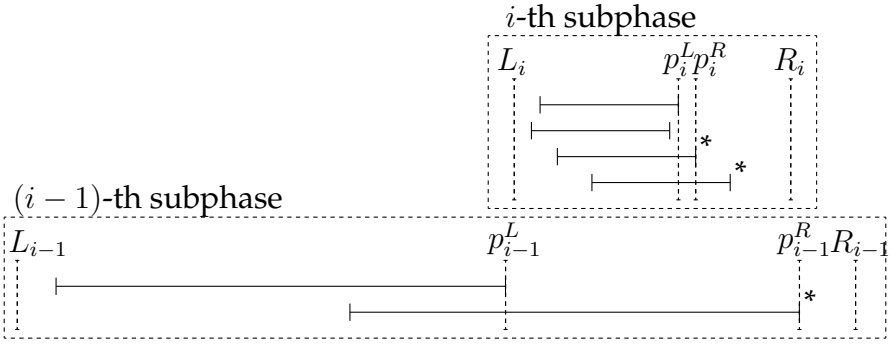

Figure 1.1: Intervals introduced in the i-th subphase in relation to the intervals introduced in the $(i-1)$-th subphase. Marked intervals are marked with *.

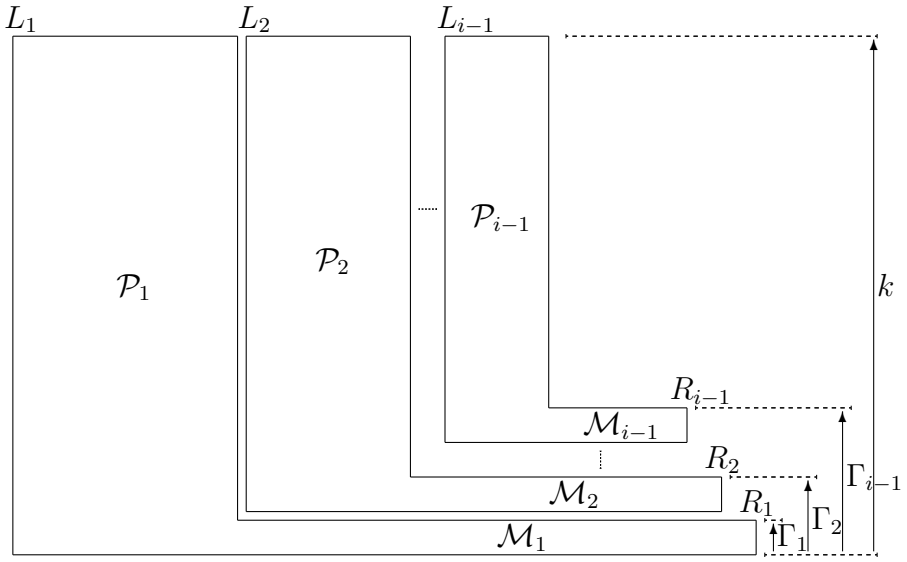

Figure 1.2: The distribution of intervals in the first $i-1$ separation subphases. The L-shape $\mathcal{P}_j$ represents the separation process of the intervals introduced in the j-th subphase, while its bottom part labeled $\mathcal{M}_j$ represents intervals marked in this phase.

1.2.2 Strategy analysis

Now, we show when a given k-schema $([j_1, \ldots, j_n], [x_1, \ldots, x_n])$ actually leads to a valid strategy for Presenter, i.e. when Presenter is able to force Algorithm to use at least x_i new colors in the i-th subphase.

During the i-th subphase, we have $L_i + s_i \leqslant p_i \leqslant R_i$. Thus, the distance between the rightmost right endpoint of an interval introduced in the i-th subphase and the leftmost right endpoint of interval introduced in this phase is at most s_i. Hence, all intervals introduced by Presenter in the i-th subphase form a clique, see Figure 1.1.

Note that to the right of p_{i-1}^L there are only marked intervals from the first $i-1$ subphases. Each interval introduced in the i-th subphase intersects with every interval previously introduced in the i-th subphase and all marked intervals from the first $i-1$ subphases. Thus, if Presenter introduces at most $\frac{k}{j_i}(k - \Gamma_{i-1})$ intervals in the i-th subphase, then all intervals can be colored with k colors.

Let $\chi_i := |\mathcal{M}_1| + \ldots + |\mathcal{M}_i|$ be the number of marked intervals after the i-th subphase, and put $\chi_0 = \Gamma_0 = 0$ for convenience. Intervals introduced in the i-th subphase intersect with exactly χ_{i-1} marked intervals from subphases $1, \ldots, i-1$ and by the definition of the set $\mathcal{M}$ each of them has a different color. Thus, in the interval $[L_i, R_i]$, each color marked in a subphase $1 \leqslant q < i$ has accumulated bandwidth $\frac{j_q}{k}$. We assumed that $j_q < j_i$, hence at most $\frac{k}{j_i} - 1$ intervals can be colored with each particular color that was used in the previous subphases. Thus, in the i-th subphase Algorithm may be forced to use at least

$$\Delta_i := \left\lceil \frac{j_i}{k}\left(\frac{k}{j_i}(k - \Gamma_{i-1}) - \chi_{i-1}\left(\frac{k}{j_i} - 1\right)\right)\right\rceil = k - \Gamma_{i-1} - \chi_{i-1} + \left\lceil \frac{j_i}{k}\chi_{i-1}\right\rceil \tag{1.1}$$

new colors.

Definition 1.3. *A k-schema $([j_1, \ldots, j_n], [x_1, \ldots, x_n])$ is a k-*strategy *if $\Gamma_n \leqslant k$ and $x_i \leqslant \Delta_i$ for all i.*

Now we will see that the presented intervals may be colored by k colors and provide a coloring procedure that uniquely determines the Γ_i's. To this end after the i-th subphase we assign offline colors to the intervals introduced in the i-th subphase as follows. First we use greedy algorithm to color the intervals from $\mathcal{M}_i$ and then to the non-marked intervals from i-th subphase using at most k colors in total. Note that some intervals might be colored with some of the already used Γ_{i-1} colors. Because the number of new marked intervals in the i-th subphase is exactly x_i, each of them has bandwidth $\frac{j_i}{k}$, and those intervals are colored using greedy algorithm, then Γ_i is a properly defined quantity for a given k-schema and depends on all the j_q's and x_q's with $q < i$, but does not depend on the Algorithm's coloring.

This shows that Presenter using a k-strategy $([j_1, \ldots, j_n], [x_1, \ldots, x_n])$ forces Algorithm to use at least $\Sigma_{q=1}^n x_q$ colors in the separation phase and $3(k - \Gamma_n) - 2$ new colors in the final phase, while the set of introduced intervals is k-colorable.

We just completed the analysis of our strategy for a given k-strategy. Now we present a very simple example that already improves the known lower bound $\frac{24}{7}$ to 4.

Example 1.4. *For a fixed $k \in \mathbb{N}_+$ and a very simple k-strategy $([1], [k])$ we have $\Gamma_1 = 1$. Presenter using this strategy forces Algorithm to use at least $k + 3(k-1) - 2 = 4k - 5$ colors, while the set of introduced intervals is k-colorable. Thus, the asymptotic competitive ratio in the online interval coloring with bandwidths is at least 4.*

With a little bit more effort we rise the lower bound to 4.1.

Example 1.5. *Consider the* 120*-strategy given by the values* j_i*,* x_i *and* Γ_i *from the Table 1.1. Presenter using this strategy forces Algorithm to use* $152 + 3(120 - 6) - 2 = 492$ *colors, while the set of introduced intervals is* 120*-colorable. Thus, the absolute competitive ratio for the online interval coloring with bandwidths is at least* $4\frac{1}{10}$*.*

j_i	1	2	3	4	5	6	8	10	12	15	20	24	30
x_i	120	1	1	1	1	1	2	2	2	4	5	4	8
Γ_i	1	2	2	2	2	2	2	2	2	3	4	4	6

Table 1.1: Example of a strategy S_{120}.

To produce the strategy from Example 1.5 we choose k to be 120 and as j_i's we consider all its divisors that are smaller than $\frac{1}{3}k = 40$. Starting from $\Gamma_0 = \chi_0 = 0$, for each i we greedily choose the maximum value of x_i that satisfy $x_i \leqslant \Delta_i$ and compute the corresponding value of Γ_i.

1.2.3 Scalable strategies

Example 1.5 is an example of a k-strategy for Presenter for a fixed k, and gives a lower bound for the absolute competitive ratio. In order to give lower bounds for the asymptotic competitive ratio, we introduce a notion of a *scalable strategy*.

Definition 1.6. *For a given* k*-schema* $S_k = ([j_1, \ldots, j_n], [x_1, \ldots, x_n])$ *and* $a \in \mathbb{N}_+$*, an* ak*-schema* $S_k^a = ([aj_1, \ldots, aj_n], [ax_1, \ldots, ax_n])$ *is called an* a*-scaled* S_k *schema.*

Note that an a-scaled k-strategy might not be an ak-strategy. For example S_{120} is a 120-strategy but S_{120}^3 is not a 360-strategy. To see this, observe that in S_{120}^3 we have $12 = x_{10} > \Delta_{10} = 11$.

To make our further analysis of a-scaled strategies simpler, we assume that a is a multiple of k, so let $z \in \mathbb{N}_+$ and assume that $a = zk$. We still need an additional constraint on a k-strategy S_k that will ensure that S_k^{zk} is a zk^2-strategy. This constraint is another bound on the x_i's and comes from the very same analysis as it was done just before Definition 1.3, but applied to the zk-scaled k-strategy. Working with the multiples of k we are simply allowed to omit the ceiling in (1.1). Indeed in this new setting $\Gamma_i = z\Sigma_{q=1}^{i} j_q x_q$ and $\chi_i = zk\Sigma_{q=1}^{i} x_q$, so that we get another bound on the number of colors used by Algorithm

$$\Delta_i^* := z\big(k^2 + \Sigma_{q=1}^{i-1}(j_i - j_q - k)x_q\big). \tag{1.2}$$

The zk-scaled k-strategy is a zk^2-strategy if $zkx_i \leqslant \Delta_i^*$ for all i. Thus, we have a condition

$$\forall i : x_i \leqslant k + \frac{1}{k}\Sigma_{q=1}^{i-1}(j_i - j_q - k)x_q. \tag{1.3}$$

Definition 1.7. *A k-strategy S_k that additionally satisfies* (1.3) *is called a* scalable strategy.

Note that (1.3) does not depend on z, and so we have the following lemma.

Lemma 1.8. *If a k-strategy S_k satisfies* (1.3)*, then for every $z \in \mathbb{N}_+$ a zk-scaled S_k schema is a zk^2-strategy.*

We just showed that having a scalable strategy S_k we can produce a family of strategies $\mathcal{S} = \{S_k, S_k^k, S_k^{2k}, \ldots\}$. Now we show that all those new strategies are at least as good as S_k in terms of the competitive ratio. Combining the Lemma 1.9 with the fact that these strategies build increasingly bigger instances, we get a lower bound on the asymptotic competitive ratio. Note that each particular strategy certifies an absolute lower bound, but the whole sequence certifies the asymptotic lower bound.

Lemma 1.9. *For every $k, z \in \mathbb{N}_+$ and a scalable k-strategy S_k the competitive ratio guaranteed by the S_k^{zk} strategy is greater than the competitive ratio guaranteed by the S_k strategy.*

Proof. Let $S_k = ([j_1, \ldots, j_n], [x_1, \ldots, x_n])$. Presenter using a strategy S_k forces Algorithm to use $X = \Sigma_{i=1}^n x_i + 3(k - \Gamma_n) - 2$ colors, while the set of intervals introduced by Presenter is k-colorable. Presenter using a strategy S_k^{zk} forces Algorithm to use $\bar{X} = \Sigma_{i=1}^n zkx_i + 3\big(zk^2 - \bar{\Gamma}_n\big) - 2$ colors, while the set of intervals introduced by Presenter is zk^2-colorable. Observe that the number of colors required in greedy coloring of all intervals marked by S_k^{zk} strategy is at most zk times bigger than the number of colors required in greedy coloring of all intervals marked by S_k strategy, i.e. $\bar{\Gamma}_n \leqslant zk\Gamma_n$. Thus, we have $\frac{1}{zk^2}\bar{X} \geqslant \frac{1}{k}X + \frac{2}{k} - \frac{2}{zk^2} > \frac{1}{k}X$. □

Knowing the additional constraint (1.3) that the S_{120} strategy presented in Example 1.5 should satisfy to be a scalable strategy, we modify it and produce a scalable strategy $\bar{S}_{120}$ in the same manner as we do in Example 1.5, but now we choose x_i to be the maximum integer that satisfy both requirements.

Example 1.10. *Consider the scalable 120-strategy given by the values j_i, x_i and Γ_i from the Table 1.2. This strategy implies a lower bound of $4\frac{1}{10} + \frac{2}{120} = 4\frac{7}{60}$ for the asymptotic competitive ratio in the online interval coloring with bandwidths.*

j_i	1	2	3	4	5	6	8	10	12	15	20	24	30
x_i	120	1	1	1	1	1	2	2	2	3	6	4	8
Γ_i	1	2	2	2	2	2	2	2	2	3	4	4	6

Table 1.2: Example of a scalable strategy $\bar{S}_{120}$

1.2.4 Experimental results

In order to obtain our best lower bound for the asymptotic competitive ratio we choose k to be a *highly composed number*. As a sequence $j_1, \ldots, j_n$ we choose consecutive divisors of k and as previously for a sequence $x_1, \ldots, x_n$ we greedily choose the maximum numbers x_i satisfying both $x_i \leqslant \Delta_i$ and (1.3).

Table 1.3 contains the list of lower bounds for the asymptotic competitive ratio we got for some values of k using this method. These data allows us to state the following theorem.

Theorem 1.11. *Asymptotic competitive ratio for the online interval coloring with bandwidths is at least* 4.1626.

k	ratio	k	ratio
60	4.0500000	2162160	4.1624500
120	4.1166667	21621600	4.1624777
360	4.1416667	183783600	4.1625239
840	4.1523809	2327925600	4.1625617
2520	4.1587301	48886437600	4.1625717
7560	4.1607142	321253732800	4.1625883
10080	4.1611111	4497552259200	4.1625893
15120	4.1614417	97821761637600	4.1625961
25200	4.1615873	866421317361600	4.1626008
27720	4.1618326	4043299481020800	4.1626015
110880	4.1621753	12129898443062400	4.1626018
554400	4.1622763	224403121196654400	4.1626043

Table 1.3: A table of asymptotic competitive ratios for different values of k

1.3 Unit Intervals

Now we modify strategy from Section 1.2 to the unit intervals realm. In this case we are restricted to only one subphase of the separation phase and instead of calling (in the final phase) the construction of Kierstead an Trotter (see Theorem 1.1) we simply present a clique.

Theorem 1.12. *For every $k \in \mathbb{N}_+$, there is a strategy for Presenter that forces Algorithm to use at least $2k-1$ different colors in the online unit interval coloring with bandwidths played on a k-colorable set of intervals.*

Proof. For a given $k \in \mathbb{N}_+$, Presenter at first plays only the separation phase of a k-strategy $([1],[k])$. Because $L_1 = 0$, $R_1 = 2$ and $s_1 = \frac{1}{2}(R_1 - L_1)$, every introduced interval has length 1. Moreover, there is a point $p_1 = \frac{1}{2}(p_1^L + p_1^R)$ such that every marked interval has its right endpoint to the right of p_1 and every non-marked interval has its right endpoint to the left of p_1.

Now, Presenter introduces $k-1$ intervals $[p_1, p_1+1]$ of bandwidth 1 each. Every interval introduced in this phase gets a new color. Thus, Algorithm uses $|\mathcal{M}| + k - 1 = 2k-1$ colors in total, while the introduced set of intervals is k-colorable. □

Applying Theorem 1.12 we improve the previous lower bound 1.831 to 2. However, in Chapter 2, with a different technique, we rise this bound to 2.1571.

Corollary 1.13. *Asymptotic competitive ratio for the online unit interval coloring with bandwidths is at least* 2.

2

Different Interval Graph Representations in Online Unit Interval Coloring

2.1 Introduction

In this chapter we investigate how the competitive ratio in the online graph coloring problem for unit interval graphs depends on the graph representation. We consider three online coloring problems regarding the way the graphs are presented: online unit interval graph coloring with bandwidths, online proper interval coloring with bandwidths, and online unit interval coloring with bandwidths.

2.1.1 Previous work

For the classic versions of those problems (without bandwidths) a natural greedy algorithm called FirstFit is 2-competitive [12]. Since FirstFit does not care about the representation, this implies an upper bound on the competitive ratio in all three problems. On the lower bound side, it is quite easy to prove that for any $\epsilon > 0$ there is no $(2-\epsilon)$-competitive algorithm for the online unit interval graph coloring. To see this, just take $N = \lceil \frac{1}{\epsilon} \rceil + 1$ and consider a graph which consists of $\binom{2N}{N} + 1$ disjoint cliques each of size N. Clearly, this graph is N-colorable, so if Algorithm uses more than $2N$ different colors, then the competitive ratio is more than 2. Hence, it uses at most $2N$ different colors and from pigeon hole principle there are two cliques colored by Algorithm with the same set of colors. Without loss of generality, we assume that those colors are $[N]$ and present additional $N-1$ vertices as in the Figure 2.1. Thus, the resulting graph is still N-colorable, and Algorithm was forced to use $2N-1$ different colors. The competitive ratio in this case is at least $2 - \frac{1}{N} > 2 - \epsilon$.

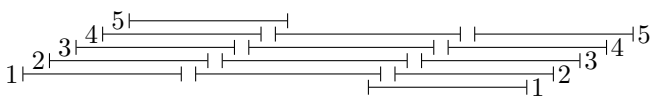

Figure 2.1: The strategy for Presenter in the online unit interval graph coloring.

For the unit representation Epstein and Levy in [12] stated a lower bound $\frac{3}{2}$ which is unbeaten till today. Hence, in case of problems without bandwidths, we know everything about the game played on a mere graph, but there is still quite big gap between $\frac{3}{2}$ and 2 for games played on unit or proper interval graph representations.

Epstein and Levy [12] also presented an Algorithm for the online unit interval coloring with bandwidths which is 3.178-competitive. In the same paper they provided a lower bound of 1.831, but we have already improved it to 2 in Chapter 1. They also proved that a natural greedy algorithm is 8-competitive and presented a variation of FirstFit that is 6-competitive.

Input	Lower Bound		Upper Bound	
Graph Representation	2	Theorem 1.12	6	Epstein & Levy [12]
Proper Interval Representation	2	Theorem 1.12	6	Epstein & Levy [12]
Unit Interval Representation	2	Theorem 1.12	3.178	Epstein & Levy [12]

Table 2.1: Best previously known bounds on the asymptotic competitive ratio in online unit interval coloring with bandwidths.

2.1.2 Our results

We prove that FirstFit achieves an asymptotic competitive ratio of 5 improving the analysis of Epstein and Levy. As it was mentioned before, FirstFit does not care about the representation, so this result improves the best known upper bounds for games played on both: a mere graph and a proper interval representation.

In order to prove our lower bounds, we combined some well known strategies for the bin packing problem with our new constructions for interval graphs. Essentially, we present many instances of some bin packing lower bound strategy, and carefully maintain endpoints of the presented intervals. If Algorithm does not use many colors during the bin packing phase, then we use some additional construction that forces Algorithm to use many new colors, while the minimum number of colors required to properly color the presented instance does not change. Table 2.2 is a summary of our results, and represents the state of the art in this matter.

Input	Lower Bound		Upper Bound	
Graph Representation	$2\frac{653}{994}$	Theorem 2.13	5	Theorem 4.18
Proper Interval Representation	$2\frac{1}{4}$	Theorem 2.14	5	Theorem 4.18
Unit Interval Representation	2.1571	Theorem 2.23	3.178	Epstein & Levy [12]

Table 2.2: Best known bounds on the asymptotic competitive ratio in online unit interval coloring with bandwidths.

2.2 Upper Bounds

As we mentioned before, the best known upper bound for the online unit interval coloring with bandwidths is by Epstein and Levy and was presented in [12]. The following theorem recalls that result.

Theorem 2.1 (Epstein, Levy [12])**.** *There is a* 3.178*-competitive algorithm for the online unit interval coloring with bandwidths.*

Chapter 4 of this thesis is devoted to the detailed analysis of the FirstFit algorithm in various variants of the online interval coloring with bandwidths. Thus, the proof of the following theorem is deferred to the Chapter 4.

Theorem (Reminder of Theorem 4.18)**.** *The maximum color used by the FirstFit algorithm in the online unit interval coloring with bandwidths is bounded from above by* $5 * OPT + 2$.

2.3 Lower Bounds

Before we present our lower bounds, we present the strategy of Epstein and Levy for the online unit interval coloring. We call this strategy an *EL-strategy*, and use it as a subprocedure in various constructions in this chapter and also in Chapter 3.

Theorem 2.2 (Epstein and Levy [12])**.** *For every positive integer* k *there is a strategy for Presenter in online unit interval coloring that forces Algorithm to use at least* $\lfloor \frac{3}{2}k \rfloor$ *colors while the presented set of intervals is* k*-colorable.*

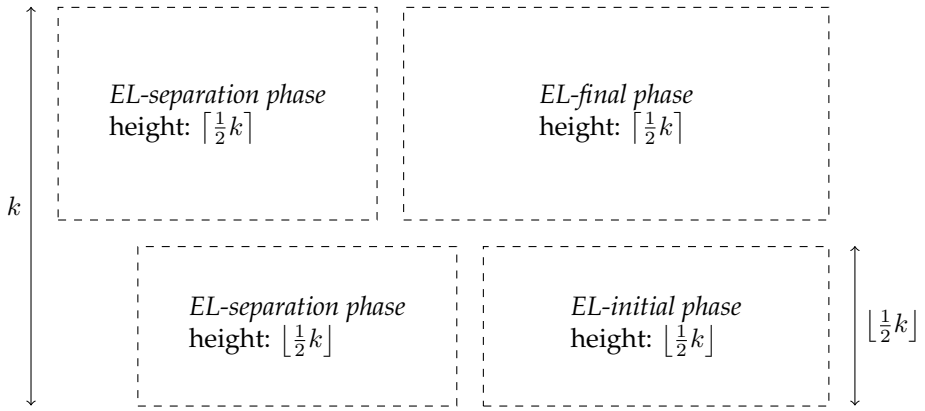

Figure 2.2: Strategy construction in Theorem 2.2.

Proof. The strategy for Presenter consists of three phases, see Figure 2.2. In the first phase, called the *EL-initial phase*, Presenter introduces $\lfloor \frac{1}{2}k \rfloor$ copies of interval $[0, 1]$. Algorithm needs to assign a different color to each copy. Let $\mathcal{X}$ be the set of $\lfloor \frac{1}{2}k \rfloor$ colors used by Algorithm in this phase.

The second phase, is called the *EL-separation phase*. In this phase, Presenter plays the following separation strategy for k rounds. Let $l_1 = -1$ and $r_1 = 0$. In the i-th round of the EL-separation phase Presenter introduces the interval $\left[\frac{l_i+r_i}{2} - 1, \frac{l_i+r_i}{2}\right]$. If Algorithm colors the interval with one of the colors in $\mathcal{X}$, let $l_{i+1} = \frac{l_i+r_i}{2}$ and $r_{i+1} = r_i$, which means that the next interval will be shifted slightly to the right. Otherwise, let $l_{i+1} = l_i$ and $r_{i+1} = \frac{l_i+r_i}{2}$, which means that the next interval will be shifted slightly to the left. Observe that all intervals introduced in the EL-separation phase have length 1 and form a clique of size k. Furthermore, the choice of l_i's and r_i's guarantees that for any two intervals x, y introduced in the EL-separation phase, x colored with a color in $\mathcal{X}$, and y colored with a color not in $\mathcal{X}$, we have that the left endpoint of x is to the left of the left endpoint of y. Let Y be the set of $\lfloor \frac{1}{2}k \rfloor$ right-most intervals introduced in the

EL-separation phase, and let $\mathcal{Y}$ be the set of colors used by Algorithm on the intervals in Y. Note that sets $\mathcal{X}$ and $\mathcal{Y}$ are disjoint.

For the last phase, called the *EL-final phase*, let r be the left-most right endpoint of an interval in Y. In the final phase Presenter introduces $\lceil \frac{1}{2}k \rceil$ copies of interval $[r, r+1]$. This interval intersects all intervals introduced in the EL-initial phase, all intervals in Y, and no other interval introduced in the EL-separation phase. Algorithm must use $\lceil \frac{1}{2}k \rceil$ colors in the EL-final phase that are different from the colors in both $\mathcal{X}$ and $\mathcal{Y}$. Let $\mathcal{Z}$ denote the set of colors used by Algorithm in the EL-final phase. The presented set of intervals is clearly k-colorable and Algorithm used at least $|\mathcal{X}| + |\mathcal{Y}| + |\mathcal{Z}| = 2\lfloor \frac{1}{2}k \rfloor + \lceil \frac{1}{2}k \rceil = \lfloor \frac{3}{2}k \rfloor$ colors. □

Note that the presented strategy introduces only intervals of length 1, but we can rescale this strategy to intervals with the other length.

2.3.1 Graph Representation

First we consider the game in which Presenter does not provide any interval graph representation but a mere graph. We require that at any point of the game, the presented graph is a unit interval graph – it has some unit interval graph representation.

Here we describe a strategy for Presenter in the *online unit interval graph coloring*. Let N be some positive integer divisible by 12, and $\epsilon \in \left(0, \frac{1}{624}\right)$. Define $w_1 = \frac{1}{13} + \epsilon$, $w_2 = \frac{1}{4} + \epsilon$, $w_3 = \frac{1}{3} + \epsilon$, and $N_1 = N_2 = N$, $N_3 = 2N$. Our strategy for Presenter works in at most 3 phases and each phase consists of at most 2 subphases. Actually the second subphase is performed at most once, in fact in the last executed phase. During the first subphases of the game Presenter maintains only a collection of $D := (4N+1)^{9N} + 1$ disjoint cliques that are initially empty. In the first subphase of the i-th phase Presenter enlarges each of those cliques by N_i vertices, each of them of the same bandwidth w_i. If Algorithm uses more than $3N$ colors in total, then Presenter stops the game. Otherwise, Presenter continues the game in the second subphase. In this subphase Presenter considers an application of some special strategy called *strategy* $\frac{3}{2}+$, see Lemma 2.5, but does not apply it immediately. Instead, Presenter computes the guaranteed, by strategy $\frac{3}{2}+$, competitive ratio using Lemma 2.5 and if this ratio is high enough, then applies the aforementioned strategy and finishes the game just after it has been executed. If the guaranteed ratio is too small, then Presenter skips this subphase and continues the game in the $(i+1)$-th phase. We ensure, that if after the first subphase of the third phase Algorithm has still used less than $3N+1$ colors in total, then the competitive ratio guaranteed by strategy $\frac{3}{2}+$ is big enough, so the game eventually terminates after at most 3 phases. That finishes the high-level description of our strategy.

Now we take a look at the details and prove that the strategy guarantees that Algorithm uses at least $2\frac{653}{994}$ times more colors than one needs to properly color the presented instance. First, we prove that if Algorithm uses more than $3N$ colors, then competitive ratio is greater than 3. In this case Presenter stops the game just after some first subphase and the presented graph is just a collection of disjoint cliques. Since $w_1 + w_2 + 2w_3 = \frac{155}{156} + 4\epsilon < 1$, we can color all vertices of the presented graph with N colors. Thus, Algorithm uses at least 3 times more colors than one needs to color the presented graph, and we are done in this case. To analyze all possible games between

Presenter and Algorithm in which Algorithm uses at most $3N$ colors, we introduce some auxiliary objects. The first definition describes the characteristics of a single colored clique.

Definition 2.3. *For a clique Q of vertices with bandwidths let $\psi_Q(c, b)$ be the number of vertices in Q that have bandwidth b and are colored with a color c. We say that function ψ_Q is a* characteristic vector *of Q.*

If Q is known from the context, we simply write ψ instead of ψ_Q. Now, we make a simple observation on characteristic vectors.

Observation 2.4. *If Algorithm uses at most $3N$ different colors in total, then there are at most $(4N+1)^{9N}$ different characteristic vectors among all presented cliques.*

Proof. We assume that all colors used by Algorithm are from the set $[3N]$. Each clique contains at most $N_1 + N_2 + N_3 = 4N$ vertices, and each vertex has associated one of three possible bandwidths. Thus, there is no more than $(4N+1)^{3*3N}$ functions of the form $f : [3N] \times [3] \to \{0, \ldots, 4N\}$. □

A consequence of the above observation is the fact that if Algorithm uses at most $3N$ different colors, then there are at least two different cliques having the same characteristic vector. Note that a collection of characteristic vectors for all presented cliques uniquely encodes the game between Presenter and Algorithm when restricted to our D cliques (up to relabeling vertices that have the same bandwidth and were colored with the same color). Thus, by considering all possible collections of D characteristic vectors we analyze all possible games in which Algorithm uses at most $3N$ different colors. In fact, we do not even need to analyze all possible collections of characteristic vectors but only one of them. This is because the characteristic vectors are used only in second subphases, fully analyzed in Lemma 2.5, and such subphase introduces new vertices that connect two particular cliques with the same characteristic vector. Thus, only one such characteristic vector is in fact used. In fact Lemma 2.5 analyzes how many new colors can be enforced in a second subphase depending on this particular characteristic vector common for two different cliques. For this analysis, in each clique we mark the first vertex in each color used in this clique by Algorithm.

Lemma 2.5 (The strategy $\frac{3}{2}+$.)**.** *Let Q_1 and Q_2 be two cliques having the same characteristic vector. Suppose that both cliques can be colored offline by H colors in such a way that exactly h colors are used to color all marked vertices in each clique. Then there is a strategy for Presenter to enlarge the graph $Q_1 \cup Q_2$ (to a new unit interval graph) which forces Algorithm to use at least $\frac{3}{2}H - \frac{1}{2}h - 2$ additional colors with the enlarged graph being still H-colorable.*

Proof. We shall describe a strategy for Presenter that adds new vertices to the graph consisting of Q_1 and Q_2. In order to prove that the enlarged graph is a unit interval graph, we provide its proper interval representation, see Figure 2.3. We also construct an offline coloring of the enlarged graph with H colors and call it the *Presenter's coloring*. To avoid confusion when talking about Algorithm's coloring and Presenter's coloring, we refer to colors used by Algorithm as *colors* and colors used by Presenter as *hues*.

Without loss of generality, let Φ be an H-coloring of $Q_1 \cup Q_2$ such that all marked vertices are colored in Φ with colors from $[h]$. For $i \in [H]$ and $j \in [2]$ let C_i^j be the set of vertices from the clique Q_j that are colored in Φ with a color i. Since $\psi_{Q_1} \equiv \psi_{Q_2}$, without loss of generality we may assume that vertices from the set C_i^1 are colored by Algorithm with the same set of colors as vertices from the set C_i^2. Additionally, note that some not marked vertices may also be colored in Φ with colors from $[h]$.

Now, Presenter starts to enlarge $Q_1 \cup Q_2$ by adding new vertices, each of bandwidth 1. At first, Presenter introduces $h-1$ new vertices $I_1, \ldots, I_{h-1}$ in such a way that the neighborhood of I_i at the moment it is presented is exactly $N(I_i) = (C_1^2 \cup \ldots \cup C_i^2) \cup \left(C_{i+1}^1 \cup \ldots \cup C_h^1\right) \cup \{I_1, \ldots, I_{i-1}\}$. Since the neighborhood of I_i is already colored with all previously used colors, the Algorithm has to assign a new color to each I_i.

Finally, Presenter plays the EL-separation strategy for $k = H-h$, see Theorem 2.2, in such a way that all vertices from EL-initial and EL-final phase are additionally adjacent to all vertices from the set $\{I_1, \ldots, I_{h-1}\} \cup (C_1^2 \cup \ldots \cup C_h^2)$ and all vertices from EL-separation phase are additionally adjacent to all vertices from the set $\{I_1, \ldots, I_{h-1}\} \cup (C_1^1 \cup \ldots \cup C_h^1)$. Thus, all colors used by Algorithm in this phase are different from the previously used colors, and Algorithm was forced to use at least $h-1+2\left\lfloor \frac{1}{2}(H-h)\right\rfloor + \left\lceil \frac{1}{2}(H-h)\right\rceil = H + \left\lfloor \frac{1}{2}(H-h)\right\rfloor - 1 \geqslant \frac{3}{2}H - \frac{1}{2}h - 2$ new colors.

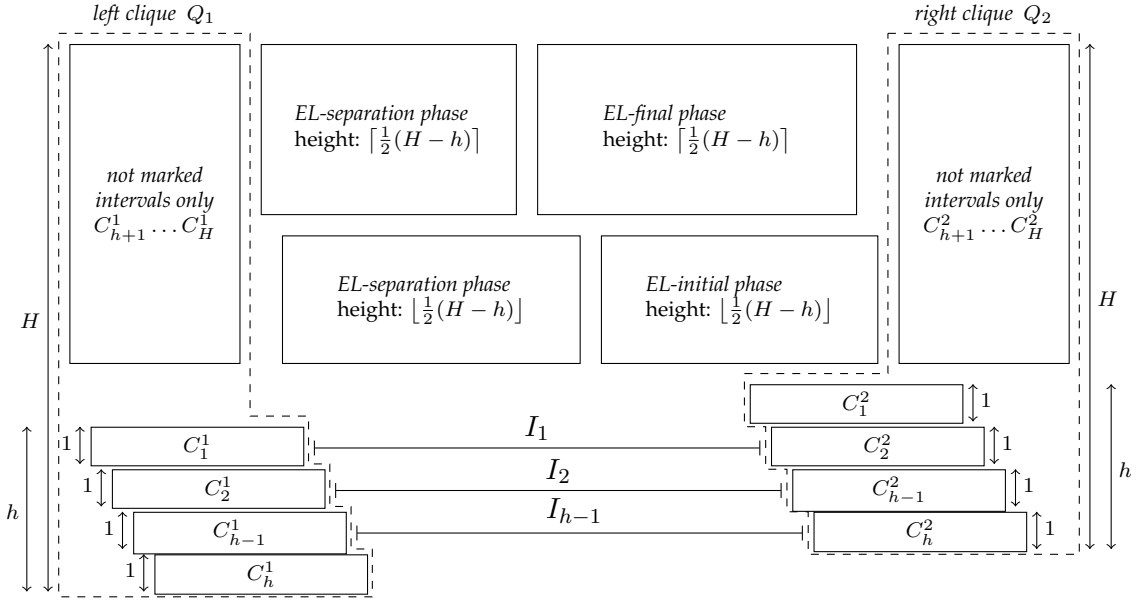

Figure 2.3: Strategy construction for Lemma 2.5

We show that the resulting unit interval graph is H-colorable. The EL-separation strategy is played for $k = H - h$, so clearly we can color all vertices presented during this phase using hues from the set $\{h+1, \ldots, H\}$. Now, we color all vertices from sets C_h^1 and C_1^2 with a hue h, and all vertices from sets $C_i^1 \cup C_{i+1}^2 \cup \{I_i\}$ with a hue i for $i \in [h-1]$. Finally, we color all vertices from the set $C_i^1 \cup C_i^2$ with a hue i for $i \in \{h+1, \ldots, H\}$. It is easy to check that the presented coloring is a proper coloring, and we are done. □

We just presented the *strategy* $\frac{3}{2}+$. Now we show when we can make use of it. First, we define a collection of variables $X_{a,b,c}$ based on a characteristic vector ψ.

Definition 2.6. *For a given colored clique of vertices with bandwidths let $X_{a,b,c}$ be the number of different colors γ such that exactly a vertices with bandwidth w_1, b vertices with bandwidth w_2 and c vertices with bandwidth w_3 are colored by γ. A* configuration *is a vector of numbers $X_{a,b,c}$ where (a, b, c) ranges over all triples satisfying $aw_1 + bw_2 + cw_3 \leqslant 1$.*

Note that configuration is a natural function of characteristic vector

$$X_{a,b,c} = |\{\gamma : \psi(\gamma, w_1) = a, \psi(\gamma, w_2) = b, \psi(\gamma, w_3) = c\}|$$

We define some auxiliary variables M_i that denote the number of marked vertices in each phase, ie. $M_1 = \sum_{a>0,b,c} X_{a,b,c}$, $M_2 = \sum_{b>0,c} X_{0,b,c}$, and $M_3 = \sum_{c>0} X_{0,0,c}$. We call vertices presented in the first phase *light vertices*, in the second phase *medium vertices* and in the last phase *heavy vertices*. In the next three lemmas we analyze how many additional colors Presenter can force on Algorithm in each of the three phases.

Lemma 2.7. *The strategy $\frac{3}{2}+$ is applicable in the first phase with $H = \frac{1}{12}N$ and $h = \lceil \frac{1}{12}M_1 \rceil$.*

Proof. In the first phase Presenter shows light vertices only, so exactly 12 vertices can be colored with one color. We color all marked vertices and $12\lceil \frac{1}{12}M_1 \rceil - M_1$ not marked ones with $h = \lceil \frac{1}{12}M_1 \rceil$ colors. To color the remaining not marked vertices we require additional $\lceil \frac{1}{12}\big(N - 12\lceil \frac{1}{12}M_1 \rceil\big) \rceil$ colors. Thus, we color the whole instance using $\lceil \frac{1}{12}N - \lceil \frac{1}{12}M_1 \rceil \rceil + \lceil \frac{1}{12}M_1 \rceil = \frac{1}{12}N$ colors since N is divisible by 12. □

Lemma 2.8. *If $M_1 \leqslant M_2$, then the strategy $\frac{3}{2}+$ is applicable in the second phase with $H = \frac{1}{3}N$ and $h = \lceil \frac{1}{3}M_2 \rceil$.*

Proof. Since $\frac{1}{13} + \epsilon + \frac{1}{4} + \epsilon = \frac{17}{52} + 2\epsilon < \frac{1}{3}$, we can color 3 light vertices and 3 medium vertices using the same color. We label light vertices $v_1^l, \ldots, v_N^l$ in such a way that marked vertices come first. Analogously, we label medium vertices $v_1^m, \ldots, v_N^m$. Now, for every $i \in \lceil \frac{1}{3}N \rceil$ we color six vertices $v_{3i-2}^l, v_{3i-1}^l, v_{3i}^l, v_{3i-2}^m, v_{3i-1}^m, v_{3i}^m$ with a color i. Thus, we properly colored all vertices using $\frac{1}{3}N$ colors, and it is easy to see that every marked vertex is colored with a color at most $\lceil \frac{1}{3}M_2 \rceil$. □

Lemma 2.9. *If $M_1 \leqslant M_2$ and $2M_2 \leqslant M_3$, then the strategy $\frac{3}{2}+$ is applicable in the third phase with $H = N$ and $h = \lceil \frac{1}{2}M_3 \rceil$*

Proof. Since $\frac{1}{13} + \epsilon + \frac{1}{4} + \epsilon + \frac{2}{3} + 2\epsilon = \frac{155}{156} + 4\epsilon < 1$, we can color 1 light vertex, 1 medium vertex and 2 heavy vertices using the same color. As in the previous lemma, we label light vertices $v_1^l, \ldots, v_N^l$, medium vertices $v_1^m, \ldots, v_N^m$, and heavy ones $v_1^h, \ldots, v_{2N}^h$. For every $i \in [N]$ we color four vertices $v_i^l, v_i^m, v_{2i-1}^h, v_{2i}^h$ with a color i. Yet again, we properly colored all vertices using N colors, and every marked vertex is colored with a color at most $\lceil \frac{1}{2}M_3 \rceil$. □

Corollary 2.10. *If Presenter decides to play the strategy $\frac{3}{2}+$ in the first phase, then Algorithm uses at least $\frac{3}{24}N + \frac{23}{24}M_1 - 3$ colors to color the resulting graph, while it can be properly colored with at most $\frac{1}{12}N$ colors.*

Corollary 2.11. *If $M_1 \leqslant M_2$ and Presenter decides to play the strategy $\frac{3}{2}+$ in the second phase, then Algorithm uses at least $\frac{1}{2}N + M_1 + \frac{5}{6}M_2 - 3$ colors to color the resulting graph, while it can be properly colored with at most $\frac{1}{3}N$ colors.*

Corollary 2.12. *If $M_1 \leqslant M_2$, $2M_2 \leqslant M_3$ and Presenter decides to play the strategy $\frac{3}{2}+$ in the third phase, then Algorithm uses at least $\frac{3}{2}N + M_1 + M_2 + \frac{3}{4}M_3 - 3$ colors to color the resulting graph, while it can be properly colored with at most N colors.*

A consequence of the above corollaries is that Presenter using the strategy $\frac{3}{2}+$ in the first phase achieves a competitive ratio at least $\frac{3}{2} + \frac{23}{2}\frac{M_1}{N} - \frac{36}{N}$, in the second phase at least $\frac{3}{2} + 3\frac{M_1}{N} + \frac{5}{2}\frac{M_2}{N} - \frac{9}{N}$, and in the last phase at least $\frac{3}{2} + \frac{M_1}{N} + \frac{M_2}{N} + \frac{3}{4}\frac{M_3}{N} - \frac{3}{N}$. Our analysis of competitive ratio achieved by all possible characteristic vectors ψ, that can be used to apply the strategy $\frac{3}{2}+$, is based on a linear program with branches, see Figure 2.4.

$$
\begin{array}{lll}
\min & C_N & \\
\text{s.t.} & X_{a,b,c}, M_1, M_2, M_3 \in \mathbb{N} & \\
& \displaystyle\sum_{abc} aX_{a,b,c} = N & \textit{\#vertices presented in 1st phase} \\
& \displaystyle\sum_{abc} bX_{a,b,c} = N & \textit{\#vertices presented in 2nd phase} \\
& \displaystyle\sum_{abc} cX_{a,b,c} = 2N & \textit{\#vertices presented in 3rd phase} \\
& \displaystyle\sum_{a>0,b,c} X_{a,b,c} = M_1 & \textit{\#new colors in 1st phase} \\
& \displaystyle\sum_{b>0,c} X_{0,b,c} = M_2 & \textit{\#new colors in 2nd phase} \\
& \displaystyle\sum_{c>0} X_{0,0,c} = M_3 & \textit{\#new colors in 3rd phase} \\
& C_N \geqslant \dfrac{3}{2} + \dfrac{23}{2N}M_1 - \dfrac{36}{N} & \\
& C_N \geqslant \dfrac{3}{2} + \dfrac{3}{N}M_1 + \dfrac{5}{2N}M_2 - \dfrac{9}{N} & \textit{applicable under } M_1 \leqslant M_2 \\
& C_N \geqslant \dfrac{3}{2} + \dfrac{1}{N}M_1 + \dfrac{1}{N}M_2 + \dfrac{3}{4N}M_3 - \dfrac{3}{N} & \textit{applicable under } 2M_2 \leqslant M_3
\end{array}
$$

Figure 2.4: The linear program with branches encoding the problem of finding the minimal competitive ratio in the described game for a given N.

To solve this linear program, we consider 3 independent linear programs:

- $M_1 \geqslant M_2$, without the last two constraints,
- $M_1 \leqslant M_2$, $2M_2 \geqslant M_3$, without the last constraint, and
- $M_1 \leqslant M_2$, $2M_2 \leqslant M_3$.

We do not solve the presented linear program directly, since it is almost an integer linear program. Note that all variables except C_N are positive integers, while C_N can be a real number. In fact one can easily transform this program into an integer linear

program. To do so it is enough to multiply both sides of the last three inequalities by $4N$ and introduce a new variable $C^* = 4NC_N$. The modified program then minimizes C^* and the sought competitive ratio is $\frac{1}{4N}C^*$.

Instead, we divide both sides of the equalities by N, and introduce new variables $x_{a,b,c} = \frac{1}{N}X_{a,b,c}$ and $q_i = \frac{1}{N}M_i$. Moreover, we move constant factors from the right hand side of inequalities to the left hand side and as a result we get a linear program presented in Figure 2.5 (left).

Now consider an approximation to the modified program presented in Figure 2.5 (right) and observe that for every N we have $C_N^{'} \geqslant \phi^{'} - \frac{36}{N}$ where $C_N^{'}$ is the solution of the modified program, and $\phi^{'}$ is the solution of the approximation. Thus, instead of solving the original hard program, we can solve an easy approximation and get a lower bound on the asymptotic competitive ratio.

Original program:			Approximation:		
min	C_N		min	ϕ	
s.t.	$\sum_{abc} ax_{a,b,c} = 1$		s.t.	$\sum_{abc} ax_{a,b,c} = 1$	
	$\sum_{abc} bx_{a,b,c} = 1$			$\sum_{abc} bx_{a,b,c} = 1$	
	$\sum_{abc} cx_{a,b,c} = 2$			$\sum_{abc} cx_{a,b,c} = 2$	
	$\sum_{a>0,b,c} x_{a,b,c} = q_1$			$\sum_{a>0,b,c} x_{a,b,c} = q_1$	
	$\sum_{b>0,c} x_{0,b,c} = q_2$			$\sum_{b>0,c} x_{0,b,c} = q_2$	
	$\sum_{c>0} x_{0,0,c} = q_3$			$\sum_{c>0} x_{0,0,c} = q_3$	
	$C_N + \frac{36}{N} \geqslant \frac{3}{2} + \frac{23}{2}q_1$			$\phi \geqslant \frac{3}{2} + \frac{23}{2}q_1$	
	$C_N + \frac{9}{N} \geqslant \frac{3}{2} + 3q_1 + \frac{5}{2}q_2$	$\mid q_1 \leqslant q_2$		$\phi \geqslant \frac{3}{2} + 3q_1 + \frac{5}{2}q_2$	$\mid q_1 \leqslant q_2$
	$C_N + \frac{3}{N} \geqslant \frac{3}{2} + q_1 + q_2 + \frac{3}{4}q_3$	$\mid 2q_2 \leqslant q_3$		$\phi \geqslant \frac{3}{2} + q_1 + q_2 + \frac{3}{4}q_3$	$\mid 2q_2 \leqslant q_3$
	$x_{a,b,c}, q_1, q_2, q_3 \in \left\{\frac{i}{N} : i \in \mathbb{N}\right\}$			$x_{a,b,c}, q_1, q_2, q_3 \geqslant 0$	

Figure 2.5: Original linear program after modification (left) and its approximation (right).

The optimal solution for the approximation linear program in the branch $q_1 \geqslant q_2$ is $\phi = \frac{155}{42} \approx 3.69$. For the branch $q_1 \leqslant q_2$, $2q_2 \geqslant q_3$ we have optimal solution $\phi = \frac{79}{28} \approx 2.82$, and finally for the branch $q_1 \leqslant q_2$, $2q_2 \leqslant q_3$ the optimal solution is $\phi = \frac{2641}{994} \approx 2.6569$.

Theorem 2.13. *The asymptotic competitive ratio for the online unit interval graph coloring problem is at least* $2\frac{653}{994} \approx 2.6569$.

2.3.2 Proper Interval Representation

Now we consider the game in which Presenter provides a proper interval graph representation. In this variant we show that Presenter has a strategy that forces Algorithm to use at least $2\frac{1}{4}$ times more colors that is required to properly color the presented set of intervals.

Our strategy for Presenter consists of 2 phases called *separation phase* and *final phase*. Let N be some positive integer. In the separation phase Presenter introduces two cliques Q_1 and Q_2. Each of presented cliques contains exactly $3N^2$ intervals and each of them has bandwidth $\frac{1}{3N}$. As in the online unit interval graph coloring, we say that an interval I belonging to a clique Q_i is marked if I is the earliest interval from Q_i colored with its color.

At first, Presenter introduces intervals from the clique Q_1 ensuring that there is a point p_L such that the right endpoints of all marked intervals from Q_1 are to the right of p_L and the right endpoints of all not marked intervals from Q_1 are to the left of p_L. Then, Presenter introduces intervals from the clique Q_2 ensuring that all of them are to the right of Q_1 and there is a point p_R such that the left endpoints of all marked intervals from Q_2 are to the left of p_R and the left endpoints of all not marked intervals from Q_2 are to the right of p_R. Let M_1 be the set of marked intervals from Q_1 and M_2 be the set of marked intervals from M_2, see Figure 2.6.

Observe that the presented set of intervals is N-colorable. Hence, if Algorithm used more than $3N$ different colors in some clique Q_i, then it used at least 3 times more colors than necessary and we are done. So, assume that Algorithm used less than $3N$ different colors in each clique. Thus, all marked intervals can be properly colored with 1 color.

All intervals presented in the final phase have bandwidth 1. There are two different ways to play the final phase, and Presenter chooses the correct one depending on the number of common marked colors. For $i \in [2]$ let C_i be the set of colors used by Algorithm to color intervals from M_i.

Case 1: $|C_1 \cup C_2| \geqslant \frac{5}{4}N$
If Algorithm used at least $\frac{5}{4}N$ different colors, then in the final phase Presenter introduces $N-1$ additional intervals $[p_L, p_R]$. Clearly, Algorithm has to use new colors to color the presented intervals. Hence, in this case Algorithm used at least $|C_1 \cup C_2| + N - 1 \geqslant \frac{9}{4}N - 1$ colors, while the presented graph is clearly N-colorable.

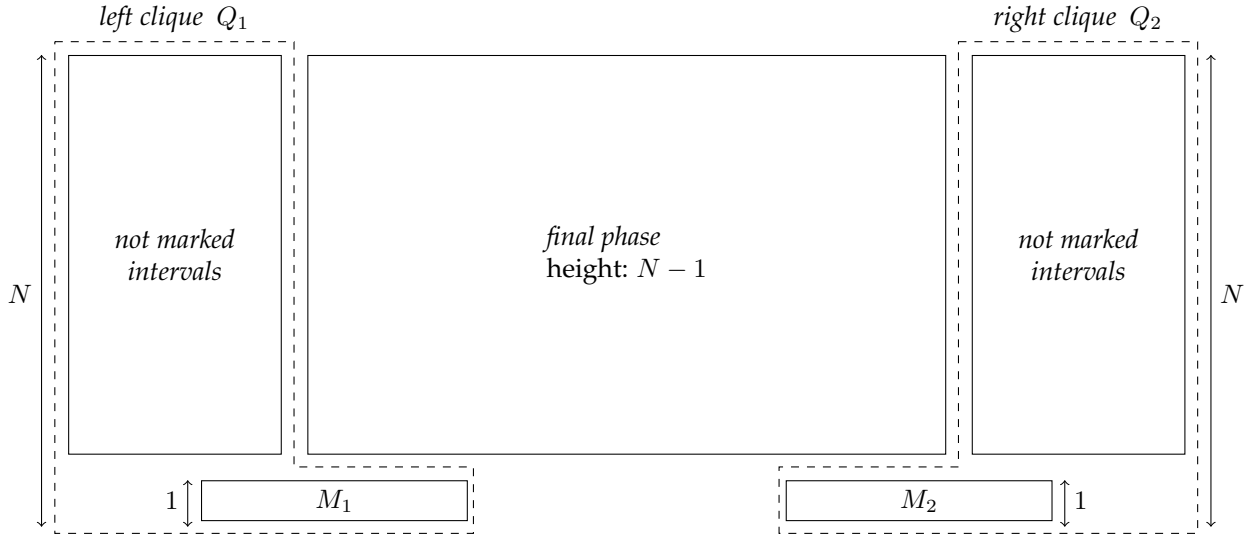

Figure 2.6: Strategy construction in case $|C_1 \cup C_2| \geqslant \frac{5}{4}N$.

Case 2: $|C_1 \cup C_2| \leqslant \frac{5}{4}N$, or equivalently $|C_1 \cap C_2| \geqslant \frac{3}{4}N$
Observe that Algorithm has to use at least N colors on each clique, so $|C_1| \geqslant N$ and $|C_2| \geqslant N$. Moreover, $|C_1| + |C_2| = |C_1 \cup C_2| + |C_1 \cap C_2|$, so if $|C_1 \cup C_2| \leqslant \frac{5}{4}N$, then $|C_1 \cap C_2| \geqslant \frac{3}{4}N$.

If Algorithm used at least $\frac{3}{4}N$ common colors in both cliques, then in the final phase Presenter plays the EL-separation for $k = N - 1$ in a similar way as it does in the strategy $\frac{3}{2}+$. So, all intervals from EL-initial and EL-final phase intersect all intervals from the set M_2 and all intervals from EL-separation phase intersect all intervals from the set M_1, see Figure 2.7. Thus, all colors used by Algorithm in this phase do not belong to $C_1 \cap C_2$, and Algorithm was forced to use at least $|C_1 \cap C_2| + 2\left\lfloor \frac{1}{2}(N-1) \right\rfloor + \left\lceil \frac{1}{2}(N-1) \right\rceil \geqslant \frac{9}{4}N - 2$ colors in total. It is easy to see that the resulting set of intervals is N-colorable, since all intervals introduced during the EL-separation can be colored with colors from $[N-1]$, we can color all marked intervals with a color N, and all not marked intervals from cliques Q_1 and Q_2 with colors from $[N-1]$.

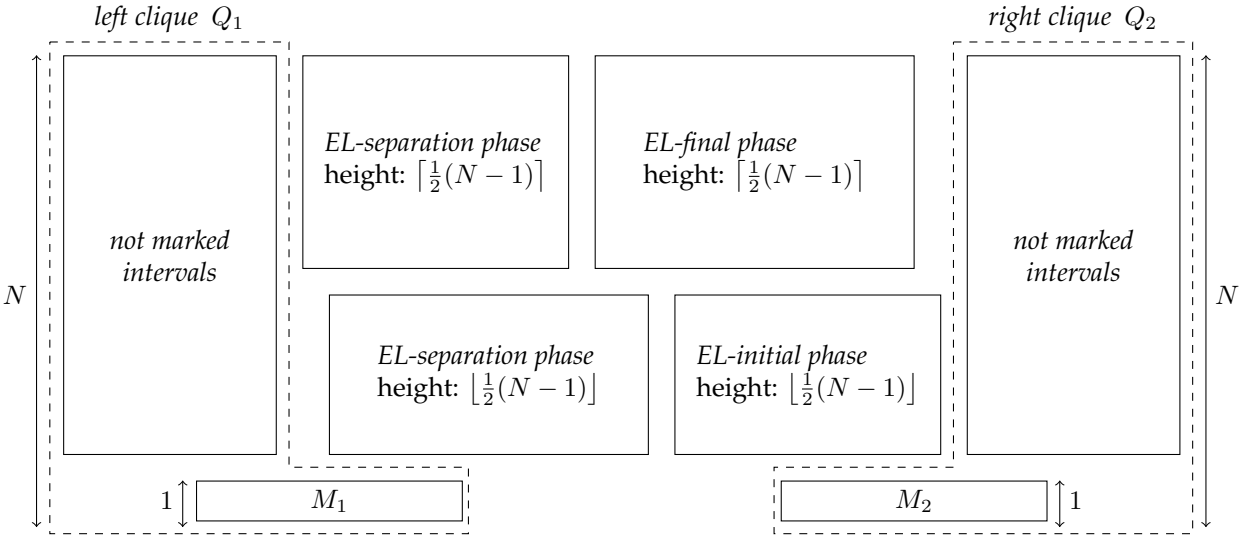

Figure 2.7: Strategy construction in case $|C_1 \cap C_2| \geqslant \frac{3}{4}N$.

We summarize our considerations in the following theorem.

Theorem 2.14. *For every positive integer N there is a strategy for Presenter in the online proper interval coloring, that forces Algorithm to use at least $\frac{9}{4}N - 2$ different colors, while the presented set of intervals is N-colorable.*

Corollary 2.15. *Both absolute and asymptotic competitive ratios in the online proper interval coloring are at least $2\frac{1}{4}$.*

Clearly, the presented strategy works when Presenter can provide a proper interval representation, but it does not work when it is forced to provide a unit interval representation. The problem lays in the final phase when Presenter has two different strategies to finish the game. Observe that the strategy used in Case 1 in the unit case requires that the distance between p_L and p_R is close to 1, while we need this distance to be roughly 2 in order to perform the EL-strategy in Case 2 in the unit intervals realm.

2.3.3 Unit Interval Representation

The last remaining problem to consider is the game in which Presenter provides a unit interval representation. In this variant we show a strategy for Presenter that forces

Algorithm to use at least 2.1571 times more colors than is needed to properly color the presented set of intervals.

Our strategy for Presenter in this case is a mixture of bin packing and interval separation technique. In order to simplify our description of the strategy, we split it into two parts.

First we show how to choose endpoints for new intervals. Here we modify the notion of a marked interval to be now the earliest shown interval colored with its own color. Our strategy for Presenter maintains two disjoint cliques Q_1 and Q_2 such that Q_1 is to the left of Q_2. During the game Presenter ensures that right endpoints of all marked intervals from Q_1 are the rightmost endpoints in Q_1, and left endpoints of all marked intervals from Q_2 are the leftmost endpoints in Q_2. Moreover, it makes sure that the cliques Q_1 and Q_2 are close enough, so that there is the unit interval I that intersects all marked intervals but none of not marked ones. To achieve this goal, Presenter uses two variables a_1, b_1 to control intervals added to Q_1 and another two variables a_2, b_2 for intervals added to Q_2. At the very beginning of the game $a_1 = 0$, $b_1 = \frac{1}{2}$, $a_2 = 1$, and $b_2 = \frac{3}{2}$. While adding a new interval to Q_1, an interval $[q-1, q]$, where $q = \frac{1}{2}(a_1 + b_1)$, is presented. If the Algorithm uses a new color for the presented interval, then Presenter changes b_1 to q and b_2 to $q+1$. Otherwise, Presenter changes a_1 to q and a_2 to $q+1$. Analogously, if Presenter wants to add a new interval to Q_2, then it presents an interval $[q, q+1]$ where $q = \frac{1}{2}(a_2 + b_2)$. If the presented interval is marked, then Presenter changes a_1 to $q-1$ and a_2 to q. Otherwise, Presenter changes b_1 to $q-1$ and b_2 to q.

To observe that Presenter using this strategy shows in fact two disjoint clique, note first that all intervals from Q_1 contain the point 0, and all intervals from Q_2 contain the point $\frac{3}{2}$. Thus, both sets of intervals in fact represent cliques. Moreover, none of the presented intervals intersect the interval $\left(\frac{1}{2}, 1\right)$. Hence, Q_1 and Q_2 are disjoint. Finally, it is easy to check that, at each moment, the interval $[p, p+1]$ where $p = \frac{1}{2}(a_1 + b_1)$ intersects all marked intervals and none of not marked ones. This finishes the description of our interval separation technique.

Now we describe the part corresponding to bandwidths. This part is very similar to the strategy we presented for the online unit interval graph coloring in Section 2.3.1, but now we have at most 2 phases instead of 3, and we use different special strategy. Again let N be some positive integer divisible by 12, $\epsilon \in \left(0, \frac{1}{624}\right)$ and $w_1 = \frac{1}{13} + \epsilon$, $w_2 = \frac{1}{4} + \epsilon$. However, this time to control the sizes of the cliques presented in the i-th phase we need two additional numbers $K_1 = \frac{1}{12}N^2$ and $K_2 = \frac{1}{4}N^2$. All intervals in Q_1 have bandwidth $\frac{1}{N}$, while intervals in Q_2 presented in the i-th phase have bandwidth w_i. In the first subphase of the i-th phase Presenter at first introduces K_i new intervals belonging to Q_1, and then N new intervals belonging to Q_2. If Algorithm uses more than N colors in total, then Presenter stops the game. Otherwise, Presenter continues the game in the second subphase. In this subphase, as in Section 2.3.1, Presenter uses some special strategy called *strategy* $1+$, see Lemma 2.16, but only if the ratio guaranteed by this strategy is high enough. If not, then Presenter skips this subphase and continues the game in the $(i+1)$-th phase. Yet again, we ensure that if after the first subphase of the second phase Algorithm has still used less than $N+1$ colors in total, then the competitive ratio guaranteed by strategy $1+$ is big enough, so the game even-

tually terminates after at most 2 phases. This finishes the high-level description of our strategy.

Lemma 2.16 (The strategy $1+$.)**.** *Suppose that Algorithm uses at most N colors on intervals from $Q_1 \cup Q_2$. Moreover, assume that $Q_1 \cup Q_2$ can be colored offline by H colors in such a way that exactly h colors are used to color all marked intervals. Then there is a strategy for Presenter, to expand the unit interval representation of $Q_1 \cup Q_2$, that forces Algorithm to use $H - h - 1$ additional colors, and the resulting set of unit intervals is still H-colorable.*

Proof. In order to force Algorithm to use additional colors Presenter introduces $H-h-1$ intervals $[p, p+1]$, where $p = \frac{1}{2}(a_1 + b_1)$, each of them of bandwidth 1. As we already noticed, those intervals intersect all marked intervals and do not intersect any other one. Hence, Algorithm has to use new colors to color the additional intervals.

Now we show that we can properly color the presented set of intervals using at most H hues in total. All intervals in the clique Q_1 have bandwidths $\frac{1}{N}$. Hence, if Algorithm used at most N different colors, then we are able to color Q_1 in such a way that all marked intervals in Q_1 receive a hue 1. Without loss of generality, we assume that we can color Q_2 with H hues in such a way that all marked intervals in Q_2 are colored with hues from $[h]$. Hence, we can use hues $h + 2, \ldots, H$ to color the last $H - h - 1$ intervals, and we are done. □

Now we prove that the presented strategy guarantees that Algorithm uses at least 2.1517 times more colors than one needs to properly color the presented instance. First, we prove that if Algorithm uses more than N colors, then competitive ratio is greater than 3. In this case Presenter stops the game just after some first subphase and the presented graph consists of two disjoint cliques Q_1 and Q_2. All intervals in Q_1 have bandwidths $\frac{1}{N}$ and there are at most $K_1 + K_2 = \frac{1}{3}N^2$ of them. Thus, one can easily color Q_1 with at most $\frac{1}{3}N$ colors. Moreover, since $w_1 + w_2 = \frac{17}{52} + 2\epsilon < \frac{1}{3}$, we can color all intervals in Q_2 with $\frac{1}{3}N$ colors. Thus, in this case Algorithm uses at least 3 times more colors than one needs to color $Q_1 \cup Q_2$, and we are done.

In order to analyze all possible games between Presenter and Algorithm in the case when Algorithm uses at most N colors, we introduce some notation. As before we define variables $X_{a,b}$, but in this case we consider only those tuples (a, b) for which we have $aw_1 + bw_2 \leqslant 1$. Let $L_{2,1}$ be the number of colors used by Algorithm in the second phase to color intervals from Q_1 that were already used in the first phase to color some intervals from Q_2. Let $L_{i,0}$ the number of intervals from Q_1 that were marked in the i-th phase. Analogously, for $1 \leqslant j \leqslant i \leqslant 2$ let $R_{i,j}$ be the number of colors used by Algorithm in the i-th phase to color intervals from Q_2 that were already used in the j-th phase to color some intervals from Q_1. Similarly, $R_{i,0}$ denotes the number of intervals from Q_2 that were marked in the i-th phase.

We also define several grouping variables, i.e. $L_i = \sum_j L_{i,j}$ denotes the total number of colors used by Algorithm in the i-th phase to color intervals from Q_1 that were not used to color intervals from Q_1 in the previous phases. Analogously, $R_i = \sum_j R_{i,j}$. We denote Z_i the total number of color used by Algorithm in the first i phases, so that $Z_i = \sum_{j \leqslant i} (L_{j,0} + R_{j,0})$.

Clearly, our strategy for Presenter implies several constraints on the introduced variables. In the first phase Presenter shows $K_1 = \frac{1}{12}N^2$ intervals on the left side, and

all those intervals have bandwidth $\frac{1}{N}$. Thus, Algorithm has to use at least $\frac{K_1}{N} = \frac{1}{12}N$ colors to color them, and so we have a constraint $L_1 \geqslant \frac{1}{12}N$. Analogously, after second phase Q_1 consists of $K_1 + K_2 = \frac{1}{3}N^2$ intervals, so $L_1 + L_2 \geqslant \frac{1}{3}N$. Moreover, it is obvious that $L_{2,1} \leqslant R_{1,0}$, and $\sum_{i=j}^{2} R_{i,j} \leqslant L_{j,0}$ since Algorithm cannot use more colors than one set contains.

Analogously to Section 2.3.1, our next lemmas and corollaries show how many additional colors Presenter can force on Algorithm using the strategy $1+$.

Lemma 2.17. *The strategy $1+$ is applicable in the first phase with $H = \frac{1}{12}N$ and $h = \left\lceil \frac{1}{12}R_{1,0} \right\rceil$.*

Proof. Consider the presented set of intervals after the first subphase of the first phase. Q_2 contains N intervals, each of bandwidth $\frac{1}{13} + \epsilon$, and exactly $R_{1,0}$ are marked. Hence, we can easily color Q_2 with $\frac{1}{12}N$ colors in such a way that $\left\lceil \frac{1}{12}R_{1,0} \right\rceil$ colors are used to color all marked intervals in Q_2. Note that Q_1 contains exactly $K_1 = \frac{1}{12}N^2$ intervals and all of them have bandwidth $\frac{1}{N}$. Thus, we can also color Q_1 using exactly $\frac{1}{12}N$ colors. □

Corollary 2.18. *If Presenter decides to play the strategy $1+$ in the first phase, then the achieved competitive ratio is at least $1 + 12\frac{Z_1}{N} - \frac{R_{1,0}}{N} - \frac{24}{N}$.*

Lemma 2.19. *If $R_{1,0} \leqslant R_{2,0}$, then the strategy $1+$ is applicable in the second phase with $H = \frac{1}{3}N$ and $h = \left\lceil \frac{1}{3}R_{2,0} \right\rceil$.*

Proof. Consider the presented set of intervals after the first subphase of the second phase. Clearly, all $\frac{1}{3}N^2$ intervals in Q_1 can be colored using exactly $\frac{1}{3}N$ colors. Since $\frac{1}{13} + \epsilon + \frac{1}{4} + \epsilon = \frac{17}{52} + 2\epsilon < \frac{1}{3}$, we can color 3 light intervals and 3 heavy ones with the same color. We label light intervals $I^l_1, \ldots, I^l_N$ in such a way that marked intervals come first. Analogously, we label heavy intervals $I^h_1, \ldots, I^h_N$. For every $i \in \left[\frac{1}{3}N\right]$ we color six intervals I^l_{3i-2}, I^l_{3i-1}, I^l_{3i}, I^h_{3i-2}, I^h_{3i-1}, I^h_{3i} with a color i. We just properly colored all intervals in Q_2 using $H = \frac{1}{3}N$ colors, and it is easy to see that all marked intervals are colored with colors from $[h]$ where $h = \left\lceil \frac{1}{3}R_{2,0} \right\rceil$. □

Corollary 2.20. *If $R_{1,0} \leqslant R_{2,0}$ and Presenter decides to play the strategy $1+$ in the second phase, then the achieved competitive ratio is at least $1 + 3\frac{Z_2}{N} - \frac{R_{2,0}}{N} - \frac{6}{N}$.*

Lemmas 2.17 and 2.19 provide colorings that use small number of colors for marked intervals and also use the optimal number of colors on the whole graph. In the next lemma we show a coloring that uses small number of colors on marked intervals, but on the whole graph requires more colors than it is needed. Unfortunately, this makes the optimum a function of $R_{1,0}$ and $R_{2,0}$ and eventually leads to a non-linear constraint.

Lemma 2.21. *If $R_{1,0} \geqslant R_{2,0}$, then the strategy $1+$ is applicable in the second phase with $H = \frac{1}{3}N + \frac{1}{12}R_{1,0} - \frac{1}{12}R_{2,0} + 3$ and $h = \frac{1}{12}R_{1,0} + \frac{1}{4}R_{2,0} + 2$.*

Proof. Consider the presented set of intervals after the first subphase of the second phase. Clearly, we can color all heavy marked intervals and $R_{2,0}$ marked light intervals using $\left\lceil \frac{1}{3}R_{2,0} \right\rceil$ colors. We color the remaining light marked intervals using additional $\left\lceil \frac{1}{12}(R_{1,0} - R_{2,0}) \right\rceil$ colors. Hence, we colored all marked intervals using $\left\lceil \frac{1}{3}R_{2,0} \right\rceil + \left\lceil \frac{1}{12}(R_{1,0} - R_{2,0}) \right\rceil \leqslant \frac{1}{12}R_{1,0} + \frac{1}{4}R_{2,0} + 2$ colors. At this point there are $N - R_{1,0}$ not colored light intervals and $N - R_{2,0}$ not colored heavy ones. We color them using $\left\lceil \frac{1}{3}(N - R_{2,0}) \right\rceil$ colors. Thus, we used at most $\frac{1}{3}N + \frac{1}{12}R_{1,0} - \frac{1}{12}R_{2,0} + 3$ colors to color all intervals from Q_2. □

Corollary 2.22. *If $R_{1,0} \geqslant R_{2,0}$ and Presenter decides to play the strategy $1+$ in the second phase, then the achieved competitive ratio is at least $1 + \frac{Z_2 - \frac{1}{12}R_{1,0} - \frac{1}{4}R_{2,0} - 3}{\frac{1}{3}N + \frac{1}{12}R_{1,0} - \frac{1}{12}R_{2,0} + 3}$.*

Let us rearrange the bound we get from the above corollary.

$$C_N \geqslant 1 + \frac{Z_2 - \frac{1}{12}R_{1,0} - \frac{1}{4}R_{2,0} - 3}{\frac{1}{3}N + \frac{1}{12}R_{1,0} - \frac{1}{12}R_{2,0} + 3} = 1 + \frac{12Z_2 - R_{1,0} - 3R_{2,0} - 36}{4N + R_{1,0} - R_{2,0} + 36}$$

$$(C_N - 1)(4N + R_{1,0} - R_{2,0} + 36) \geqslant 12Z_2 - R_{1,0} - 3R_{2,0} - 36$$

$$4(C_N - 1)N + 36C_N \geqslant 12Z_2 - C_N R_{1,0} + (C_N - 4)R_{2,0}$$

Note that if C_N is a constant, then this inequality is a linear constraint in terms of $Z_2, R_{1,0}$ and $R_{2,0}$.

$X_{a,b}, L_i, R_i, L_{i,j}, R_{i,j} \in \mathbb{N}$	
$\sum_{ab} aX_{a,b} = N$	*#intervals in Q_2 in 1st phase*
$\sum_{ab} bX_{a,b} = N$	*#intervals in Q_2 in 2nd phase*
$\sum_{a>0,b} X_{a,b} = R_1$	*#new colors in Q_2 in 1st phase*
$\sum_{b>0} X_{0,b} = R_2$	*#new colors in Q_2 in 2nd phase*
$\forall_i \sum_j L_{i,j} = L_i$	*#new colors in Q_1 in i-th phase*
$\forall_i \sum_j R_{i,j} = R_i$	*#new colors in Q_2 in i-th phase*
$\forall_i \sum_{j=1}^{i} L_{j,0} + R_{j,0} = Z_i$	*#different colors after i-th phase*
$\forall_j \sum_{i=j+1}^{2} L_{i,j} \leqslant R_{j,0}$	
$\forall_j \sum_{i=j}^{2} R_{i,j} \leqslant L_{j,0}$	
$L_1 \geqslant \frac{1}{12}N$	
$L_1 + L_2 \geqslant \frac{1}{3}N$	
$C_N \geqslant 1 + \frac{12}{N}Z_1 - \frac{1}{N}R_{1,0} - \frac{24}{N}$	*ratio after 1st phase*
$C_N \geqslant 1 + \frac{3}{N}Z_2 - \frac{1}{N}R_{2,0} - \frac{6}{N}$	*2nd phase ratio under $R_{1,0} \leqslant R_{2,0}$*
$4(C_N - 1) + \frac{36C_N}{N} \geqslant \frac{12}{N}Z_2 - \frac{C_N}{N}R_{1,0} + \frac{C_N - 4}{N}R_{2,0}$	*2nd phase ratio under $R_{1,0} \geqslant R_{2,0}$*

Table 2.3: The set of constraints describing the game between Algorithm and Presenter for a given N.

We just presented how to encode all possible games between Presenter and Algorithm in the proposed schema, see Table 2.3 for a quick summary. As in the online unit interval graph coloring, see Section 2.3.1, we do not deal with the original set of constraints, but instead we introduce new variables $x_{a,b} = \frac{1}{N}X_{a,b}$, $r_{i,j} = \frac{1}{N}R_{i,j}$ etc., and rearrange inequalities to get an approximation presented in Table 2.4. Observe that constraints listed in Table 2.4 do not represent a simplex. This is because in the last inequality there is a combined term $\frac{1}{N}C_N r_{1,0}$ and both C_N and $r_{1,0}$ are variables.

We solve the first branch ($r_{1,0} \leqslant r_{2,0}$, without the last constraint), treating it as a regular linear program and we get the optimal solution $C_N = 2.1571$.

To solve the branch $r_{1,0} \geqslant r_{2,0}$ without the last second constraint, we binary search the value of C_N by testing the emptiness of the simplex defined by the set of constraints

presented in Table 2.4 under the assumption that C_N is a constant. We are binary searching between 1 and 3, since competitive ratio cannot be less than 1 and if it is greater than 3, then we are done. Using this method we get the result $C_N = 2.211 \pm 0.001$.

Thus, we obtained a lower bound of 2.1571 on the asymptotic competitive ratio in the online unit interval coloring.

Theorem 2.23. *Both absolute and asymptotic competitive ratios in the online unit interval coloring are at least* 2.1571.

$x_{a,b}, l_i, r_i, l_{i,j}, r_{i,j} \geqslant 0$	
$\sum_{ab} a x_{a,b} = 1$	
$\sum_{ab} b x_{a,b} = 1$	
$\sum_{a>0,b} x_{a,b} = r_1$	
$\sum_{b>0} x_{0,b} = r_2$	
$\forall_i \sum_j l_{i,j} = l_i$	
$\forall_i \sum_j r_{i,j} = r_i$	
$\forall_i \sum_{j=1}^{i} l_{j,0} + r_{j,0} = z_i$	
$\forall_j \sum_{i=j+1}^{2} l_{i,j} \leqslant r_{j,0}$	
$\forall_j \sum_{i=j}^{2} r_{i,j} \leqslant l_{j,0}$	
$l_1 \geqslant \frac{1}{12}$	
$l_1 + l_2 \geqslant \frac{1}{3}$	
$C_N - 1 \geqslant 12 z_1 - r_{1,0}$	
$C_N - 1 \geqslant 3 z_2 - r_{2,0}$	*under* $r_{1,0} \leqslant r_{2,0}$
$4(C_N - 1) \geqslant 12 z_2 - C_N r_{1,0} + (C_N - 4) r_{2,0}$	*under* $r_{1,0} \geqslant r_{2,0}$

Table 2.4: The set of constraints describing an approximation on the game between Algorithm and Presenter.

3

Online Interval Coloring of Short Intervals

3.1 Introduction

The competitive ratio for the online interval coloring was established by Kierstead and Trotter [26]. They constructed a strategy for Algorithm that uses at most $3\omega-2$ colors on ω-colorable set of intervals. They also presented a matching lower bound – a strategy for Presenter that forces Algorithm to use at least $3\omega - 2$ colors. The unit variant of the online interval coloring was studied by Epstein and Levy [12]. They presented a strategy for Presenter that forces Algorithm to use at least $\left\lfloor \frac{3\omega}{2} \right\rfloor$ colors. Moreover, they noticed that a natural greedy algorithm uses at most $2\omega - 1$ colors.

A natural question arises, what happens in between the interval and unit interval graph classes. In this chapter we ask about the optimal competitive ratio of online coloring algorithms for intersection graphs of intervals with lengths restricted to a fixed range, say $[1, \sigma]$. In particular, we give lower and upper bounds on the asymptotic competitive ratio in this setting as functions of σ.

It seems natural to ask if it is possible to improve the bound of $3\omega - 2$ by assuming that interval lengths belong to a fixed range. The study of interval graphs with bounded length representations was initiated by Fishburn and Graham [14]. However, the work done here focused mainly on the combinatorial structure, and not on its algorithmic applications. The lengths of presented intervals in the Kierstead and Trotter strategy increase with the increasing ω. For this reason, with the interval lengths restricted to $[1, \sigma]$, their lower bound is only for the absolute competitive ratio and does not exclude, say, an algorithm that always uses at most $2\omega+\sigma^{10}$ colors. In Theorem 3.20 we rule out the existence of such an algorithm.

3.1.1 Our results

Algorithm.

Our algorithm is inspired by the recent result for online coloring of unit disk intersection graphs [22]. We cover the real line with overlapping blocks, grouped into a constant number of classes. Each class gets a private set of available colors. When an interval is presented, the algorithm chooses a block in a round-robin fashion, and greedily assigns a color from its class.

Lower Bounds.

In order to prove lower bounds we present a series of strategies for Presenter with the following consequences:

1. For every $\sigma > 1$ there is no online algorithm for σ-interval coloring with the asymptotic competitive ratio less than $5/3$.
2. For every $\sigma > 2$ there is no online algorithm for σ-interval coloring with the asymptotic competitive ratio less than $7/4$.
3. For every $\epsilon > 0$ there is a large enough $\sigma \geqslant 1$ such that there is no online algorithm for σ-interval coloring with the asymptotic competitive ratio $5/2 - \epsilon$.

The following, more illustrative, statement is a direct corollary of the last result.

Corollary 3.1. *There is no online algorithm that for every σ, ω and f uses at most $2.499 \cdot \omega + f(\sigma)$ colors for ω-colorable graphs in σ-interval coloring.*

All our lower bounds can be considered as generalizations of the *EL-separation* strategy presented in Chapter 2. The novel technique is the recursive composition of strategies, materialized in Lemmas 3.7, 3.10, 3.13, and 3.18. Our $5/2$ lower bound borrows also from the work of Kierstead and Trotter [26]. However, in order to control the length of intervals independently of the number of colors, we cannot simply use the pigeonhole principle, as they did. Instead, we develop Lemmas 3.16 and 3.17, which let us overcome this issue, at a cost of a worse bound for the competitive ratio, i.e. $5/2$ instead of 3.

3.2 Algorithm

In this section we present our $(1 + \sigma)$-competitive algorithm for the online σ-interval coloring.

Theorem 3.2. *For every $\sigma \in \mathbb{Q}$, $\sigma \geqslant 1$, there is an algorithm for online σ-interval coloring with $1 + \sigma$ asymptotic competitive ratio.*

Proof. Let us present an algorithm which, in principle, works for any real σ, however only for a rational σ it achieves the declared competitive ratio. The algorithm has a positive integer parameter b. Increasing the parameter brings the asymptotic competitive ratio closer to $1 + \sigma$ at the cost of increasing the additive constant. More precisely, given an ω-colorable set of intervals our algorithm colors it using at most $\lceil b \cdot (1 + \sigma) \rceil \cdot \left(\frac{\omega}{b} + b - 1\right)$ colors, and thus its competitive ratio is $\frac{\lceil b \cdot (1+\sigma) \rceil}{b} + O(1/\omega)$. For a rational σ, in order to obtain exactly the declared $1 + \sigma$ asymptotic competitive ratio it is sufficient to set b to the smallest possible denominator of a simple fraction representation of σ. Let $\varphi = \lceil b \cdot (1 + \sigma) \rceil$. The algorithm will use colors from the set $\{0, 1, \ldots, \varphi - 1\} \times \mathbb{N}$.

Now, let us consider the partition of the real line into *small blocks*. For $i \in \mathbb{Z}$, the i-th small block occupies the interval $[i \cdot \frac{1}{b}, (i + 1) \cdot \frac{1}{b})$. Moreover, we define *large blocks*. The i-th large block occupies the interval $[i \cdot \frac{1}{b}, i \cdot \frac{1}{b} + 1)$. See Figure 3.1.

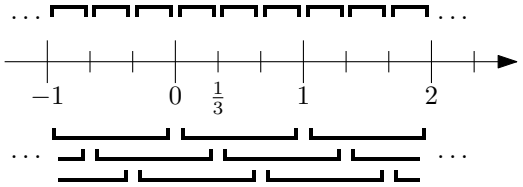

Figure 3.1: Small blocks (up), and large blocks (down), for $b = 3$

Let us point out certain properties of the blocks, which will be useful in the further analysis. Each large block is the sum of b consecutive small blocks, and each small block is a subset of b consecutive large blocks. Further, length of a large block is 1, and for any two intervals of length in $[1, \sigma]$ that both have the left endpoint in the same large block, the two intervals intersect. Thus, the intervals whose left endpoints belong to a fixed large block form a clique. Finally, if the indices of two large blocks differ by at least φ, then any two intervals – one with the left endpoint in one block, the other with the left endpoint in the other – do not intersect.

With each small block the algorithm associates a *small counter*, and with each large block the algorithm associates a *large counter*. Let S_i denote the small counter of the i-th small block, and L_i denote the large counter of the i-th large block. Initially, all the small and large counters are set to zero.

To assign a color to an interval, the algorithm proceeds as follows:

1. Let i be the index of the small block containing the left endpoint of the interval.
2. Let j be the index of the large block containing the left endpoint of the interval such that $j \equiv S_i \pmod{b}$. Note that there is exactly one such j.
3. Assign to the interval the color $(j \bmod \varphi, L_j)$.
4. Increase the small counter S_i by one.
5. Increase the large counter L_j by one.

First let us argue that the algorithm outputs a proper coloring. Consider any two intervals which were assigned the same color. Let j_1 and j_2 denote the indices of the large blocks selected for these intervals by the algorithm. Since the colors of the two intervals have the same first coordinates, we have that $j_1 \equiv j_2 \pmod{\varphi}$. However, since the second coordinates, which are determined by large counters, are also the same, j_1 and j_2 must be different, and thus they differ by at least φ. Hence the left endpoints of the large blocks j_1 and j_2 are at least $1 + \sigma$ apart, and the two considered intervals do not intersect, thus the coloring is proper.

It remains to bound the number of colors in terms of the clique number ω. Let j be the index of the maximum large counter L_j. Clearly, the algorithm used at most $\varphi \cdot L_j$ colors in total. Let C denote the set of intervals with the left endpoints in the j-th large block and colored with a color in $\{j \bmod \varphi\} \times \mathbb{N}$. Observe that $|C| = L_j$. Let x_k denote the number of intervals in C which have the left endpoint in the k-th small block. Recall that the j-th large block is the sum of b small blocks – indexed $j, j+1, \ldots,$

$j + b - 1$ – and thus $L_j = x_j + x_{j+1} + \cdots + x_{j+b-1}$. Now, observe that, because of the round robin selection in the step 2 of the algorithm,

$$S_k \geqslant b \cdot (x_k - 1) + 1.$$

Let D denote the set of all intervals with the left endpoints in the j-th large block. We can bound the number of intervals in D

$$|D| \;=\; \sum_{k=j}^{j+b-1} S_k \;\geqslant\; \sum_{k=j}^{j+b-1} (b \cdot (x_k - 1) + 1) \;=\; b \cdot (L_j - b) + b.$$

Recall that D is a clique and thus the clique number ω of the input graph is at least the size of D. Therefore $L_j \leqslant \frac{\omega + b \cdot (b-1)}{b}$, and the algorithm used at most

$$\lceil b \cdot (1 + \sigma) \rceil \cdot \left(\frac{\omega}{b} + b - 1 \right)$$

colors. □

Note that for $\sigma' > \sigma$ every σ'-interval coloring algorithm is also a correct σ-interval coloring algorithm, with the same upper bound on its competitive ratio. Therefore, for $\sigma \in \mathbb{R} \setminus \mathbb{Q}$ Theorem 3.2 yields an online σ-interval coloring algorithm with a competitive ratio arbitrarily close to $1 + \sigma$. This distinction between rational and irrational values of σ becomes somewhat less peculiar in the light of the results of Fishburn and Graham [14], who proved, among other things, that the classes of graphs with interval representation with lengths in $[1, \sigma]$ are right-continuous exactly at irrational σ.

Until now, the state-of-the art was the 2-competitive FirstFit algorithm [13] for $\sigma = 1$ and the 3-competitive Kierstead-Trotter algorithm [26] for $\sigma > 1$. Thus, our algorithm matches the performance of FirstFit for $\sigma = 1$, and beats the Kierstead-Trotter algorithm up until $\sigma = 2$.

3.3 Lower Bounds

In this section we show several families of strategies for Presenter that force Algorithm to use many colors while the introduced set of intervals is colorable with a smaller number of colors, and contains only short intervals. We start with a short, informal presentation of these strategies. First, let us recall the *EL-strategy*:

1. Presenter plays a clique of $\frac{\omega}{2}$ *initial* intervals. Algorithm has to use $\frac{\omega}{2}$ different colors, let $\mathcal{X}$ denote the set of these colors.

2. To the left of the initial intervals, Presenter plays a clique of ω *separation* intervals so that all intervals colored with colors in $\mathcal{X}$ are slightly shifted to the left of all intervals colored with colors not in $\mathcal{X}$.

3. Presenter plays a clique of $\frac{\omega}{2}$ *final* intervals that intersect all the initial intervals, and $\frac{\omega}{2}$ right-most separation intervals.

In Section 3.3.1 we generalize this strategy. We observe, that instead of presenting a clique in the first step, Presenter can use an arbitrary strategy that requires slightly shorter intervals. For σ-interval coloring we can construct a recursive strategy that applies this trick roughly σ times. Using this simple observations we obtain a family of strategies for different σ. Corollary 3.8 gives that, for example, there is no online algorithm with $(1.6 - \epsilon)$ asymptotic competitive ratio for $(2 + \epsilon)$-interval coloring (for any $\epsilon > 0$).

Theorem (Reminder of Theorem 3.12)**.** *For every $\sigma > 1$ there is no online algorithm for σ-interval coloring with the asymptotic competitive ratio less than $5/3$.*

Now, consider the following strategy for Presenter in online $(1+\epsilon)$-interval coloring (see the proof of Lemma 3.10 for all the details, Figure 3.3 may help visualize this strategy).

1. Presenter plays a clique of $\frac{\omega}{3}$ *initial* intervals of length 1. Algorithm has to use $\frac{\omega}{3}$ different colors, let $\mathcal{X}$ denote the set of these colors.

2. To the left of the initial intervals, Presenter plays a clique of ω *left separation* intervals of length 1 so that all intervals colored with colors in $\mathcal{X}$ are slightly shifted to the left of all intervals colored with colors not in $\mathcal{X}$. Let $\mathcal{Y}$ denote the set of colors of $\frac{\omega}{3}$ right-most left separation intervals.

3. To the right of initial intervals, Presenter plays a clique of ω *right separation* intervals of length 1 so that all intervals colored with colors in $\mathcal{X} \cup \mathcal{Y}$ are slightly shifted to the right of all intervals colored with colors not in $\mathcal{X} \cup \mathcal{Y}$.

4. Presenter plays a clique of $\frac{2\omega}{3}$ *final* intervals of length $1 + \epsilon$ that intersect all the initial intervals, $\frac{\omega}{3}$ right-most left separation intervals, and $\frac{\omega}{3}$ left-most right separation intervals.

We get that there is no online algorithm with $(\frac{5}{3} - \epsilon_1)$ asymptotic competitive ratio for $(1 + \epsilon_2)$-interval coloring (for any $\epsilon_1, \epsilon_2 > 0$), i.e. we've sketched an informal proof of Theorem 3.12. In Section 3.3.2 we use the above strategy for $(1 + \epsilon)$-intervals as a recursive step that can be used to obtain strategies for larger σ's. We get another family of strategies, where for σ-interval coloring we can apply the recursive step roughly $\frac{\sigma}{2}$ times. Corollary 3.11 gives that, for example, there is no online algorithm with 1.7 asymptotic competitive ratio for $(3 + \epsilon)$-interval coloring (for any $\epsilon > 0$).

Theorem (Reminder of Theorem 3.15)**.** *For every $\sigma > 2$ there is no online algorithm for σ-interval coloring with the asymptotic competitive ratio less than $7/4$.*

Now, consider the following strategy for Presenter in online $(2+\epsilon)$-interval coloring (see the proof of Lemma 3.13 for all the details, Figures 3.4 and 3.5 may help visualize this strategy).

1. Presenter plays a clique of $\frac{\omega}{2}$ *left initial* intervals of length 1. Algorithm has to use $\frac{\omega}{2}$ different colors, let $\mathcal{X}_1$ denote the set of these colors.

2. To the right of the left initial intervals, Presenter plays a clique of $\frac{\omega}{2}$ *right initial* intervals of length 1. Algorithm has to use $\frac{\omega}{2}$ different colors, let $\mathcal{X}_2$ denote the set of these colors.

3. To the left of the left initial intervals, Presenter plays a clique of ω *left separation* intervals of length 1 so that all intervals colored with colors in $\mathcal{X}_1$ are slightly shifted to the left of all intervals colored with colors not in $\mathcal{X}_1$. Let $\mathcal{Y}_1$ denote the set of colors of $\frac{\omega}{2}$ right-most left separation intervals.

4. To the right of the right initial intervals, Presenter plays a clique of ω *right separation* intervals of length 1 so that all intervals colored with colors in $\mathcal{X}_2$ are slightly shifted to the right of all intervals colored with colors not in $\mathcal{X}_2$. Let $\mathcal{Y}_2$ denote the set of colors of $\frac{\omega}{2}$ left-most right separation intervals.

5. Let $\mathcal{C}_1 = \mathcal{X}_1 \cup \mathcal{Y}_1$, and $\mathcal{C}_2 = \mathcal{X}_2 \cup \mathcal{Y}_2$.

   (a) If $|\mathcal{C}_1 \cup \mathcal{C}_2| \geqslant \frac{5\omega}{4}$, Presenter plays a clique of $\frac{\omega}{2}$ *final* intervals of length $2+\epsilon$ that intersect all the initial intervals, $\frac{\omega}{2}$ right-most left separation intervals, and $\frac{\omega}{2}$ left-most right-separation intervals. In total, Algorithm has to use at least $\frac{7\omega}{4}$ colors in this case.

   (b) If $|\mathcal{C}_1 \cup \mathcal{C}_2| < \frac{5\omega}{4}$ (which implies $|\mathcal{C}_1 \cap \mathcal{C}_2| > \frac{3\omega}{4}$), Presenter plays a clique of $\frac{\omega}{2}$ *pre-final* intervals of length $1+\epsilon$ that intersect all the right initial intervals, and $\frac{\omega}{2}$ left-most right separation intervals. Then, Presenter plays a clique of $\frac{\omega}{2}$ *final* intervals of length $1+\epsilon$ that intersect all the left initial intervals, $\frac{\omega}{2}$ right-most left separation intervals, and all the pre-final intervals. The sets of colors of final and pre-final intervals are disjoint, and moreover do not intersect with $\mathcal{C}_1 \cap \mathcal{C}_2$. A short calculation shows that in this case Algorithm also has to use at least $\frac{7\omega}{4}$ colors.

Thus, we get that there is no online algorithm with $(\frac{7}{4}-\epsilon_1)$ asymptotic competitive ratio for $(2+\epsilon_2)$-interval coloring (for any $\epsilon_1, \epsilon_2 > 0$), i.e. we've sketched an informal proof of Theorem 3.15. In Section 3.3.3 we use the above strategy for $(2+\epsilon)$-intervals as a recursive step that can be used to obtain strategies for larger σ's. We get another family of strategies, where for σ-interval coloring we can apply the recursive step roughly $\log \sigma$ times. Corollary 3.14 gives that, for example, there is no online algorithm with 1.8 asymptotic competitive ratio for $(8+\epsilon)$-interval coloring (for any $\epsilon > 0$). To get a lower bound better than 2 we combine our method with some ideas from the lower bound by Kierstead and Trotter [26]. In Section 3.3.4 we prove Corollary 3.19, which gives that for any $\epsilon > 0$:

1. there is no 2-competitive online algorithm for $(4^{39}+\epsilon)$-interval coloring,

2. there is no 2.4-competitive online algorithm for $(4^{486}+\epsilon)$-interval coloring, and

3. there is no 2.49-competitive online algorithm for $(4^{7970}+\epsilon)$-interval coloring.

Table 3.1 summarizes the aforementioned strategies, and illustrates the growth of the interval length σ required to prove better and better lower bounds. To properly capture asymptotic properties of the introduced strategies we give the following formal definitions.

ratio	interval length	strategy
1.5	1	Epstein and Levy [13]
1.6	$2+\epsilon$	Corollary 3.8, $n=2$ iterations
1.66	$1+\epsilon$	Corollary 3.11, $n=1$ iteration
1.72	$3+\epsilon$	Corollary 3.11, $n=2$ iterations
1.75	$2+\epsilon$	Corollary 3.14, $n=1$ iteration
1.81	$8+\epsilon$	Corollary 3.14, $n=2$ iterations
2	$4^{39}+\epsilon$	Corollary 3.19, $n=3$ iterations, $\gamma=0.21030395$
2.4	$4^{486}+\epsilon$	Corollary 3.19, $n=6$ iterations, $\gamma=0.0339$
2.49	$4^{7970}+\epsilon$	Corollary 3.19, $n=10$ iterations, $\gamma=0.003449$

Table 3.1: Summary of selected strategies for Presenter

Definition 3.3. *For $\omega, C \in \mathbb{N}_+$ and $\sigma, M \in \mathbb{R}_+$, an* $\langle \omega, C, \sigma, M\rangle$-strategy *is a strategy for Presenter that forces Algorithm to use at least C colors subject to the following constraints:*

1. *the set of introduced intervals is ω-colorable,*
2. *every introduced interval has length at least 1 and at most σ,*
3. *every introduced interval is contained in the interval $[0, M]$.*

We are interested in providing strategies that achieve the biggest possible ratio $\frac{C}{\omega}$ for large ω. This motivates the following definition.

Definition 3.4. *An* $\langle \alpha, \sigma, M\rangle$-schema *is a set of $\langle \omega, C_\omega, \sigma, M\rangle$-strategies for all $\omega \in \mathbb{N}_+$ such that $C_\omega = \alpha\omega - o(\omega)$.*

The $o(\omega)$ term in the above definition accounts for the fact that sometimes in a proof we would like to introduce, say, $\frac{\omega}{2}$-clique. Then, for odd ω's a rounding is required, which results in small inaccuracies we need to control.

Remark 3.5. *Note that the existence of an* $\langle \alpha, \sigma, M\rangle$-schema *implies a lower bound of α for the asymptotic competitive ratio of any online algorithm solving the σ-interval coloring problem.*

To put the above definitions in context, note that Kierstead and Trotter [26] give, for all $\omega \in \mathbb{N}_+$, an $\langle \omega, 3\omega-2, f(\omega), f(\omega)\rangle$-strategy. However, their family of strategies does not yield an $\langle \alpha, \sigma, M\rangle$-schema, because the length of the presented intervals grows with ω.

Example 3.6 ($\langle 1,1,1\rangle$-schema)**.** *For any $\omega \in \mathbb{N}_+$, a strategy that introduces the interval $[0,1]$ in every round $1,\ldots,\omega$ is an $\langle \omega, \omega, 1, 1\rangle$-strategy. The set of these strategies is a $\langle 1,1,1\rangle$-schema.*

In this section we show a series of constructions that use an existing schema to create another schema with different parameters. The $\langle 1,1,1\rangle$-schema given above is the initial step for those constructions.

Let S be an $\langle \omega, C, \sigma, M\rangle$-strategy. We say that *Presenter uses strategy S in the interval* $[x, x+M]$ to denote that Presenter plays according to S, presenting intervals shifted by x, until Algorithm uses C colors.

3.3.1 Warm-up

Our first construction is a natural generalization of the strategy for unit intervals given by Epstein and Levy [13]. It is surpassed by more involved strategies coming later, but it serves as a gentle introduction to our framework.

Lemma 3.7. *If there is an $\langle \alpha, \sigma, M \rangle$-schema, then there is a $\left\langle 2 - \frac{1}{\alpha+1}, M + \epsilon, M + 1 + \epsilon \right\rangle$-schema for every $\epsilon > 0$.*

Proof. To prove the lemma we need to provide an $\left\langle \omega, (2 - \frac{1}{\alpha+1})\omega - o(\omega), M + \epsilon, M + 1 + \epsilon \right\rangle$-strategy for every $\omega \in \mathbb{N}_+$. Let us fix an $\omega \in \mathbb{N}_+$, and let $\omega' = \left\lfloor \frac{\omega}{\alpha+1} \right\rfloor$. The $\langle \alpha, \sigma, M \rangle$-schema contains an $\langle \omega', \alpha\omega' - \delta, \sigma, M \rangle$-strategy S for some $\delta = o(\omega')$. Similarly to the *EL-separation*, the strategy for Presenter consists of three phases, see Figure 3.2. In the first phase, called the *initial phase*, Presenter uses strategy S inside the interval $[1 + \epsilon, M + 1 + \epsilon]$. Let $C = \alpha\omega' - \delta$ and let $\mathcal{X}$ denote the set of C colors used by Algorithm in the initial phase.

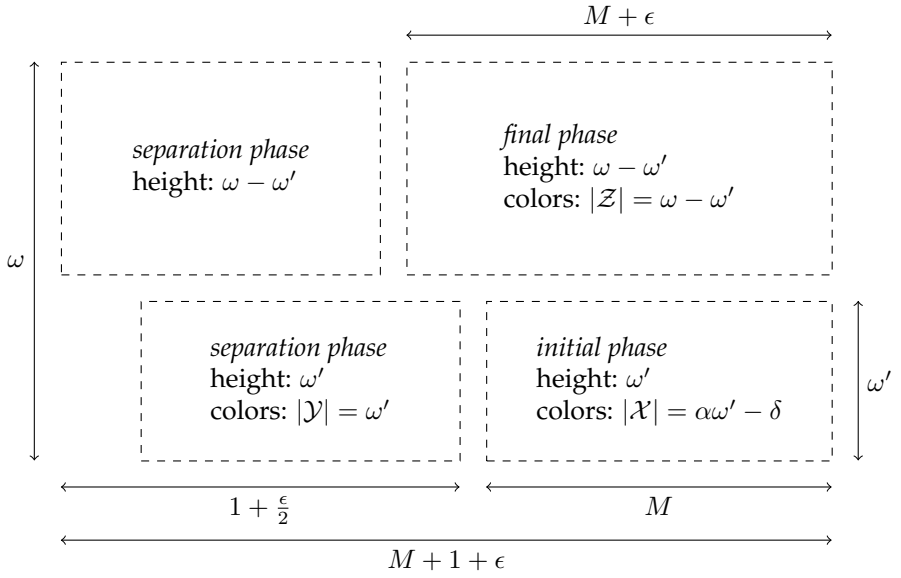

Figure 3.2: Strategy construction in Lemma 3.7

The second phase, borrowed from [13, 4], is called the *separation phase*. In this phase, Presenter plays the following separation strategy for ω rounds. Let $l_1 = 0$ and $r_1 = \frac{\epsilon}{2}$. In the i-th round of the separation phase Presenter introduces the interval $[\frac{l_i+r_i}{2}, \frac{l_i+r_i}{2} + 1]$. If Algorithm colors the interval with one of the colors in $\mathcal{X}$, let $l_{i+1} = \frac{l_i+r_i}{2}$ and $r_{i+1} = r_i$, which means that the next interval will be shifted slightly to the right. Otherwise, let $l_{i+1} = l_i$ and $r_{i+1} = \frac{l_i+r_i}{2}$, which means that the next interval will be shifted slightly to the left. Observe that all intervals introduced in the separation phase have length 1 and $\forall_i \frac{l_i+r_i}{2} < \frac{\epsilon}{2}$. Thus, every interval introduced in the separation phase is contained in $\left[0, 1 + \frac{\epsilon}{2}\right]$ and any two of those intervals intersect. Furthermore, the choice of l_i's and r_i's guarantees that for any two intervals x, y introduced in the separation phase, x colored with a color in $\mathcal{X}$, and y colored with a color not in $\mathcal{X}$, we have that the left endpoint of x is to the left of the left endpoint of y. Let Y be the set of ω' rightmost intervals introduced in the separation phase. Let $\mathcal{Y}$ be the set of colors used by

Algorithm on the intervals in Y. As $C + \omega' < \omega$, we get that sets of colors $\mathcal{X}$ and $\mathcal{Y}$ are disjoint.

For the last phase, called the *final phase*, let r be the left-most right endpoint of an interval in Y. In the final phase Presenter introduces $\omega - \omega'$ times the same interval $[r, M + 1 + \epsilon]$. This interval intersects all intervals introduced in the initial phase, all intervals in Y, and no other interval introduced in the separation phase. Thus, Algorithm must use $\omega - \omega'$ colors in the final phase that are different from the colors in both $\mathcal{X}$ and $\mathcal{Y}$. Let $\mathcal{Z}$ denote the set of colors used by Algorithm in the final phase.

The presented set of intervals is clearly ω-colorable and Algorithm used at least $|\mathcal{X}| + |\mathcal{Y}| + |\mathcal{Z}| = \alpha\omega' - \delta + \omega' + \omega - \omega' = \left(2 - \frac{1}{\alpha+1}\right)\omega - o(\omega)$ many colors. The longest interval presented has length $M + \epsilon$, and all intervals are contained in $[0, M + 1 + \epsilon]$. Thus, we have constructed a $\left\langle 2 - \frac{1}{\alpha+1}, M + \epsilon, M + 1 + \epsilon \right\rangle$-schema. □

Corollary 3.8. *There is an* $\left\langle \frac{F_{2n+1}}{F_{2n}}, n + \epsilon, n + 1 + \epsilon \right\rangle$*-schema, for every* $n \in \mathbb{N}_+$ *and every* $\epsilon > 0$*, where* F_n *is the* n*-th Fibonacci number (*$F_0 = F_1 = 1$*,* $F_{n+2} = F_{n+1} + F_n$*).*

Proof. Starting with a $\langle 1, 1, 1\rangle$-schema and repeatedly applying Lemma 3.7 one can generate[1] a family of $\langle \alpha_n, \sigma_n + \epsilon_n, M_n + \epsilon_n\rangle$-schemes, such that $\alpha_{n+1} = 2 - \frac{1}{\alpha_n+1}$, $\sigma_{n+1} = M_n$, $M_{n+1} = M_n + 1$ and $\alpha_0 = \sigma_0 = M_0 = 1$. Solving the recurrence equations we get $\alpha_n = \frac{F_{2n+1}}{F_{2n}}$, $\sigma_n = n$, and $M_n = n + 1$. □

Note that this method cannot give a lower bound with the multiplicative factor better than $\lim_{n\to\infty} \frac{F_{2n+1}}{F_{2n}} = \frac{1+\sqrt{5}}{2} \approx 1.61803$. However, we can get arbitrarily close to this bound. That is, for every $\epsilon > 0$ there is a σ and ω_0 such that for each $\omega \geqslant \omega_0$ there is a strategy for Presenter to present intervals of length at most σ and force Algorithm to use $\left(\frac{1+\sqrt{5}}{2} - \epsilon\right) \cdot \omega$ colors on an ω-colorable set of intervals.

Observation 3.9. *There is no online algorithm that works for all* $\sigma \geqslant 1$ *and uses at most* $1.618 \cdot \omega + f(\sigma)$ *colors for* ω*-colorable graphs (for any function* f*).*

3.3.2 The $5/3$ Lower Bound

Lemma 3.10. *If there is an* $\langle \alpha, \sigma, M\rangle$*-schema, then there is a* $\left\langle 2 - \frac{1}{\alpha+2}, M + \epsilon, M + 2 + \epsilon \right\rangle$*-schema for every* $\epsilon > 0$*.*

Proof. The proof of this lemma is very similar to the proof of Lemma 3.7, but now we have two separation phases instead of just one, see Figure 3.3. Let us fix an $\omega \in \mathbb{N}_+$, and let $\omega' = \left\lfloor \frac{\omega}{\alpha+2} \right\rfloor$. Let S be an $\langle \omega', \alpha\omega' - \delta, \sigma, M\rangle$-strategy for some $\delta = o(\omega')$.

In the initial phase, Presenter uses S inside interval $\left[1 + \frac{\epsilon}{2}, M + 1 + \frac{\epsilon}{2}\right]$, and forces Algorithm to use $C = \alpha\omega' - \delta$ colors. Let $\mathcal{X}$ denote the set of those colors.

In the separation phase, Presenter plays the separation strategy two times. First, Presenter plays the separation strategy for ω rounds in the region $\left[0, 1 + \frac{\epsilon}{4}\right]$ pushing to the right colors not in $\mathcal{X}$. Let Y_1 be the set of ω' right-most intervals from this first separation. Let $\mathcal{Y}_1$ denote the set of colors used by Algorithm to color Y_1. Then, Presenter plays the separation strategy for ω rounds in the region $\left[M + 1 + \frac{3\epsilon}{4}, M + 2 + \epsilon\right]$

[1] Knowing the desired target values of n and ϵ, one needs to properly adjust the ϵ value for each application of Lemma 3.7, e.g., it is sufficient to set it to ϵ/n.

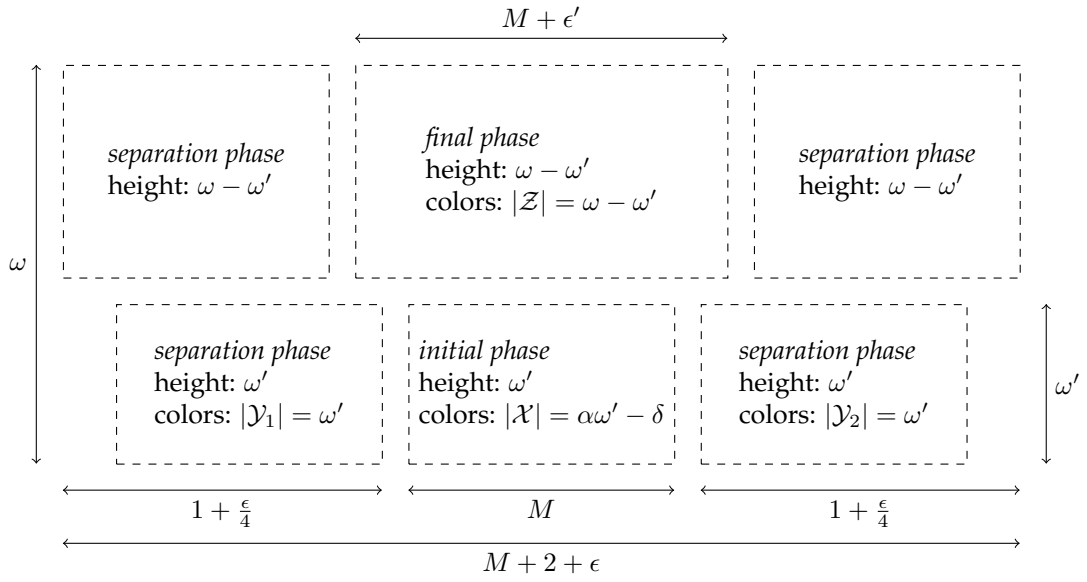

Figure 3.3: Strategy construction in Lemma 3.10

pushing to the left colors not in $\mathcal{X} \cup \mathcal{Y}_1$. Let Y_2 be the set of ω' left-most intervals from this second separation. Let $\mathcal{Y}_2$ denote the set of colors used by Algorithm to color Y_2.

Let r be the left-most right endpoint of an interval in Y_1. Let l be the right-most left endpoint of an interval in Y_2. In the final phase Presenter introduces $\omega - \omega'$ times the same interval $[r, l]$.

The presented set of intervals is clearly ω-colorable and Algorithm used at least $|\mathcal{X}| + |\mathcal{Y}_1| + |\mathcal{Y}_2| + |\mathcal{Z}| = \alpha\omega' - \delta + \omega' + \omega' + \omega - \omega' = \big(2 - \frac{1}{\alpha+2}\big)\omega - o(\omega)$ many colors. The longest interval presented has length at most $M + \epsilon$, and all intervals are contained in $[0, M + 2 + \epsilon]$. Thus, we have constructed a $\big\langle 2 - \frac{1}{\alpha+2}, M + \epsilon, M + 2 + \epsilon \big\rangle$-schema. □

Corollary 3.11. *There is an $\langle \alpha_n, 2n - 1 + \epsilon, 2n + 1 + \epsilon \rangle$-schema, for every $n \in \mathbb{N}_+$ and every $\epsilon > 0$, where*

$$\alpha_n = \frac{\big(\sqrt{3} - 3\big)\big(\sqrt{3} - 2\big)^n + \big(\sqrt{3} + 3\big)\big(-\sqrt{3} - 2\big)^n}{\big(\sqrt{3} - 1\big)\big(\sqrt{3} - 2\big)^n + \big(\sqrt{3} + 1\big)\big(-\sqrt{3} - 2\big)^n}.$$

Proof. The argument is similar to Corollary 3.8, but now we solve the recurrence equation $\alpha_0 = 1$, $\alpha_{n+1} = 2 - \frac{1}{\alpha_n + 2}$. □

Note that, similarly to Observation 3.9, one could already use Corollary 3.11 to get a lower bound arbitrarily close to $\lim_{n\to\infty} \alpha_n = \sqrt{3} \approx 1.73205$ for the asymptotic competitive ratio of any online algorithm that work for all $\sigma \geqslant 1$. Nonetheless in Section 3.3.4 we prove a stronger $5/2$ lower bound.

Theorem 3.12. *For every $\sigma > 1$ there is no online algorithm for σ-interval coloring with the asymptotic competitive ratio less than $5/3$.*

Proof. Assume for contradiction that for some $\sigma > 1$ there exists an online algorithm for σ-interval coloring with the asymptotic competitive ratio $\frac{5}{3} - \epsilon$, for some $\epsilon > 0$. By the definition of the asymptotic competitive ratio, there is an ω_A such that for

every $\omega \geqslant \omega_A$ the algorithm colors every ω-colorable set of intervals using at most $\left(\frac{5}{3} - \epsilon + \frac{\epsilon}{3}\right)\omega = \left(\frac{5}{3} - \frac{2\epsilon}{3}\right)\omega$ colors. Observe that, for $n = 1$, Corollary 3.11 gives a $\left\langle \frac{5}{3}, 1 + (\sigma - 1), 3 + (\sigma - 1) \right\rangle$-schema. By the definition of schema, there is an ω_P such that for every $\omega \geqslant \omega_P$ there is a strategy for Presenter to present an ω-colorable set of intervals, of length in $[1, \sigma]$, and force Algorithm to use $\left(\frac{5}{3} - \frac{\epsilon}{3}\right)\omega$ colors. For $\omega = \max(\omega_A, \omega_P)$ we reach a contradiction. □

3.3.3 The $7/4$ Lower Bound

Lemma 3.13. *If there is an $\langle \alpha, \sigma, M \rangle$-schema, then there is a $\left\langle 2 - \frac{1}{2\alpha+2}, 2M + \epsilon, 2M + 2 + \epsilon \right\rangle$-schema for every $\epsilon > 0$.*

Proof. The proof of this lemma is a bit more complicated than the previous ones, as we now have two initial phases, two separation phases and a strategy branching, see Figure 3.4 and Figure 3.5. Let us fix an $\omega \in \mathbb{N}_+$, and let $\omega' = \left\lfloor \frac{\omega}{\alpha+1} \right\rfloor$. Let S be an $\langle \omega', \alpha\omega' - \delta, \sigma, M \rangle$-strategy for some $\delta = o(\omega')$.

In the initial phase, Presenter uses the strategy S twice: first, inside the interval $\left[1 + \frac{\epsilon}{3}, M + 1 + \frac{\epsilon}{3}\right]$, and then inside the interval $\left[M + 1 + \frac{2\epsilon}{3}, 2M + 1 + \frac{2\epsilon}{3}\right]$. Algorithm uses $C = \alpha\omega' - \delta$ colors in each of these games. We get a set of colors $\mathcal{X}_1$ used by Algorithm in the first game, and a set of colors $\mathcal{X}_2$ used by Algorithm in the second game. Note that $\mathcal{X}_1 \cap \mathcal{X}_2$ might be non-empty.

In the separation phase, Presenter plays the separation strategy two times. First, Presenter plays the separation strategy for ω rounds in the region $\left[0, 1 + \frac{\epsilon}{6}\right]$ pushing to the right colors not in $\mathcal{X}_1$. Let Y_1 be the set of ω' right-most intervals from the first separation phase, and denote $\mathcal{Y}_1$ the set of colors used by Algorithm to color Y_1. Then, Presenter plays the separation strategy for ω rounds in the region $\left[2M + 1 + \frac{5\epsilon}{6}, 2M + 2 + \epsilon\right]$ pushing to the left colors not in $\mathcal{X}_2$. Let Y_2 be the set of ω' left-most intervals from the second separation phase, and denote $\mathcal{Y}_2$ the set of colors used by Algorithm to color Y_2. Let r be the left-most right endpoint of an interval in Y_1, and l be the right-most left endpoint of an interval in Y_2.

There are two cases in the final phase. Let $\mathcal{C}_1 := \mathcal{X}_1 \cup \mathcal{Y}_1$, and analogously $\mathcal{C}_2 := \mathcal{X}_2 \cup \mathcal{Y}_2$. We have that $|\mathcal{C}_1| = |\mathcal{C}_2| = (\alpha + 1)\omega' - \delta = \omega - o(\omega)$.

Case 1: If $|\mathcal{C}_2 \setminus \mathcal{C}_1| \geqslant \frac{\omega}{2\alpha+2}$, then Presenter introduces $\omega - \omega'$ times the same interval $[r, l]$. Each interval introduced in the final phase intersects with all intervals from both initial phases and all intervals in $Y_1 \cup Y_2$. Thus, Algorithm is forced to use $|\mathcal{C}_1 \cup \mathcal{C}_2| + \omega - \omega' = |\mathcal{C}_1| + |\mathcal{C}_2 \setminus \mathcal{C}_1| + \omega - \omega' \geqslant \omega - o(\omega) + \frac{\alpha+\frac{1}{2}}{\alpha+1}\omega = \left(2 - \frac{1}{2\alpha+2}\right)\omega - o(\omega)$ colors in total.

Case 2: If $|\mathcal{C}_2 \setminus \mathcal{C}_1| < \frac{\omega}{2\alpha+2}$, then Presenter introduces ω' intervals, all of them having endpoints $[M + 1 + 5\epsilon/12, l]$. Let Q be the set of colors used by Algorithm in this *pre-final phase*. We have $\mathcal{C}_2 \cap Q = \emptyset$, and we assumed that $|\mathcal{C}_2 \setminus \mathcal{C}_1| \leqslant \frac{\omega}{2\alpha+2}$, thus we have $|Q \setminus \mathcal{C}_1| \geqslant \frac{\omega}{2\alpha+2}$, and now we are in Case 1 with $\mathcal{C}_2 \to Q$, see Figure 3.5.

The longest interval introduced by Presenter in both cases has length strictly less than $2M + \epsilon$, and the whole game is played in the region $[0, 2M + 2 + \epsilon]$. □

Corollary 3.14. *There is an $\langle \alpha_n, 3 \cdot 2^n - 4 + \epsilon, 3 \cdot 2^n - 2 + \epsilon \rangle$-schema, for every $n \in \mathbb{N}_+$ and every $\epsilon > 0$, where*

$$\alpha_n = \frac{\left(\sqrt{7} - 4\right)\left(\sqrt{7} - 3\right)^n + \left(\sqrt{7} + 4\right)\left(-\sqrt{7} - 3\right)^n}{\left(\sqrt{7} - 1\right)\left(\sqrt{7} - 3\right)^n + \left(\sqrt{7} + 1\right)\left(-\sqrt{7} - 3\right)^n}.$$

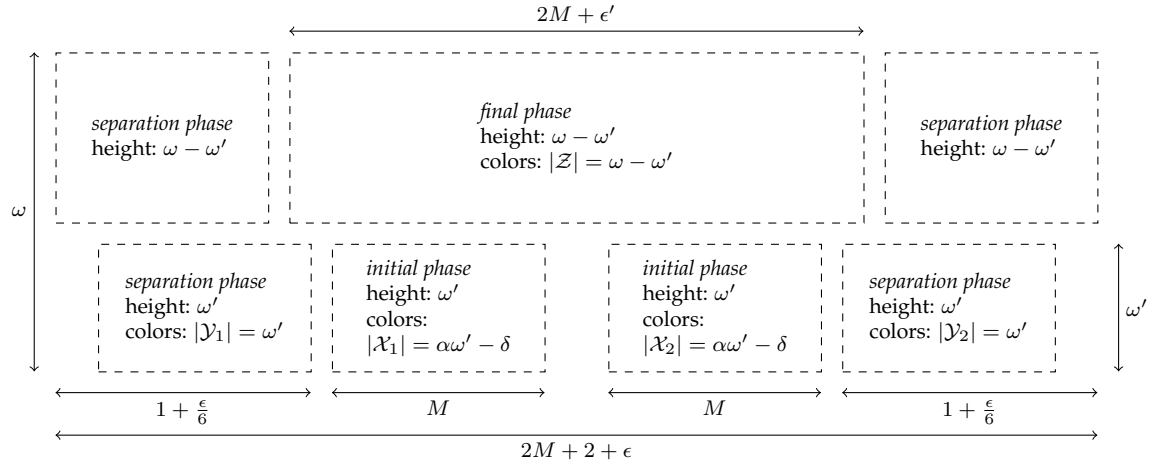

Figure 3.4: Lemma 3.13, Case 1: $|\mathcal{C}_2 \setminus \mathcal{C}_1| \geqslant \frac{\omega}{2\alpha+2}$

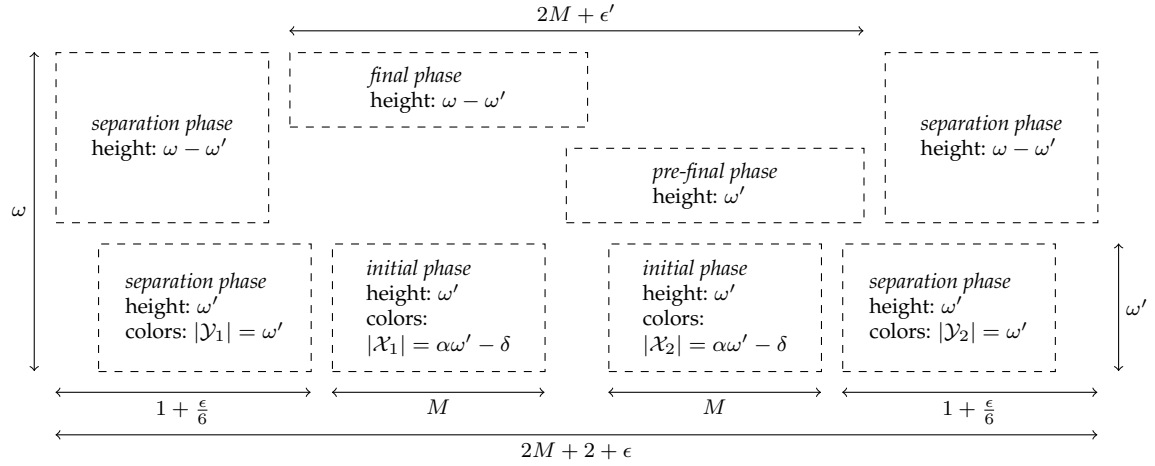

Figure 3.5: Lemma 3.13, Case 2: $|\mathcal{C}_2 \setminus \mathcal{C}_1| < \frac{\omega}{2\alpha+2}$

Proof. The argument is similar to Corollaries 3.8 and 3.11, but now we solve the recurrence equations $\alpha_0 = 1$, $\alpha_{n+1} = 2 - \frac{1}{2\alpha_n+2}$, and $M_0 = 1$, $M_{n+1} = 2M_n + 2$, $\sigma_0 = 1, \sigma_{n+1} = 2M_n$. □

Note that, similarly to Observation 3.9, one could already use Corollary 3.14 to get a lower bound arbitrarily close to $\lim_{n\to\infty} \alpha_n = \frac{1+\sqrt{7}}{2} \approx 1.82287$ for the asymptotic competitive ratio of any online algorithm that work for all $\sigma \geqslant 1$. Nonetheless in Section 3.3.4 we prove a stronger $5/2$ lower bound.

Theorem 3.15. *For every $\sigma > 2$ there is no online algorithm for σ-interval coloring with the asymptotic competitive ratio less than $7/4$.*

Proof. Observe that, for $\sigma > 2$ and $n = 1$, Corollary 3.14 gives a $\left\langle \frac{7}{4}, 2 + (\sigma - 2), 4 + (\sigma - 2) \right\rangle$-schema. Then proceed analogously to the proof of Theorem 3.12. □

3.3.4 The $5/2$ Lower Bound

To prove our main negative result we need two simple combinatorial lemmas.

Lemma 3.16. *Let $\gamma \in [0,1]$. For every four sets $X_1, \ldots, X_4$, each of size k, if their intersection is small: $\left|\bigcap_{i=1}^{4} X_i\right| \leqslant (1-\gamma)\cdot k$, then their sum is large: $\left|\bigcup_{i=1}^{4} X_i\right| \geqslant \frac{3+\gamma}{3}\cdot k$.*

Proof. Each element which belongs to the sum but does not belong to the intersection can belong to at most three sets. Thus, we have

$$3\cdot\left(\left|\bigcup_{i=1}^{4} X_i\right| - \left|\bigcap_{i=1}^{4} X_i\right|\right) \geqslant 4\cdot\left(k - \left|\bigcap_{i=1}^{4} X_i\right|\right),$$

and so

$$3\cdot\left|\bigcup_{i=1}^{4} X_i\right| \geqslant 4k - \left|\bigcap_{i=1}^{4} X_i\right| \geqslant (3+\gamma)\cdot k.$$

□

Lemma 3.17. *Let $\gamma \in [0,1]$, and $X_1, \ldots, X_{4^n}$ be a family of 4^n sets, each of size k. Then, either*

$$\left|\bigcup_{i=1}^{4^n} X_i\right| \geqslant \left(\frac{3+\gamma}{3}\right)^n k,$$

or the sequence $1, 2, \ldots, 4^n$ can be covered with four disjoint intervals $[l_1, r_1], \ldots, [l_4, r_4]$, $l_1 = 1$, $l_{i+1} = r_i + 1$, $r_4 = 4^n$, such that for $Y_i = \bigcup_{j=l_i}^{r_i} X_j$ the intersection of Y_i's is large:

$$|Y_1 \cap Y_2 \cap Y_3 \cap Y_4| \geqslant (1-\gamma)\cdot k.$$

Proof. Consider $n+1$ families of sets defined as follows: $\mathcal{X}_i^0 := X_i$ for every $i \in [4^n]$, and $\mathcal{X}_i^j := \bigcup_{l=4i-3}^{4i} \mathcal{X}_l^{j-1}$ for every $j \in [n]$ and $i \in [4^{n-j}]$, see Figure 3.6.

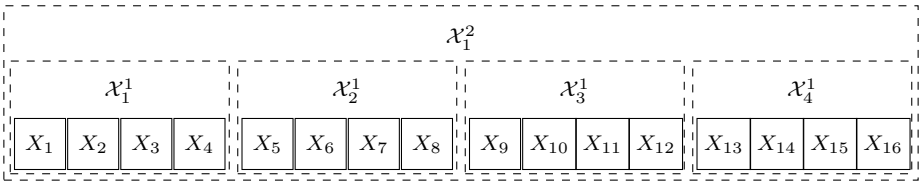

Figure 3.6: $\mathcal{X}_i^j$ sets in Lemma 3.17

If for some i, j we have $|\bigcap_{l=4i-3}^{4i} \mathcal{X}_l^j| \geqslant (1-\gamma)\cdot k$, then we are done. Thus, we assume that $\forall_{i,j}\, |\bigcap_{l=4i-3}^{4i} \mathcal{X}_l^j| < (1-\gamma)\cdot k$. Let $\varrho := \frac{3+\gamma}{3} \in \left[1, \frac{4}{3}\right]$. We prove that $\forall_{i,j}\, |\mathcal{X}_i^j| \geqslant \varrho^j k$, by induction on j. For $j = 0$ the statement is obvious because $\forall_i\, |\mathcal{X}_i^0| = |X_i| = k = \varrho^0 k$. For $j+1$ and arbitrary i, let $k' = \varrho^j k$. By the induction hypothesis $|\mathcal{X}_{4i-3}^j|, \ldots, |\mathcal{X}_{4i}^j| \geqslant \varrho^j k = k'$. We may ignore some elements of those sets and assume that $|\mathcal{X}_{4i-3}^j| = \ldots = |\mathcal{X}_{4i}^j| = k'$, moreover we assumed that $|\mathcal{X}_{4i-3}^j \cap \ldots \cap \mathcal{X}_{4i}^j| < (1-\gamma)k = \frac{1-\gamma}{\varrho^j}\varrho^j k = (1-\gamma')k'$, where $\gamma' \in [0,1]$ and $\gamma' > \gamma$. We apply Lemma 3.16 and get $|\mathcal{X}_{4i-3}^j \cup \ldots \cup \mathcal{X}_{4i}^j| \geqslant \frac{3+\gamma'}{3}k'$. Thus, $|\mathcal{X}_i^{j+1}| \geqslant \frac{3+\gamma'}{3}k' > \frac{3+\gamma}{3}k' = \varrho k' = \varrho^{j+1}k$. □

Finally, we are ready to show our best lower bound strategies.

Lemma 3.18. *If there is an $\langle \alpha, \sigma, M\rangle$-schema, then for every $\epsilon > 0$ and for every $\gamma \in (0,1)$, there is a $\left\langle \frac{5}{4} + \frac{1}{2}(1-\gamma)\alpha, 4^n M + \epsilon, 4^n M + \epsilon \right\rangle$-schema, for some $n := n(\gamma)$.*

Proof. Let us fix an $\omega \in \mathbb{N}_+$, and let $\omega' = \left\lfloor \frac{\omega}{2} \right\rfloor$. Let S be an $\langle \omega', \alpha\omega' - \delta, \sigma, M\rangle$-strategy for some $\delta = o(\omega')$. Presenter repeats strategy S in the initial phase 4^n times. For each $i \in [4^n]$ the i-th game is played inside interval $\left[(i-1)(M + \frac{\epsilon}{4^n}), (i-1)(M + \frac{\epsilon}{4^n}) + M\right]$, see Figure 3.7. Algorithm uses $C = \alpha\omega' - \delta$ colors in each of these games. Let $\mathcal{X}_i$ denote the set of C colors used by Algorithm in the i-th game. Let $\mathcal{X}$ denote the set of all colors used in the initial phase, i.e., $\mathcal{X} = \bigcup_{i\in[4^n]} \mathcal{X}_i$.

We apply Lemma 3.17 to the family $\mathcal{X}_1, \ldots, \mathcal{X}_{4^n}$ and get that either the union of these sets has at least $\left(\frac{3+\gamma}{3}\right)^n C$ elements, or we get four disjoint consecutive subfamilies $\mathcal{Y}_1, \ldots, \mathcal{Y}_4$ ($\mathcal{Y}_i = \bigcup_{j=l_i}^{r_i} \mathcal{X}_j$) such that the size of the intersection $\mathcal{Y}_1 \cap \mathcal{Y}_2 \cap \mathcal{Y}_3 \cap \mathcal{Y}_4$ has at least $(1-\gamma)\cdot C$ elements.

Case 1: If the size of the union $|\mathcal{X}|$ is at least $\left(1 + \frac{\gamma}{3}\right)^n \cdot C$, then Presenter introduces ω' intervals, all of them having endpoints $[0, 4^n M + \epsilon]$, see Figure 3.7. Each interval introduced in the final phase intersects with all intervals introduced in the initial phase. Thus, Algorithm is forced to use at least $|\mathcal{X}| + \omega' \geqslant \frac{1}{2}\left(\left(1 + \frac{\gamma}{3}\right)^n \alpha + 1\right)\omega - o(\omega)$ colors in total. Easy calculation shows that for $\gamma \in (0,1)$, $\alpha \in [1,3]$ and for any $n \geqslant \log_{1+\frac{\gamma}{3}}(5/2 - \gamma)$, we have $\frac{1}{2} + \frac{1}{2}\left(1 + \frac{\gamma}{3}\right)^n \alpha \geqslant \frac{5}{4} + \frac{1}{2}(1-\gamma)\alpha$.

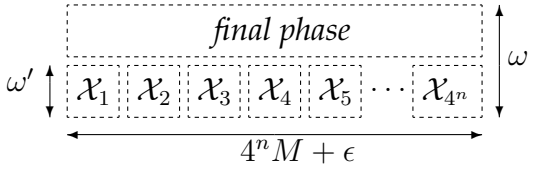

Figure 3.7: Case 1: $|\mathcal{X}|$ is large

Case 2: The size of the intersection $|\mathcal{Y}_1 \cap \ldots \cap \mathcal{Y}_4|$ is at least $(1-\gamma)\cdot C$. Let $\mathcal{Y} = \mathcal{Y}_1 \cap \mathcal{Y}_2 \cap \mathcal{Y}_3 \cap \mathcal{Y}_4$ denote the colors that appear in all four parts of the initial phase. Presenter introduces set Z_1 of ω' identical intervals covering all intervals contributing to $\mathcal{Y}_1$ and disjoint with intervals contributing to $\mathcal{Y}_2$, see Figure 3.8. Let $\mathcal{Z}_1$ be the set of colors used by Algorithm to color Z_1. Then Presenter introduces set Z_2 of ω' identical intervals covering all intervals contributing to $\mathcal{Y}_4$ and disjoint with intervals contributing to $\mathcal{Y}_3$. Let $\mathcal{Z}_2$ be the set of colors used by Algorithm to color Z_2. Clearly, $|\mathcal{Z}_1| = |\mathcal{Z}_2| = \omega'$, and $\mathcal{Z}_1 \cap \mathcal{Y} = \mathcal{Z}_2 \cap \mathcal{Y} = \emptyset$. Now we distinguish two subcases depending on the size of the set $\mathcal{Z}_2 \setminus \mathcal{Z}_1$.

Case 2.1: If $|\mathcal{Z}_2 \setminus \mathcal{Z}_1| \geqslant \frac{1}{4}\omega$, then Presenter introduces set W of ω' identical intervals intersecting all the intervals in Z_1 and Z_2, and covering all the intervals contributing to $\mathcal{Y}_2$ and $\mathcal{Y}_3$. Let $\mathcal{W}$ be the set of colors used by Algorithm to color W. By the definition, we have $\mathcal{W} \cap \mathcal{Y} = \mathcal{W} \cap \mathcal{Z}_1 = \mathcal{W} \cap \mathcal{Z}_2 = \emptyset$. Algorithm was forced to use $|\mathcal{W}| + |\mathcal{Z}_1| + |\mathcal{Z}_2 \setminus \mathcal{Z}_1| + |\mathcal{Y}| \geqslant \left(\frac{1}{2} + \frac{1}{2} + \frac{1}{4}\right)\omega + \frac{1}{2}(1-\gamma)\alpha\omega - o(\omega) = \left(\frac{5}{4} + \frac{1}{2}(1-\gamma)\alpha\right)\omega - o(\omega)$ colors in total, see Figure 3.8.

Case 2.2: If $|\mathcal{Z}_2 \setminus \mathcal{Z}_1| < \frac{1}{4}\omega$, then let $\mathcal{Z} = \mathcal{Z}_1 \cap \mathcal{Z}_2$ and observe that $|\mathcal{Z}| \geqslant \left\lfloor \frac{\omega}{4} \right\rfloor$. Presenter introduces set W_1 of ω' identical intervals intersecting all the intervals in Z_1,

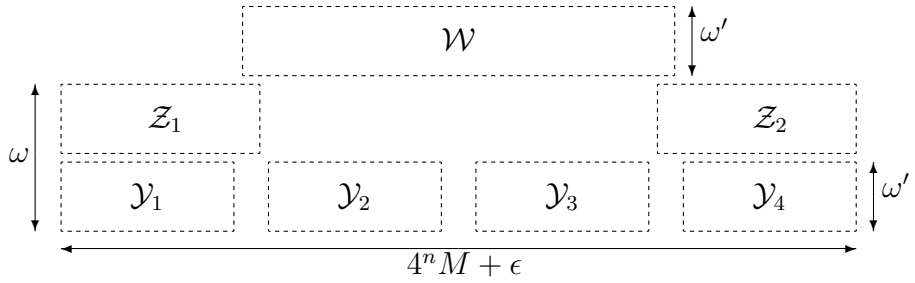

Figure 3.8: Case 2.1: $|\mathcal{Y}|$ is large and $|\mathcal{Z}_2 \setminus \mathcal{Z}_1| \geqslant \frac{1}{4}\omega$

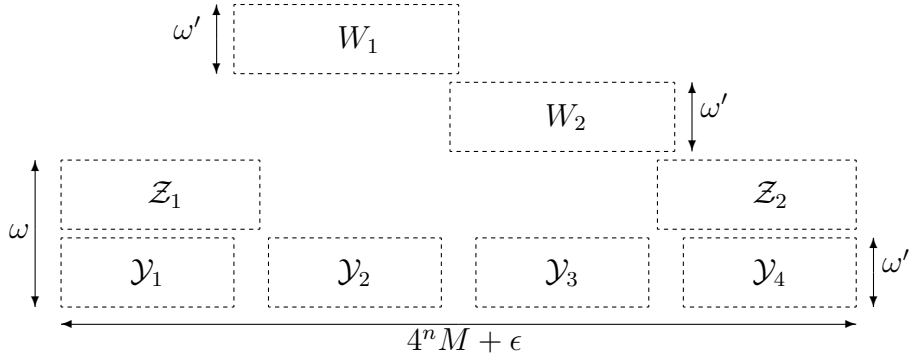

Figure 3.9: Case 2.2: $|\mathcal{Y}|$ is large and $|\mathcal{Z}_2 \setminus \mathcal{Z}_1| < \frac{1}{4}\omega$

and covering all the intervals contributing to $\mathcal{Y}_2$. Then, presenter introduces set W_2 of ω' identical intervals, intersecting all the intervals in W_1 and Z_2, and covering all the intervals contributing to $\mathcal{Y}_3$. Let $\mathcal{W}$ be the set of colors used by Algorithm to color intervals in $W_1 \cup W_2$. We have that $|\mathcal{W}| = 2\omega'$, and $\mathcal{W} \cap \mathcal{Y} = \mathcal{W} \cap \mathcal{Z} = \emptyset$. Algorithm was forced to use $|\mathcal{W}| + |\mathcal{Z}| + |\mathcal{Y}| \geqslant \left(1 + \frac{1}{4}\right)\omega + \frac{1}{2}(1-\gamma)\alpha\omega - o(\omega) = \left(\frac{5}{4} + \frac{1}{2}(1-\gamma)\alpha\right)\omega - o(\omega)$ colors in total, see Figure 3.9. □

Corollary 3.19. *There is an an $\langle \alpha_n, 4^{nf(\gamma)} + \epsilon, 4^{nf(\gamma)} + \epsilon \rangle$-schema, for every $n \in \mathbb{N}_+$, every $\epsilon > 0$, and every $\gamma \in (0, 1)$, where*

$$\alpha_n = \frac{5}{2} \cdot \frac{1}{1+\gamma} - \frac{(3-2\gamma)}{2(1+\gamma)} \cdot \left(\frac{1-\gamma}{2}\right)^n, \quad f(\gamma) = \left\lceil \frac{\log\left(\frac{5}{2} - \gamma\right)}{\log\left(1 + \frac{\gamma}{3}\right)} \right\rceil.$$

Proof. The argument is similar to Corollaries 3.8, 3.11, and 3.14, but now we solve the recurrence equations $\alpha_0 = 1$, $\alpha_{n+1} = \frac{5}{4} + \frac{1}{2}(1-\gamma)\alpha_n$ for competitive ratio, and $M_0 = 1$, $M_{n+1} = 4^{f(\gamma)} M_n$, $\sigma_n = M_n$ for region and interval lengths. □

Theorem 3.20. *For every $\epsilon > 0$ there is $\sigma \geqslant 1$ such that there is no online algorithm for σ-interval coloring with the asymptotic competitive ratio $5/2 - \epsilon$.*

Proof. Assume for contradiction that, for some $\epsilon > 0$, there are $(5/2 - \epsilon)$-competitive algorithms for every $\sigma \geqslant 1$. Setting γ small enough and n large enough, Corollary 3.19 gives us a $\langle \frac{5}{2} - \frac{\epsilon}{4}, \sigma, \sigma \rangle$-schema, for some value of σ. This means, there is ω_P such that for every $\omega \geqslant \omega_P$ there exists an $\langle \omega, (\frac{5}{2} - \frac{2\epsilon}{4})\omega, \sigma, \sigma \rangle$-strategy. On the other hand, for the assumed σ-interval coloring algorithm, there exists ω_A such that for every $\omega \geqslant \omega_A$ the algorithm uses at most $\left(\frac{5}{2} - \frac{3\epsilon}{4}\right)\omega$ colors for every ω-colorable set of intervals. For $\omega = \max(\omega_A, \omega_P)$ we reach a contradiction. □

3.4 Open Problems

There are still large gaps between the best known lower and upper bounds for the optimal competitive ratios for online σ-interval coloring problems, see Figure 3.10. It would be interesting to close the gap, even for a single specific σ. For example, for $\sigma = 3/2$ the optimal online algorithm has the competitive ratio somewhere between $5/3$ and $5/2$.

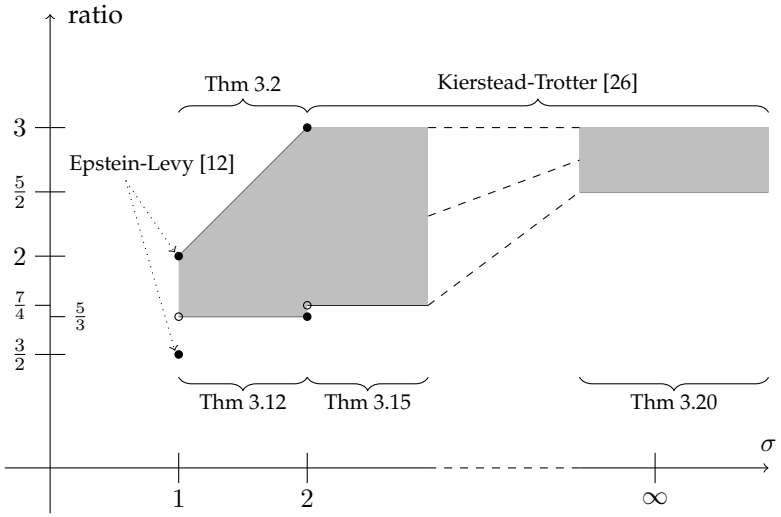

Figure 3.10: The state of art bounds on the asymptotic competitive ratio as a function of σ.

Finally, let us conjecture that the lower bound of Theorem 3.20 is tight.

Conjecture 3.21. *There is a $5/2$-competitive online algorithm for σ-interval coloring, for every $\sigma \geqslant 1$.*

4

FirstFit in Online Interval Coloring of Short Intervals with restricted Bandwidths

4.1 Introduction

The FirstFit algorithm is the simplest and the most natural greedy algorithm for any graph coloring problem. When a new vertex is presented, FirstFit colors it with the smallest positive integer that does not violate the problem-specific constraints. It is quite easy to see that FirstFit is very inefficient on the family of bipartite graphs (or even on forests). One can easily produce a bipartite graph on $2n$ vertices on which FirstFit uses n colors. However, in the class of interval graphs, FirstFit achieve a constant competitive ratio. The FirstFit algorithm has been intensively studied in the case of interval graphs [7, 23, 24, 25, 31, 33]. Narayanaswamy and Babu [31] proved that the competitive ratio of the FirstFit algorithm in this class is at most 8, while Kierstead et al. [25] showed a lower bound of 5. Unfortunately, FirstFit performs arbitrarily bad when intervals have associated bandwidths [2, 34].

In this chapter we show a series of bounds on the competitive ratio of the FirstFit algorithm in the online σ-interval coloring with $[\alpha, \beta]$-bandwidths under some reasonable assumptions about the lengths and bandwidths of presented intervals.

We decide to include the interval representation in the input, instead of presenting the mere graph. That's because the FirstFit algorithm works on the graph and it's representation in the same way, and this assumption makes some arguments easier. Moreover, wlog. we assume that all intervals introduced by Presenter have lengths at least 1.

4.1.1 Our Results

In Section 4.2 we analyze the case in which all bandwidths are allowed, i.e. $\alpha = 0$ and $\beta = 1$. FirstFit does not have a constant competitive ratio in this case [2, 34], but it turns out that this ratio is bounded by a function of σ. In fact, we prove that in this case both absolute and asymptotic competitive ratios of the FirstFit algorithm are $\sigma + o(\sigma)$.

In Section 4.3 we prove that if Presenter cannot use arbitrarily small or big bandwidths i.e. $\alpha > 0$ or $\beta < 1$, then the linear in terms of σ lower bound is not valid anymore. We show that in this case the competitive ratio depends only on α and β.

Sections 4.4 and 4.5 contain results in the case where $\sigma = 1$, i.e. we deal with the online unit interval coloring with bandwidths. In Section 4.4, we investigate the case in which all bandwidths are available. In that case we prove that the asymptotic competi-

tive ratio of the FirstFit algorithm is between 3.38 and 5, while the absolute competitive ratio is between 3.38 and 6.

In Section 4.5 we focus on the competitive ratio in the case where all small bandwidths are available but Presenter cannot use big bandwidths. Considering such a scenario is a common case. For instance in the bin-packing problem there is a series of papers that deal with the r-parameterized bin-packing, i.e. a bin-packing problem in which all items are not greater than $\frac{1}{r}$. The online interval coloring with bandwidths is a generalization of the bin-packing problem. Hence, we investigate that case too, and present quite tight analysis especially for small values of β.

Definitions

For an interval $I = [l, r]$ we denote:

- $length(I)$ the length of I, i.e. $length(I) = r - l$,
- $w(I)$ the bandwidth associated with I, and
- $col(I)$ the color assigned by the FirstFit algorithm to I.

For a color γ and a point $p \in \mathbb{R}$ we denote $\omega_p^*(\gamma)$ the sum of bandwidths of all intervals that are colored with γ and contain p. For a point $p \in \mathbb{R}$ we denote ω_p^* the sum of bandwidths of all intervals that contain p, i.e. $\omega_p^* = \sum_\gamma \omega_p^*(\gamma)$. We also denote ω^* the maximum value of ω_p^* over all points p. Clearly, at least $\lceil \omega^* \rceil$ colors are required to properly color the input graph.

4.2 FirstFit Online σ-Interval Coloring with Bandwidths.

We start with a simple observation about the FirstFit algorithm.

Observation 4.1. *If $\omega^* \leqslant 1$, then FirstFit is optimal.*

Clearly, each interval gets the same color and the proper coloring constraint is satisfied. Thus, in the rest of this chapter we assume that $\omega^* > 1$.

The following construction, shows that FirstFit performs arbitrarily bad. We modify the construction presented by Adamy and Erlebach [2] in such a way that it works with intervals with lengths in $[1, \sigma]$ and gives an asymptotic lower bound, which is a linear function in terms of σ.

Theorem 4.2. *For every $\sigma \geqslant 1$, both absolute and asymptotic competitive ratios of the FirstFit algorithm in the online σ-interval coloring with bandwidths are at least $\lceil \sigma \rceil + 1$.*

Proof. Let $\epsilon \in \left(0, \frac{1}{2}\right)$, $k \in \mathbb{N}_+$, $n := \lceil \sigma \rceil + 1$ and $\delta := \frac{\sigma + 1 - \lceil \sigma \rceil}{\lceil \sigma \rceil}$ which is positive for every $\sigma \geqslant 1$. The strategy for Presenter consists of kn phases. In the i-th phase Presenter introduces two intervals: a long interval $I_i = [0, \sigma]$ with bandwidth ϵ^i and a short one $J_i = [(1+\delta)(j-1) - 1, (1+\delta)(j-1)]$ with bandwidth $1 - \epsilon^i$ where $i = an + j$ for some $j \in [n]$, see Figure 4.1.

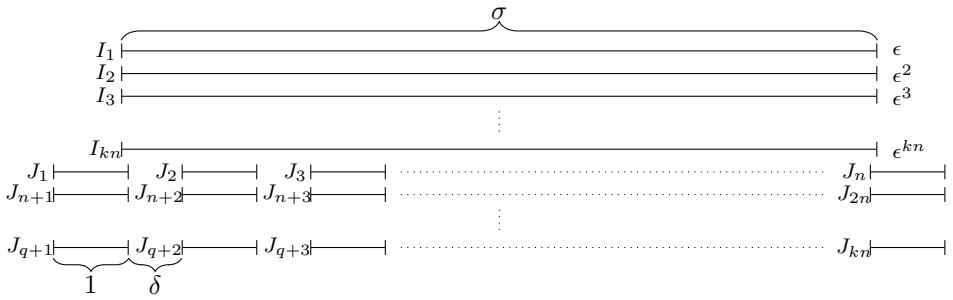

Figure 4.1: A strategy for Presenter in the online σ-interval coloring with bandwidths. In order to simplify indices we put $q := (k-1)n$.

Observe that each long interval intersects all short intervals. It is easy to see that the right endpoint of the leftmost short interval is 0 and the left endpoint of the rightmost short interval is $(1+\delta)(n-1)-1 = \sigma$.

We induct on i to show that both intervals I_i and J_i are colored with a color i. It is easy to check that for all $j < i$ there is a point $p \in [0, \sigma]$ such that $\omega_p^*(j) = 1$, thus the long interval I_i cannot get any color $j < i$. Moreover, there is no interval colored with i yet, so the interval I_i gets a color i. The interval J_i with a bandwidth $1-\epsilon^i$ intersects all long intervals $I_1, \ldots, I_{i-1}$ with bandwidths $\epsilon, \ldots, \epsilon^{i-1}$ respectively. Because for every $1 \leqslant j < i$ we have $1 - \epsilon^i + \epsilon^j > 1$, the interval J_i cannot get any color $j < i$, but it can be colored with i.

Now we show that the presented set of intervals can be colored offline with $k+1$ colors. We color all short intervals using k colors by coloring intervals J_{an+j} with a color a for every a and j. Since $\epsilon + \epsilon^2 + \ldots + \epsilon^{kn} < \frac{\epsilon}{1-\epsilon} < 1$ for $\epsilon < \frac{1}{2}$, we can color all long intervals with a single color. Therefore, we colored all intervals using $k+1$ colors in total.

For fixed $k \in \mathbb{N}_+$ the presented strategy ensures that the competitive ratio of the FirstFit algorithm is at least $\frac{nk}{k+1}$. Taking large values of k we can get as close to $n = \lceil \sigma \rceil + 1$ as we want to. □

We just proved that the competitive ratio of the FirstFit algorithm in the online σ-interval coloring with bandwidths dominates the linear growth of σ. In the next few lemmas and theorems we show a matching upper bound of the form $\sigma + o(\sigma)$ on the competitive ratio of the FirstFit algorithm in this problem. At first, we introduce a notion of the *accumulation points* – a characteristic set of points associated with an interval that helps in the analysis of the FirstFit algorithm.

Definition 4.3. *For an interval $I = [l, r]$, a set of points $p_1 = l, p_2 = l+1, \ldots, p_q = l+q-1$, where $q := \lceil r - l \rceil + 1$ is called a set of* accumulation points *of I, see Figure 4.2.*

First, we prove a quite easy and natural lemma, which then will be used in more complicated constructions to obtain better results.

Observation 4.4. *If FirstFit colored an interval I with a color c, then $c \leqslant \left\lfloor \frac{S}{1-w(I)} \right\rfloor + 1$, where S is the sum of bandwidths of all intervals that intersect I.*

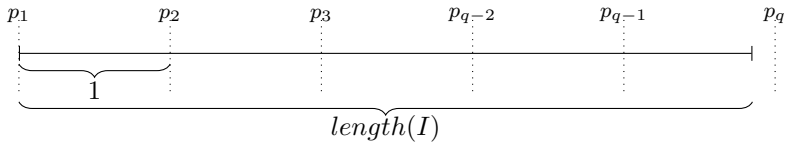

Figure 4.2: Accumulation points for an interval I.

Proof. If for some color j the sum of bandwidths of all intervals that I intersects and are colored with j does not exceed $1 - w(I)$, then I may be colored with j. Thus, the color of an interval I is not greater than $\left\lfloor \frac{S}{1-w(I)} \right\rfloor + 1$. □

Lemma 4.5. *If FirstFit colored an interval I with a color c, then $c \leqslant \left\lfloor \frac{(\lceil length(I) \rceil + 1)\omega^*}{1-w(I)} \right\rfloor + 1$.*

Proof. If all intervals have length at least 1, then any interval I' which intersects I has to contain at least one accumulation point of I. Moreover, at any accumulation point the sum of bandwidths of all intervals that contain this point cannot exceed ω^*. Thus, Observation 4.4 for $S = (\lceil length(I) \rceil + 1)\omega^*$ leads to the desired result. □

Now we improve the upper bound of Lemma 4.5 to the one that does not depend on the bandwidth associated with an interval.

Lemma 4.6. *If all intervals have lengths in $[1, \sigma]$, then for every integer $k \geqslant 2$, FirstFit colors an interval I with a color not greater than $\left(\frac{k}{k-1}(\lceil \sigma \rceil + 1) + (k-1)(\lceil length(I) \rceil + 1)\right)OPT + 2$.*

Proof. If all intervals have bandwidths greater than $\frac{1}{k}$, then at most $\lfloor (k-1)\omega^* \rfloor$ different intervals may contain a given accumulation point of I. Interval I has $\lceil length(I) \rceil + 1$ accumulation points. Thus, I can intersect at most $(\lceil length(I) \rceil + 1)\lfloor (k-1)\omega^* \rfloor$ intervals in total, and we are done in this case.

In case that light intervals exist we carefully pick one of them. First, put $n := col(I) - 1$, $q := \lceil length(I) \rceil + 1$ and let $\mathcal{I}_i$, for $i \in [n]$, be the set of all neighbors of I that are colored with a color i. For each set $\mathcal{I}_i$ choose an interval $I_i \in \mathcal{I}_i$ with the smallest bandwidth to be the representative of this set. If for some set $\mathcal{I}_i$ there are many candidates with the same bandwidth, then pick any of them. We assumed that $col(I) = n+1$, so all sets $\mathcal{I}_i$ are not empty and such representatives exist. Let I_a be a representative with the highest color and bandwidth not greater than $\frac{1}{k}$, i.e. $\forall_{j>a} : w(I_j) > \frac{1}{k}$ and $w(I_a) \leqslant \frac{1}{k}$. Call representatives with colors higher than a *heavy representatives*.

For each heavy representative define its *characteristic point* as the leftmost accumulation point of I that this representative contains. For $i \in [q]$, let x_i be the number of heavy representatives for whom p_i is a characteristic point. Clearly, $\sum_{i=1}^{q} x_i = n - a$ and from the definition $a = col(I_a)$, which implies that $col(I) = col(I_a) + \sum_{i=1}^{q} x_i + 1$. Moreover, if there are x_i intervals containing p_i and each of them has bandwidth strictly greater than $\frac{1}{k}$, then the proper coloring requires at least $\left\lceil \frac{x_i}{k-1} \right\rceil$ colors. Thus, the minimal number of colors required in the proper coloring is at least $\left\lceil \max\left\{\omega^*, \frac{x_1}{k-1}, \ldots, \frac{x_q}{k-1}\right\} \right\rceil$.

Lemma 4.5 implies that $col(I_a) \leqslant \left\lfloor \frac{(\lceil length(I_a) \rceil + 1)\omega^*}{1 - w(I_a)} \right\rfloor + 1$. Thus,

$$col(I) \leqslant \frac{k}{k-1}(\lceil \sigma \rceil + 1)\omega^* + \sum_{i=1}^{q} x_i + 2$$

Case 1: $\forall_i : \omega^* \geqslant \frac{x_i}{k-1}$, which means that $\forall_i : x_i \leqslant (k-1)\omega^*$.
Hence, $col(I) \leqslant \frac{k}{k-1}(\lceil \sigma \rceil + 1)\omega^* + q(k-1)\omega^* + 2$ and we are done, because $OPT \geqslant \lceil \omega^* \rceil$.

Case 2: $\exists_i : \frac{x_i}{k-1} \geqslant \omega^*$. Take i with the biggest x_i.
In this case we have, $col(I) \leqslant \frac{k}{k-1}(\lceil \sigma \rceil + 1)\frac{x_i}{k-1} + q(k-1)\frac{x_i}{k-1} + 2$ and yet again we are done, because $OPT \geqslant \left\lceil \frac{x_i}{k-1} \right\rceil$. □

Using Lemma 4.6, one can easily prove that both absolute and asymptotic competitive ratios of the FirstFit algorithm in the online σ-interval coloring with bandwidths are linear functions in terms of σ.

Corollary 4.7. *If all intervals have lengths in $[1, \sigma]$, then the maximum color used by FirstFit is bounded from above by $3(\lceil \sigma \rceil + 1)OPT + 2$.*

Proof. Lemma 4.6 for $length(I) = \sigma$ yields a bound $col(I) \leqslant \left(\frac{k}{k-1} + k - 1\right)(\lceil \sigma \rceil + 1)OPT + 2$. A function $f(k) = \frac{k}{k-1} + k - 1 = \frac{k^2 - k + 1}{k-1}$ minimizes for $k = 2$ with $f(2) = 3$, which gives the desired result. □

Thus, for every $\sigma \geqslant 1$, the asymptotic competitive ratio of the FirstFit algorithm in the online σ-interval coloring with bandwidths is at most $3\lceil \sigma \rceil + 3$, while the absolute competitive ratio is at most $3\lceil \sigma \rceil + 4$ (take $OPT = 2$). Although, this is not the best we can do. In Theorem 4.9 we prove an upper bound with a multiplicative constant 1 instead of 3, which matches the lower bound presented in Theorem 4.2. Before we state that theorem, we make an obvious but useful observation about the FirstFit behavior.

Observation 4.8. *Let $\mathcal{I} = [I_1, \ldots, I_n]$ be a sequence of intervals originally colored by FirstFit by the colors $col_{\mathcal{I}}(I_i)$, and $\mathcal{J} = [I_{i_1}, \ldots, I_{i_k}]$ be a subsequence of $\mathcal{I}$ after removing all the I_i's with $col_{\mathcal{I}}(I_i) = 1$. Then, while coloring the intervals from $\mathcal{J}$, FirstFit assigns $col_{\mathcal{J}}(I_i) = col_{\mathcal{I}}(I_i) - 1$.*

Now we are ready to state our best upper bound on the competitive ratio of the FirstFit algorithm in the online σ-interval coloring with bandwidths.

Theorem 4.9. *For every $\sigma \geqslant 1$, the maximal color used by FirstFit in the online σ-interval coloring with bandwidths is bounded from above by $\left(\sigma + 5\sigma^{2/3} + 3\sigma^{1/3} + \frac{2}{\sigma^{1/3}} + 10\right)OPT + 4$.*

Proof. Let $x \in (1, \sigma)$ be some constant determined later. Lemma 4.6 implies that for every integer $k \geqslant 2$, the maximal color used by FirstFit on an interval shorter than x does not exceed $C(x) := \left(\frac{k}{k-1}(\lceil \sigma \rceil + 1) + (k-1)(\lceil x \rceil + 1)\right)OPT + 2$. Let $\mathcal{I}_{high}$ be the set of intervals colored by FirstFit with colors higher than $C(x)$. Applying Observation 4.8 $C(x)$ times, we get that, if one use FirstFit to color all intervals from the set $\mathcal{I}_{high}$ with the same order as in the original sequence, then all colors are shifted by $C(x)$ in respect to the original ones.

Clearly, all intervals in the set $\mathcal{I}_{high}$ have lengths in $[x, \sigma]$. Hence, one can see this as an instance of the online $\left(\frac{\sigma}{x}\right)$-interval coloring with bandwidths, by rescaling all intervals in $\mathcal{I}_{high}$ by the factor x. Corollary 4.7 implies now that the maximal color that FirstFit uses when it is applied to the set $\mathcal{I}_{high}$ does not exceed $C_{high}(x) := 3\left(\left\lceil \frac{\sigma}{x} \right\rceil + 1\right)OPT + 2$. This combined with the fact that colors of those intervals in the original sequence are shifted by $C(x)$ implies that the maximal color that FirstFit uses on the original instance is at most $C(x) + C_{high}(x)$. Thus, the color of every interval is bounded from above by

$$\left(\frac{k}{k-1}(\sigma+2) + (k-1)(x+2) + 3\left(\frac{\sigma}{x}+2\right)\right)OPT + 4$$

and this bound holds for every integer $k \geqslant 2$, and every $x \in (1, \sigma)$. Take $k := \left\lceil \sigma^{1/3} \right\rceil + 1$, which is not less than 2 for every $\sigma \geqslant 1$, and $x = \sigma^{1/3}$, which clearly belongs to $(1, \sigma)$ for every $\sigma \geqslant 1$. This leads to the following upper bound

$$\left(\left(1 + \frac{1}{\left\lceil \sigma^{1/3} \right\rceil}\right)(\sigma+2) + \left\lceil \sigma^{1/3} \right\rceil\left(\sigma^{1/3}+2\right) + 3\left(\frac{\sigma}{\sigma^{1/3}}+2\right)\right)OPT + 4.$$

Since $\left\lceil \sigma^{1/3} \right\rceil \leqslant \sigma^{1/3} + 1$ and $\frac{1}{\left\lceil \sigma^{1/3} \right\rceil} \leqslant \frac{1}{\sigma^{1/3}}$, the above expression reduces to

$$\left(\sigma + 5\sigma^{2/3} + 3\sigma^{1/3} + \frac{2}{\sigma^{1/3}} + 10\right)OPT + 4,$$

and we are done. □

4.3 FirstFit Online σ-Interval Coloring with $[\alpha, \beta]$-Bandwidths.

Theorems 4.2 and 4.9 provide a tight analysis of the FirstFit algorithm in the case where all intervals have associated bandwidths from $[0, 1]$. Although, the construction in Theorem 4.2 used both arbitrarily small and arbitrarily big bandwidths in order to give a linear in terms of σ lower bound. In this section we investigate the online σ-interval coloring with bandwidths, in which all intervals have bandwidths in $[\alpha, \beta]$ with $0 < \alpha$ or $\beta < 1$. In fact in this setting the linear in terms of σ lower bound does not apply anymore. The destruction of this linear bound can be easily seen for $\alpha > \frac{1}{2}$.

Observation 4.10. *If $\alpha > \frac{1}{2}$, then any two intersecting intervals have to get different colors, and the problem reduces to the online σ-interval coloring problem.*

Next we recall that for $\beta < 1$ Pemmaraju et al. [33] proved an upper bound of form $\frac{16}{1-\beta}$ while analyzing the Adamy and Erlebach algorithm [2].

Theorem 4.11 (Pemmaraju et al. [33])**.** *For any set $\mathcal{I}$ of intervals whose maximum bandwidth is $\beta \in (0, 1)$, the FirstFit algorithm uses at most $\frac{16}{1-\beta} \cdot OPT(\mathcal{I}) + 1$ colors.*

A careful analysis of the original proof of Theorem 4.11 allows us to almost literally repeat their arguments to get the following.

Theorem 4.12. *For any set $\mathcal{I}$ of intervals whose minimum bandwidth is $\alpha \in (0, 1)$, the FirstFit algorithm uses at most $\frac{16}{\alpha} \cdot OPT(\mathcal{I}) + 1$ colors.*

Indeed, in the proof of Theorem 4.11 authors combined the *column construction method* with the following observation on the FirstFit algorithm.

Observation 4.13 (Pemmaraju et al. [33])**.** *If all intervals have bandwidths at most $\beta \in (0, 1)$ and an interval I is colored with a color $i \geqslant 1$, then for any color $j < i$, there is a point $p \in I$ such that $\omega_p^*(j) > 1 - \beta$.*

Proof. Clearly if there is no such point, then since the bandwidth of I is at most β, it can be colored with a smaller color contradicting the fact that FirstFit always uses the smallest possible color. □

Equally easily we can prove the next observation.

Observation 4.14. *If all intervals have bandwidths at least $\alpha \in (0, 1)$ and an interval I is colored with a color $i \geqslant 1$, then for any color $j < i$, there is a point $p \in I$ such that $\omega_p^*(j) \geqslant \alpha$.*

Proof. Interval I is colored with a color i, so I intersects at least one interval colored with a color j for every $j < i$. Since each interval has bandwidth at least α, we are done. □

Coming back to the argument for Theorem 4.11 we note that replacing Lemma 4.13 by Lemma 4.14, and changing RULE 1 in the column construction method to assign the symbol R when the density of a given color in a given elementary interval is at least $\alpha/2$ instead of $(1 - \beta)/2$, see [33] for details, we get the required upper bound.

Theorems 4.11 and 4.12 together provide an upper bound on the competitive ratio of the FirstFit algorithm of the form $\frac{16}{\max\{\alpha, 1-\beta\}}$. A slight modification of the strategy presented in Theorem 4.2 can be used to show a lower bound which is roughly $\frac{1}{\max\{\alpha, 1-\beta\}}$.

Theorem 4.15. *For every $\sigma \geqslant 1$, $\alpha \in \left(0, \frac{1}{2}\right)$, and $\beta \in \left(\frac{1}{2}, 1\right)$, both absolute and asymptotic competitive ratios of the FirstFit algorithm in the online σ-interval coloring with $[\alpha, \beta]$-bandwidths are at least $\frac{(\lceil\sigma\rceil+1)(\lfloor 1/\mu\rfloor-1)}{\lceil\sigma\rceil+\lfloor 1/\mu\rfloor}$, where $\mu = \max\{\alpha, 1 - \beta\}$.*

Proof. As in the proof of Theorem 4.2 we present the intervals $I_1, \ldots, I_{kn}, J_1, \ldots, J_{kn}$ in the very same order. Although, we need to make sure that the bandwidths of the I_i's and J_i's fall into the interval $[\alpha, \beta]$. This forces some additional restrictions so that $\epsilon < \min\{1 - \alpha, \beta\} - \mu$, and the bandwidth ϵ^i of I_i is replaced by $\mu + \epsilon^i$ while the bandwidth $1 - \epsilon^i$ of J_i is replaced by $1 - \mu - \epsilon^i$.

Observe that $\mu + \epsilon < \beta$ and $1 - \mu - \epsilon > 1 - \min\{1 - \alpha, \beta\} > \alpha$. Thus, all presented intervals have bandwidths greater than α and less than β. So now, as in the original proof, FirstFit uses kn colors. Although, we are not able to color the instance with at most $k + 1$ colors. Instead, we may use at most k colors for short intervals and at most $\frac{kn}{\lfloor 1/\mu\rfloor-1}$ for long ones (take small enough ϵ). Thus, the competitive ratio is at least $\frac{(\lceil\sigma\rceil+1)(\lfloor 1/\mu\rfloor-1)}{\lceil\sigma\rceil+\lfloor 1/\mu\rfloor}$. □

This lower bound becomes a constant as σ goes to the infinity, what is a significant difference between the case where all bandwidths are allowed and the case where

Presenter cannot use arbitrarily small or arbitrarily big bandwidths.

$$\lim_{\sigma\to\infty} \frac{(\lceil\sigma\rceil+1)\Big(\Big\lfloor\frac{1}{\mu}\Big\rfloor-1\Big)}{\lceil\sigma\rceil+\Big\lfloor\frac{1}{\mu}\Big\rfloor} = \Big\lfloor\frac{1}{\mu}\Big\rfloor$$

It is easy to check that the function $f(x)=\frac{(x+1)(k-1)}{x+k}$ is a non-decreasing function of x since $\frac{d}{dx}f(x)=\left(\frac{k-1}{x+k}\right)^2\geqslant 0$. Thus we have the following corollary.

Corollary 4.16. *For every $\alpha\in\left(0,\frac{1}{2}\right)$, and $\beta\in\left(\frac{1}{2},1\right)$, both absolute and asymptotic competitive ratios of the FirstFit algorithm in the online interval coloring with $[\alpha,\beta]$-bandwidths are at least $\left\lfloor\frac{1}{\max\{\alpha,1-\beta\}}\right\rfloor$.*

4.4 FirstFit Online Unit Interval Coloring with Bandwidths.

In this section we analyze the performance of the FirstFit algorithm in the game played on the unit interval graph (i.e. with $\sigma=1$). Corollary 4.7 and the discussion after it yield an absolute upper bound of 7 and asymptotic upper bound of 6 on the competitive ratio of the FirstFit algorithm in this case. In Theorem 4.18 we show that in fact asymptotic bound can be improved to 5. To prove this, we need an obvious observation.

Observation 4.17. *Let $\mathcal{I}=[I_1,\ldots,I_n]$ be a sequence of intervals, originally colored by FirstFit with the colors $col_{\mathcal{I}}(I_i)\leqslant k$ for some k. For each color i let $\mathcal{I}_i$ be the set of all intervals that FirstFit colored with i. Then for each modification of $\mathcal{I}$ to $\mathcal{J}=[\mathcal{I}_1,\ldots,\mathcal{I}_k]$ we have $col_{\mathcal{J}}(I_i)=col_{\mathcal{I}}(I_i)$ independently of the order of presentation inside the $\mathcal{I}_i$'s.*

Theorem 4.18. *The maximal color used by the FirstFit algorithm in the online unit interval coloring with bandwidths is bounded from above by $5OPT+2$.*

Proof. If all intervals have bandwidths greater than $\frac{1}{2}$, then the online unit interval coloring with bandwidths reduces to the online unit interval coloring problem, and in that case we know that FirstFit is 2-competitive. Thus, we assume that there is at least one interval with bandwidth at most $\frac{1}{2}$.

Our proof here is modeled after the proof of Lemma 4.6. Analogously to that lemma, we split all neighbors of I into color classes, and denote I_i the representative of the color class $\mathcal{I}_i$. Let I_a be a representative with the highest color and bandwidth not greater than $\frac{1}{2}$. Representatives with colors higher than a call *heavy representatives*. Let x_L be the number of heavy representatives that intersect the left endpoint of I, and x_R be the number of heavy representatives that intersect only right endpoint of I (prevent double counting). Clearly, $col(I)=col(I_a)+x_L+x_R+1$.

Without loss of generality, assume that I_a intersects the left endpoint of I. According to Observation 4.17, we may safely assume that intervals are presented in the order of their colors. Thus, when the interval I_a is presented, there are no heavy representatives yet, and so the sum of bandwidths of all intervals that I_a intersects is not greater than $2\omega^*-\frac{1}{2}x_L$. That is because each heavy representative that intersects the left endpoint of I intersects one endpoint of I_a, the bandwidth of a heavy representative is at

least $\frac{1}{2}$, and the sum of bandwidths at any point cannot exceed ω^*. The bandwidth of the interval I_a is at most $\frac{1}{2}$. Hence, Observation 4.4 implies that $col(I_a) \leqslant \lfloor 4\omega^* - x_L \rfloor + 1$, and so $col(I) \leqslant 4\omega^* + x_R + 2$.

There are x_L intervals intersecting the left endpoint of I and each of them has bandwidth strictly greater than $\frac{1}{2}$. Thus, one has to use at least x_L colors to properly color the presented set of intervals. Analogous argument applies to the right endpoint of I. Hence, the minimal number of colors required in the proper coloring is at least $\max\{\lceil \omega^* \rceil, x_L, x_R\}$.

Case 1: $\lceil \omega^* \rceil \geqslant \max\{x_L, x_R\} \Rightarrow col(I) \leqslant 4\omega^* + x_R + 2 \leqslant (4+1)\lceil \omega^* \rceil + 2 \leqslant 5OPT + 2$.
Case 2: $x_L \geqslant \max\{\lceil \omega^* \rceil, x_R\} \Rightarrow col(I) \leqslant 4\omega^* + x_R + 2 \leqslant (4+1)x_L + 2 \leqslant 5OPT + 2$.
Case 3: $x_R \geqslant \max\{\lceil \omega^* \rceil, x_L\} \Rightarrow col(I) \leqslant 4\omega^* + x_R + 2 \leqslant (4+1)x_R + 2 \leqslant 5OPT + 2$. □

For a full analysis of the FirstFit algorithm on unit intervals with bandwidths from $[0, 1]$ we need also a lower bound. This bound will follow directly from our general result (presented in Section 4.5) for bandwidths in $[0, \beta]$. Using Theorem 4.32 we get the following corollary.

Corollary 4.19. *Both absolute and asymptotic competitive ratios of the FirstFit algorithm in the online unit interval coloring with bandwidths are at least* 3.38.

Corollary 4.20. *FirstFit is not an optimal algorithm for the online unit interval coloring with bandwidth.*

Proof. As we mentioned in Chapter 2, Epstein and Levy [12] designed a 3.178-competitive Algorithm for the online unit interval coloring with bandwidths. □

4.5 FirstFit Online Unit Interval Coloring with $[0, \beta]$-Bandwidths.

Our construction for the lower bound on the competitive ratio of the FirstFit algorithm in the online unit interval coloring with $[0, \beta]$-bandwidths is a combination of two well know results: an interval construction presented by Epstein and Levy [13], and a lower bound for the online bin-packing problem presented by Johnson et al. [21, 20]. In fact, we prove that the lower bound in the online unit interval coloring with bandwidths is twice as big as the lower bound in the online bin-packing. Below, we present a modification of the construction from [13]. For a positive integer n and positive even integer m, both to be determined later, consider an infinite set of intervals $\mathcal{I}_{nm} = \{I_{i,j} : i \in [nm], j \in \mathbb{Z}\}$ where $I_{i,j} = [P(i,j), P(i,j) + 1]$, with

$$P(i,j) = \begin{cases} 2j - \frac{i}{nm}, & \text{when } i \text{ is odd} \\ 2j - \frac{i}{nm} + 1, & \text{when } i \text{ is even,} \end{cases}$$

see Figure 4.3. We call the set $L_i = \{I_{i,j} : j \in \mathbb{Z}\}$ the i-th layer of $\mathcal{I}_{nm}$. We claim that $\mathcal{I}_{nm}$ has the following properties.

Observation 4.21. *The size of the maximum clique in* $L_1 \cup \ldots \cup L_{2i-1}$ *is* i.

Proof. It is easy to check that the point $2j - \frac{1}{nm}$ belongs to the intervals $I_{1,j}, I_{3,j}, \ldots, I_{2i-1,j}$. Thus, there is a clique of size i. Now assume that there is a point p contained in $i + 1$

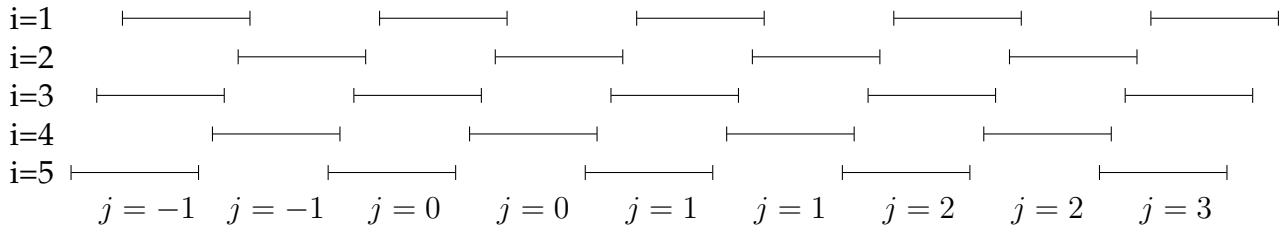

Figure 4.3: A part of the infinite set of intervals $\mathcal{I}_{nm}$.

intervals. Consider the set of intervals from two consecutive layers L_{2k} and L_{2k+1}. It is easy to check that the only regions where those intervals form a clique of size 2 are of the form $\left[2j - \frac{2k+1}{nm}, 2j - \frac{2k}{nm}\right]$. Those regions are disjoint for $k' \neq k$, and so the maximum clique formed by all layers except the first one is of size at most i. Hence, p has to lie in some interval from the first layer. But, intervals of the form $\left[2j - \frac{2k+1}{nm}, 2j - \frac{2k}{nm}\right]$ have an empty intersection with L_1. Thus, the maximum clique in this case is i (you can pick at most one interval from layers L_{2k} and L_{2k+1} and one interval from the first layer). □

Observation 4.22. *Each interval $I_{i,j}$ intersects at least one interval from each layer L_1 up to L_{i-1}.*

Proof. It is easy to see that if i is odd, then the left endpoint of the interval $I_{i,j}$ intersect intervals $I_{2,j-1}, I_{4,j-1}, \ldots, I_{i-1,j-1}$, while the right endpoint of $I_{i,j}$ intersects intervals $I_{1,j}, I_{3,j}, \ldots, I_{i-2,j}$. Analogously, if i is even, then the left endpoint of the interval $I_{i,j}$ intersect intervals $I_{1,j}, I_{3,j}, \ldots, I_{i-1,j}$, while the right endpoint of $I_{i,j}$ intersects intervals $I_{2,j}, I_{4,j}, \ldots, I_{i-2,j}$. □

Observation 4.23. *$I_{i,j} \cap I_{i'j'} \neq \emptyset$ if and only if $I_{i+2,j} \cap I_{i'+2,j'} \neq \emptyset$.*

Proof. Without loss of generality, assume that the left endpoint of $I_{i,j}$ is to the left of the left endpoint of $I_{i',j'}$. Thus, if those intervals intersect, then

$$P(i,j) \leqslant P(i',j') \leqslant P(i,j) + 1$$

$$P(i,j) - \frac{2}{nm} \leqslant P(i',j') - \frac{2}{nm} \leqslant P(i,j) + 1 - \frac{2}{nm}$$

$$P(i+2,j) \leqslant P(i'+2,j') \leqslant P(i+2,j) + 1$$

Hence, the intersection of $I_{i+2,j}$ and $I_{i'+2,j'}$ is nonempty. The proof of the second implication goes the same way. □

Now, we describe our strategy for Presenter against the FirstFit algorithm in the online unit interval coloring with $[0,\beta]$-bandwidths. The strategy consists of n phases and each phase consists of m subphases. Let $k_1, \ldots, k_n$ be a weakly decreasing sequence of natural numbers satisfying $\sum_{i=1}^{n} \frac{1}{k_i+1} < 1$, and put $K := \prod_{i=1}^{k} k_i$. The k_i's are going to control the bandwidths of presented intervals. Now, let ϵ be the smaller of the numbers $\frac{1}{n}\left(1 - \sum_{i=1}^{n} \frac{1}{k_i+1}\right)$ and $\frac{1}{k_1(k_1+1)}$. In the a-th phase Presenter introduces intervals with a bandwidth $w_a = \frac{1}{k_a+1} + \epsilon$ each. For the b-th subphase of the a-th phase let $i_b^a :=$

$(a-1)m+b$ and $j_b^a := \left\lfloor \frac{1}{2}(i_b^a - 1) \right\rfloor$. Presenter in this subphase introduces $nm - i_b^a + 1$ cliques, each containing exactly K intervals with the same endpoints. The first clique consists of K intervals $I_{i_b^a, j_b^a}$, the second one contains K intervals $I_{i_b^a, j_b^a+1}$ and so on up to the last clique with K intervals $I_{i_b^a, j_b^a+nm-i_b^a}$, see Figure 4.4. This finishes the description of our strategy for given integers n, m and a sequence $k_1, \ldots, k_n$.

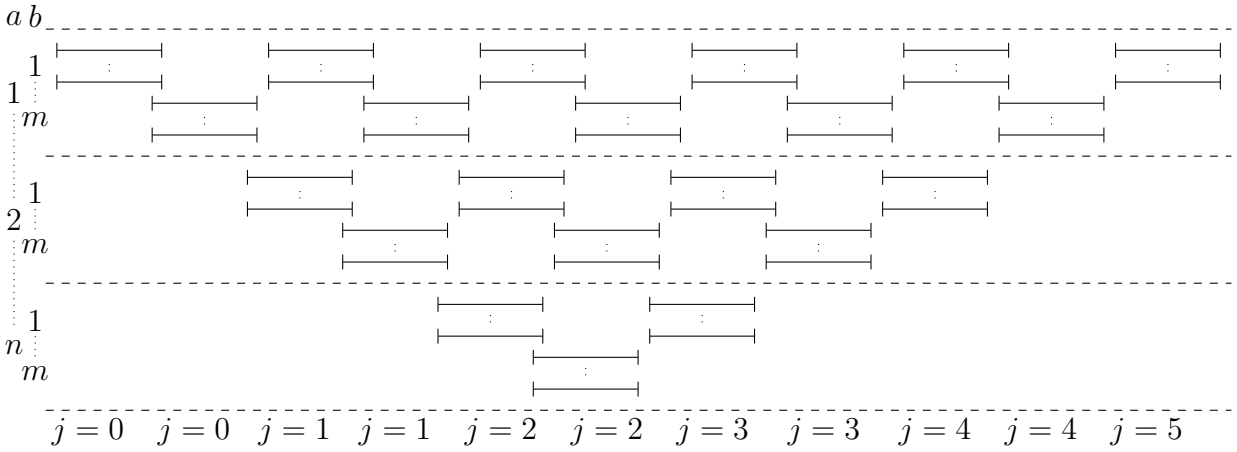

Figure 4.4: Intervals presented in subphases.

Note that, each interval I can be uniquely determined by the tuple $\langle a, b, i, j, k \rangle$ where the pair $\langle a, b \rangle \in [n] \times [m]$ identifies the subphase I was introduced in, $\langle i, j \rangle$ determines the endpoints of I, and $k \in [K]$ distinguishes between intervals belonging to the same clique. Actually, the coordinate i is redundant, since $i = (a-1)m+b$. Thus, we use the tuples $\langle a, b, j, k \rangle$ instead.

Now we count colors used by FirstFit to color the presented set of intervals. We start with an obvious remark on the behavior of FirstFit on intervals from the same subphase.

Remark 4.24. *All cliques introduced in the same subphase are colored by FirstFit in the very same way.*

Due to Remark 4.24 we may restrict to only one clique in each subphase.

Observation 4.25. *In each subphase of the a-th phase FirstFit uses only new colors, exactly $\frac{K}{k_a}$ of them, and in each clique each such color covers intervals with total bandwidth $\frac{k_a}{k_a+1} + k_a\epsilon$.*

Proof. Looking at the intervals from a single clique of the very first subphase, we see that FirstFit can (and actually has to) use $\frac{K}{k_1}$ colors. Moreover, each of these colors has to be used exactly k_1 times, covering $k_1 w_1 = \frac{k_1}{k_1+1} + k_1\epsilon$ total bandwidth. A slightly more careful analysis is needed to observe that the colors used in the first q subphases cannot be used anymore. This is because all intervals introduced later intersect at least one clique from all previous subphases due to Observation 4.22; the sequence of bandwidths $w_1, \ldots, w_n$ is not decreasing; and by induction we know that each already used color covers $k_a w_a = \frac{k_a}{k_a+1} + k_a\epsilon$ total bandwidth, where a is the number of phase in which this particular color was introduced. □

Summing up Observations 4.24 and 4.25 we know that FirstFit used $mK\sum_{i=1}^{n}\frac{1}{k_i}$ colors in total. In order to estimate the offline optimum, we color the intervals by $\left(\frac{1}{2}m+1\right)K$ colors. First, we consider the set $\mathcal{I}^*$ of intervals represented by tuples $\langle 1,b,j,1\rangle$ for all j and $b\in[m]$. Due to Observation 4.21, the maximum clique formed by these intervals is $\frac{1}{2}m+1$. Thus, ignoring bandwidths, all intervals from $\mathcal{I}^*$ can be colored offline by $\frac{1}{2}m+1$ colors. Let $C_{b,j}$ be the color of the interval represented by $\langle 1,b,j,1\rangle$ in that coloring. Now, we expand this particular coloring to some proper coloring of the presented intervals (with bandwidths). We color the interval represented by the tuple $\langle a,b,j,k\rangle$ by $\langle C_{b,j},k\rangle$. Clearly, this coloring uses $\left(\frac{1}{2}m+1\right)K$ colors. In Observation 4.27 we actually prove that our coloring is a proper coloring.

Observation 4.26. *If two intervals from the same phase have the same color, then they are disjoint.*

Proof. Assume to the contrary that intervals I_1 and I_2 were presented in the same phase, $I_1\cap I_2\neq\emptyset$, I_1 was colored by $\langle C_{b_1,j_1},k\rangle$, and I_2 was colored by $\langle C_{b_2,j_2},k\rangle$, such that $C_{b_1,j_1}=C_{b_2,j_2}$. Both intervals were presented in the a-th phase for some $a\in[n]$, so they are of the form $I_1=I_{(a-1)m+b_1,j_1}$ and $I_2=I_{(a-1)m+b_2,j_2}$. We assumed that m is an even integer, so after applying the Observation 4.23, $(a-1)m$ times, we conclude that intervals I_{b_1,j_1} and I_{b_2,j_2} have a nonempty intersection. Hence, the coloring of intervals from the set $\mathcal{I}^*$ is not a proper coloring and we reached a contradiction. □

Observation 4.27. *For every point p and every color c, the sum of bandwidths of intervals containing p and colored by c does not exceed* 1.

Proof. Let $\mathcal{I}$ be the set of intervals containing p and are colored by c. Assume to the contrary, that the sum of bandwidths of these intervals exceed 1. All intervals in the i-th phase have bandwidths $w_i=\frac{1}{k_i+1}+\epsilon$, where $\epsilon\leqslant\frac{1}{n}\left(1-\sum_{i=1}^{n}\frac{1}{k_i+1}\right)$. If the set $\mathcal{I}$ contains at most one interval from each phase, then the sum of bandwidths of all intervals in $\mathcal{I}$ is not greater than $\sum_{i=1}^{n}\left(\frac{1}{k_1+1}+\epsilon\right)=\sum_{i=1}^{n}\frac{1}{k_i+1}+n\epsilon\leqslant 1$. Thus, the set $\mathcal{I}$ contains at least two intervals that were presented in the same phase. Both of them contain p, so we have two intersecting intervals from the same phase colored by the same color. Thus, Observation 4.26 leads to a contradiction. □

The presented coloring is a proper coloring, and the above construction implies a lower bound on the competitive ratio of form

$$\frac{mK\sum_{i=1}^{n}\frac{1}{k_i}}{\left(\frac{1}{2}m+1\right)K}=\frac{2m}{m+2}\sum_{i=1}^{n}\frac{1}{k_i}$$

We assumed that m is some positive even integer. Hence, taking large values of m brings us arbitrarily close to

$$2\sum_{i=1}^{n}\frac{1}{k_i}$$

In fact, is is enough to take $m\geqslant\frac{2(1-\delta)}{\delta}$ to prove a bound of form $(1-\delta)2\sum_{i=1}^{n}\frac{1}{k_i}$. Now, we use the presented construction to prove several lower bounds on the competitive ratio of the FirstFit algorithm in the online unit interval coloring with $[0,\beta]$-bandwidths. In order to do that, we present a series of sequences that satisfy assumptions of the presented construction.

We are mostly interested in a sequence that was widely applied in many papers on the online bin packing problem [5, 15, 29, 41, 42]. This sequence, mostly called a *Saltzer sequence*, was introduced by Sylvester in 1880 [38]. Below we present a definition of this sequence in a parametric case.

Definition 4.28 (Sylvester sequence)**.** *For a positive integer r a sequence $m_{1,r}, m_{2,r}, \ldots, m_{n,r}$ is defined as follows:*

- $m_{1,r} = r + 1$
- $m_{2,r} = r + 2$
- $m_{j,r} = m_{j-1,r}(m_{j-1,r} - 1) + 1$ *for* $j \in \{3, \ldots, n\}$

Observation 4.29. *For every positive integers r and k, $\frac{r}{m_{1,r}} + \sum_{i=2}^{k} \frac{1}{m_{i,r}} = 1 - \frac{1}{m_{k+1,r}-1}$.*

Proof. By induction on k. For $k = 1$ we have $\frac{r}{r+1} = 1 - \frac{1}{r+1} = 1 - \frac{1}{m_{2,r}-1}$. For $k = 2$, $\frac{r}{r+1} + \frac{1}{r+2} = 1 - \frac{1}{r+1} + \frac{1}{r+2} = 1 - \frac{1}{(r+1)(r+2)} = 1 - \frac{1}{m_{3,r}-1}$. Assume that this observation is true for k. Then $\frac{r}{m_{1,r}} + \sum_{i=1}^{k+1} \frac{1}{m_{i,r}} = 1 - \frac{1}{m_{k+1,r}-1} + \frac{1}{m_{k+1,r}} = 1 - \frac{1}{m_{k+1,r}\left(m_{k+1,r}-1\right)} = 1 - \frac{1}{m_{k+2,r}-1}$. □

We define a sequence of parameters for our lower bound construction as follows. For positive integers r and n, let $k_{1,r}, \ldots, k_{n+r-1,r}$ be an $(n + r - 1)$-element sequence such that:

- $k_{i,r} = m_{n-i+1,r} - 1$ for $i \in \{1, \ldots, n-1\}$
- $k_{i,r} = m_{1,r} - 1$ for $i \in \{n, \ldots, n + r - 1\}$

Example 4.30. *For $r = 3$ and $n = 5$, the Sylvester sequence is $[4, 5, 21, 421, 176821]$, while the $k_{i,r}$ sequence is $[176820, 420, 20, 4, 3, 3, 3]$.*

Observation 4.31. *The sequence $k_{i,r}$ satisfies $\sum_{i=1}^{n} \frac{1}{k_i+1} < 1$.*

Proof. $k_{i,r}$ is an $(n' + r - 1)$-element sequence based on an n'-element Sylvester sequence. From the definition we have $\sum_{i=1}^{n} \frac{1}{k_i+1} = \sum_{i=1}^{n'-1} \frac{1}{m_{n'-i+1,r}} + r\frac{1}{m_{1,r}} = \frac{r}{m_{1,r}} + \sum_{i=2}^{n'} \frac{1}{m_{i,r}}$. Observation 4.29 implies that the last expression is equal to $1 - \frac{1}{m_{n'+1,r}-1}$, which is less than 1. □

Observation 4.31 implies that we can use sequences $k_{i,r}$ as an input for the lower bound construction. Hence, we have a following theorem.

Theorem 4.32. *For every positive integer $r \geqslant 2$ and every $0 < \delta \leqslant 1 - \frac{1}{r}$, both absolute and asymptotic competitive ratios of the FirstFit algorithm in the online unit interval coloring with $\left[0, \frac{1}{r} + \delta\right]$-bandwidths are at least $2\Big(1 + \frac{1}{r+1} + \frac{1}{(r+1)(r+2)} + \frac{1}{(r+1)(r+2)((r+1)(r+2)+1)} + O\big(\frac{1}{r^8}\big)\Big)$.*

Proof. Use the lower bound construction with a sequence $k_{i,r}$, but in order not to violate bandwidths constraint use $\epsilon := \min\{\epsilon, \delta\}$. □

r	ratio	r	ratio
1	3.3809	6	2.3220
2	2.8461	7	2.2781
3	2.6047	8	2.2446
4	2.4688	9	2.2183
5	2.3820	10	2.1970

Table 4.1: Lower bounds on the competitive ratio of the FirstFit algorithm obtained with a Theorem 4.32.

Table 4.1 shows a list of lower bounds that are a consequence of Theorem 4.32.

Johnson et al. [21] presented an upper bound on the competitive ratio of the FirstFit algorithm in the online bin-packing problem in which all items are not greater than $\frac{1}{m}$ for some integer m. They proved that this ratio is bounded from above by $1 + \frac{1}{m}$. Unfortunately, their argument does not work in the case of the online unit interval coloring with bandwidth (there is no point p in which all intervals intersect). Lemma 4.5 implies that the asymptotic competitive ratio of the FirstFit algorithm int the online unit interval coloring with $[0, \beta]$-bandwidths is at most $\frac{2}{1-\beta}$. The following theorem gives a better bound especially for big values of β.

Theorem 4.33. *The asymptotic competitive ratio of the FirstFit algorithm in the online unit interval coloring with $[0, \beta]$-bandwidths is at most $-2W_{-1}\Big(-\frac{\lceil 1/\beta\rceil - 1}{\lceil 1/\beta\rceil + \beta - 2} exp\Big(-\frac{\lceil 1/\beta\rceil}{\lceil 1/\beta\rceil - 1}\Big)\Big)$, where W is the Lambert function.*

Proof. Let I be the last interval presented by Presenter, and for technical reasons assume that FirstFit colored I with a color $n + 2$. Let $\mathcal{I}_i$ be the set of neighbors of I that are colored with a color i. The bandwidth associated with the interval I is not greater than β. Thus, the sum of bandwidths of all intervals in the set $\mathcal{I}_i$ for $i \leqslant n + 1$ is at least $1 - \beta$. Otherwise, I wouldn't be colored with a color $n + 2$. Hence, each set $\mathcal{I}_i$ contains at least $k := \lceil (1 - \beta)/\beta \rceil = \lceil 1/\beta \rceil - 1$ elements. Lemma 4.5 implies that in the online unit interval coloring with bandwidths, if FirstFit assigns a color c to the interval I, then $c \leqslant \frac{2\omega^*}{1-w(I)} + 1$. This inequality may be rearranged to get a lower bound on the bandwidth of an interval. Thus,

$$\forall I : w(I) \geqslant 1 - \frac{2\omega^*}{col(I) - 1}$$

Now we estimate the sum of bandwidths of all neighbors of I. Clearly, this sum cannot exceed $2\omega^*$. Let $\gamma = \frac{2(\lceil 1/\beta\rceil - 1)}{\lceil 1/\beta\rceil + \beta - 2}$, and assume that $n = \delta\omega^*$ for some $\delta \geqslant \gamma$. We bound the sum of bandwidths of intervals in the set $\mathcal{I}_i$ for $i \leqslant \gamma\omega^* + 1$ by $1 - \beta$, while for sets with higher colors we use a bound $k\big(1 - \frac{2\omega^*}{i-1}\big)$. Thus we have,

$$2\omega^* \geqslant (\gamma\omega^* + 1)(1 - \beta) + \sum_{i=\gamma\omega^*+2}^{\delta\omega^*+1} k\left(1 - \frac{2\omega^*}{i-1}\right)$$

$$2\omega^* \geqslant (\gamma\omega^* + 1)(1 - \beta) + k(\delta - \gamma)\omega^* - 2k\omega^*(H_{\delta\omega^*} - H_{\gamma\omega^*})$$

where H_j is the j-th harmonic number ($H_j = \sum_{i=1}^{j} \frac{1}{j}$). We bound harmonics by a natural logarithm $H_b - H_a \leqslant \int_{a+1}^{b+1} \frac{dx}{x-1} = \ln \frac{b}{a}$, and remove some positive constants from the right hand side. Hence,

$$2 \geqslant \gamma(1-\beta) + k(\delta - \gamma) - 2k \ln \frac{\delta}{\gamma}$$

$$\ln \frac{\delta}{\gamma} \geqslant \frac{1}{2}\delta + \frac{\gamma}{2k}(1-\beta) - \frac{1}{2}\gamma - \frac{1}{k}$$

The left hand side of the above inequality is a logarithm of δ, while the right hand side is a linear function of δ. Thus, there is a maximum δ for which this inequality is satisfied, and hence, the interval I received a color that is not greater than $\delta^*\omega^* + 2$. In order to find an explicit value of δ^*, we need to solve an equation of form $\ln(ax) = bx + c$. Thus,

$$\ln(ax) = bx + c \Leftrightarrow ax = e^{bx}e^c \Leftrightarrow \frac{e^c}{ax} = e^{-bx} \Leftrightarrow -\frac{b}{a}e^c = -bxe^{-bx}$$

From the definition of the *Lambert W function* we get $-bx = W\left(-\frac{b}{a}e^c\right)$, and so $x = -\frac{1}{b}W\left(-\frac{b}{a}e^c\right)$. To get the final form of δ^* we put $a = \frac{1}{\gamma}$, $b = \frac{1}{2}$ and $c = \frac{\gamma}{2k}(1-\beta) - \frac{1}{2}\gamma - \frac{1}{k}$. Hence,

$$\delta^* = -2W_{-1}\left(-\frac{\lceil 1/\beta \rceil - 1}{\lceil 1/\beta \rceil + \beta - 2} exp\left(-\frac{\lceil 1/\beta \rceil}{\lceil 1/\beta \rceil - 1}\right)\right)$$

It is easy to check that the argument is negative, so because we are interested in those values of δ that are greater than 1, we take the branch -1 of the Lambert W function. □

Figure 4.5 illustrates the best known lower and upper bounds on the competitive ratio of the FirstFit algorithm in the online unit interval coloring with $[0, \beta]$-bandwidths. For small values of β the analysis is tight, while there is still some room for improvement for larger values of β.

4.6 Open problems

In this chapter we presented a series of bounds on the competitive ratio of the FirstFit algorithm, but there are still some unanswered question. The analysis for the case in which all bandwidths are allowed is very tight. We proved a linear in terms of σ upper bound which matches the lower bound in this problem up to a sub-linear function of σ. Although, in the case where Presenter cannot use small bandwidths or big ones, there is quite a big gap between a constant lower bound and a logarithmic in terms of σ upper bound. Another interesting question is whether the competitive ratio of the FirstFit algorithm in the online unit interval coloring with $[0, \beta]$-bandwidths is a continuous function of β. In the online bin-packing problem Johnson et al. [21] proved that this function is in fact a step function, and hence not continuous. Both our bounds are not continuous (the points of incontinuity are of form $\frac{1}{r}$ for $r \in \mathbb{N}_+$), but the results we proved do not exclude a continuous solution for this question. However, around points of form $\frac{1}{r}$, the difference between upper bound and lower bound is very small. For instance, the best lower bound for $\beta = \frac{1}{2} + \epsilon$ is 3.38, while the best upper bound for $\frac{1}{2} - \epsilon$ is 3.60.

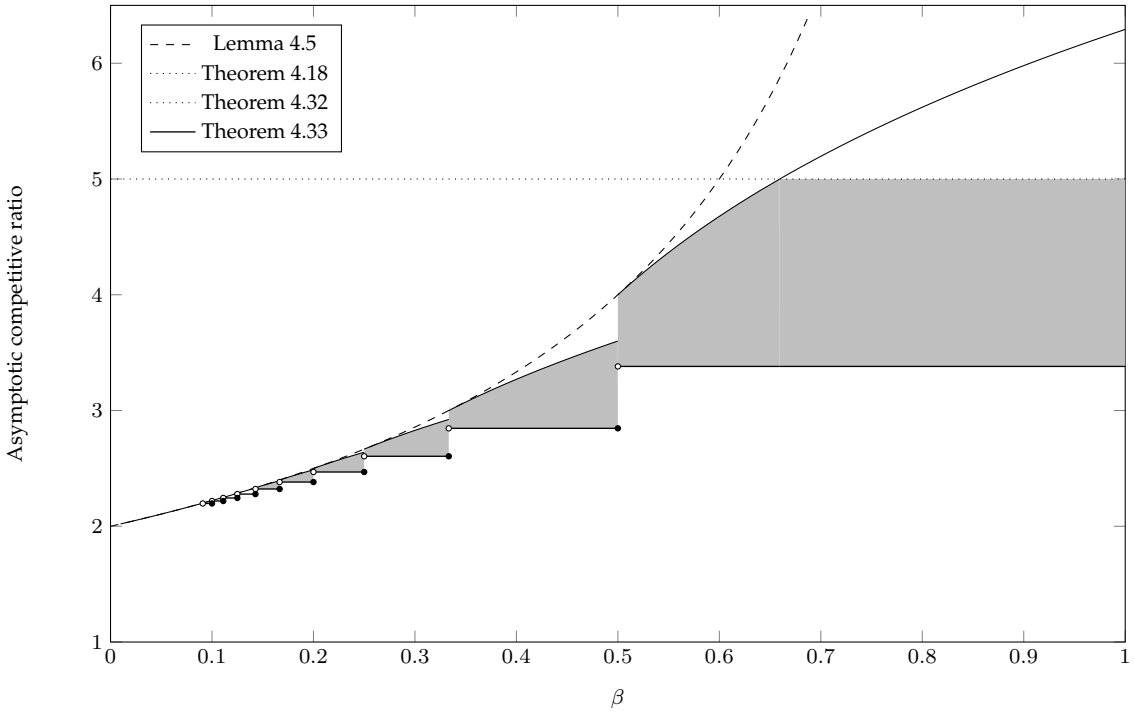

Figure 4.5: Asymptotic competitive ratio of the FirstFit algorithm in online unit interval coloring with $[0, \beta]$-bandwidths.

5

Efficient Enumeration of Non-isomorphic Interval Graphs

5.1 Introduction

Graph enumeration problems, besides their theoretical value, are of interest not only for computer scientists, but also to other fields, such as physics, chemistry, or biology. Enumeration is helpful when we want to verify some hypothesis on a quite big set of different instances, or find a small counterexample. For graphs it is natural to say that two graphs are "different" if they are non-isomorphic. Many papers dealing with the problem of enumeration were published for certain graph classes, see [27, 37, 36, 43]. A series of potential applications in molecular biology, DNA sequencing, network multiplexing, resource allocation, job scheduling, and many other problems, makes *interval graphs* a particularly interesting class of graphs. In this chapter, we present an efficient algorithm that for a given positive integer n lists all non-isomorphic interval graphs on n vertices. It is well-known that the number of such graphs is roughly n^{nc} for some constant c, see [1, 18, 19]. For that reason, we measure the efficiency of the enumeration algorithm by the worst-case time delay between output of any two successive graphs.

5.1.1 Previous work

Yamazaki et al. [43] presented an enumeration algorithm for non-isomorphic interval graphs that works with the worst-case time delay $O(n^6)$, and recently [44] improved it to $O(n^4)$. Their algorithm is based on the *reverse search method*, invented by Avis and Fukuda [3], and in its general form works in the following way. Let $\mathcal{C}$ be a family of graphs we want to enumerate, and let $G_1, \ldots, G_k \in \mathcal{C}$ be some special graphs called *roots*. We define a *family forest* $\mathcal{F}$ which spans $\mathcal{C}$, and consists of k rooted trees $\mathcal{T}_1, \ldots, \mathcal{T}_k$ such that the root of $\mathcal{T}_i$ is G_i. For graphs that are not roots, let $par : \mathcal{C} \setminus \{G_1, \ldots, G_k\} \to \mathcal{C}$ be the *parent function*. In order to enumerate all graphs in the family $\mathcal{C}$, we independently enumerate all graphs in each tree $\mathcal{T}_i$. To enumerate all graphs in the tree $\mathcal{T}_i$ we use any tree traversal algorithm like BFS or DFS. The most time consuming operation in the tree traversal is computing the children of a graph G. From the definition, children of G are those graphs $G' \in \mathcal{C}$ whose parent is G. Hence, if we want to design a fast enumeration algorithm that uses this technique, we need to carefully define the parent function. Authors in [44] used the fact that for every interval graph $G = (V, E)$ that is not a complete graph, there is at least one edge $e \notin E$ such that the graph $G_e = (V, E \cup \{e\})$ is also an interval graph. They defined the only root G_1 to be a com-

plete graph on n vertices, and $par(G) = G_e$, where the edge e is uniquely defined - for more details see [44]. The consequence of this approach is the fact that for every graph G there are at most $|E(G)|$ candidates for the children of G in $\mathcal{F}$, and so this number of potential children does not depend on the size of the enumerated family. Moreover, the authors observed that in order to enumerate only non-isomorphic graphs, it is enough to filter out isomorphic copies from the set of children. Isomorphism test in the class of interval graphs is an easy problem thanks to *MPQ*-trees, see Section 5.2. Hence, to compute the set of children for a graph $G = (V, E)$, the authors consider all graphs $G_e = (V, E \setminus \{e\})$. For each of them, they check whether G_e is an interval graph using some linear time recognition algorithm. Then, if G_e is an interval graph, they check whether $G = par(G_e)$, and build a corresponding *MPQ*-tree. Finally, they store a set of children trees, effectively removing all duplicates.

5.1.2 Our results

We revisit the work of Yamazaki et al. and show how to modify their enumeration algorithm to reduce the worst-case time delay between the output of two successive graphs from $O(n^4)$ to $O(n^3 \log n)$. Our key observation is the fact that having an *MPQ*-tree corresponding to a graph $G = (V, E)$ we are able to list all edges e such that a graph $G_e = (V, E \setminus \{e\})$ is an interval graph. Moreover, for each such edge we show how to build an *MPQ*-tree corresponding to the graph G_e without constructing it explicitly.

Organization of this chapter

In the next section we introduce concepts and definitions that are widely used in this chapter, and also provide a detailed description of *MPQ*-trees along with their most important properties. In Section 5.3, we present a total ordering over all *MPQ*-trees, define a canonical *MPQ*-tree using this ordering, and also present a fast algorithm that for a given *MPQ*-tree $\mathcal{T}$ computes its canonical form $\mathcal{T}'$. In Section 5.4 we consider an *MPQ*-tree $\mathcal{T}$ corresponding to an interval graph $G = (V, E)$, and characterize edges e such that the graph $G_e = (V, E \setminus \{e\})$ is also an interval graph. Moreover, for every edge e we either show a linear time algorithm that produces a string representing G_e if it is an interval graph, or show an induced chordless cycle on four vertices or an asteroidal triple in G_e that certifies that G_e is not an interval graph. In Section 5.5 we develop data structures and algorithms that make use of combinatorial characterization from Section 5.4 and present a fast algorithm which for a given *MPQ*-tree lists all edges e such that G_e is an interval graph. Finally, in Section 5.6 we show how to combine all parts together and build the graph enumeration algorithm. We also show the worst-case performance analysis of our algorithm in this section. The last section contains a discussion of some implementation heuristics that do not change the worst-case analysis, but significantly speedup the execution.

5.2 Preliminaries

Graph notation. For a graph $G = (V, E)$ and a pair of vertices $x, y \in V$, we denote $G + (x, y)$ a graph $G' = (V, E \cup \{(x, y)\})$, and $G - (x, y)$ a graph $G' = (V, E \setminus \{(x, y)\})$.

Interval edge. For an interval graph G, we say that an edge $(x, y) \in E(G)$ is an *interval edge* if $G - (x, y)$ is also an interval graph.

String representation. A sequence $\mathcal{S}$ of length $2n$ is called a *string representation* if each element $z \in [n]$ appears exactly two times in $\mathcal{S}$. Note that a string representation $\mathcal{S}$ encodes an interval graph in a natural way. For every $z \in [n]$ let $first(z)$ denote the index of the first appearance of z in $\mathcal{S}$, and $second(z)$ denote the second one. Then z is represented by an interval $I_z = [first(z), second(z)]$.

5.2.1 *PQ*-trees

It is easy to notice that an interval graph can have many different interval representations. Booth and Lueker in [30] introduced a data structure, called a *PQ*-tree, which encodes all normalized interval representations of an interval graph. A *PQ*-tree is a rooted labeled plane tree composed of leaves and two kinds of internal nodes called *P*-nodes, and *Q*-nodes respectively. The left to right ordering of the leaves of a *PQ*-tree T is called the *frontier* of T. We say that T encodes an interval graph G, if each maximal clique of the graph G is stored in exactly one leaf of T, and each vertex $v \in V(G)$ belongs to a consecutive sequence of cliques in the frontier of T. Having a *PQ*-tree T one can obtain another *PQ*-tree T' which is *equivalent* to T using the following two operations: arbitrarily permute the children of a *P*-node, or reverse the order of the children of a *Q*-node. The crucial property of a *PQ*-tree T is the fact that for every permutation σ of maximal cliques of the graph G such that each vertex belongs to a consecutive sequence of cliques, there is a *PQ*-tree T' that is equivalent to T, and frontier of T' represents σ. In other words, each normalized interval representation of the graph G is represented by some tree equivalent to T.

5.2.2 *MPQ*-trees

PQ-trees are quite simple and easy to understand data structure representing interval graphs, but unfortunately they may occupy up to $O(n^2)$ space. To reduce the space consumption, Korte and Möhring [28] presented *modified PQ*-trees called *MPQ*-trees. In an *MPQ*-tree, we do not store maximal cliques in leaves, but we assign to each *P*-node and each child of a *Q*-node a set of vertices in such a way that vertices laying on a path from the root of the tree to some leaf represent a maximal clique in G, see Figure 5.1C for an example. For a *Q*-node Q with children $T_1, \ldots, T_k$, we denote S_i the set of vertices assigned to T_i, and call it the *i-th section* of Q. Note that, each vertex belongs to the consecutive sequence of maximal cliques, so it has to belong to consecutive sequence of sections of a *Q*-node. Hence, in order to limit the used space, we can store the information about the vertex x only in the first and last section it belongs to. Thanks to this modification, an *MPQ*-tree is an $O(n)$ space representation of an interval graph. In this chapter we show several drawings of *MPQ*-trees. We represent *P*-nodes as circles, and *Q*-nodes as rectangles divided into smaller rectangles representing sections of the *Q*-node. For instance, in the Figure 5.1C the root is an empty *P*-node, and the vertex 6 belongs to the sections S_2 and S_3 of the only *Q*-node.

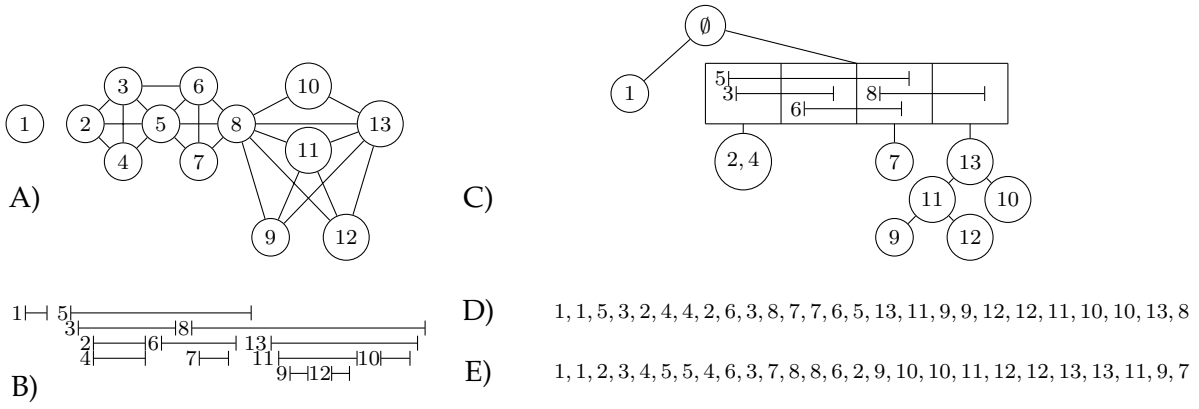

Figure 5.1: A) An interval graph G, B) Its interval representation $\mathcal{I}$, C) Its *MPQ*-tree $\mathcal{T}$, D) String representation $\mathcal{S}$ of tree $\mathcal{T}$, E) Canonical string representation

5.2.3 Known results

During past decades, many researchers published their results on constructing both *PQ*-trees and *MPQ*-trees. Those trees were mostly used to test whether a given graph is an interval graph. Booth and Lueker used *PQ*-trees in their recognition algorithm [30] and proved that for a given graph G the corresponding *PQ*-tree can be computed in $O(n + m)$ time. In [28] Korte and Möhring presented analogous result for *MPQ*-trees. Although, we are most interested in work of Saitoh et al. [39] who presented an algorithm that constructs an *MPQ*-tree for a given interval graph representation and works in $O(n \log n)$ time, or $O(n)$ if the endpoints of intervals are already given in an ascending order.

Theorem 5.1 ([39] Thm.12)**.** *If the graph G is given as an interval representation such that the endpoints are sorted by the coordinates, then there is an algorithm that produces an* MPQ*-tree corresponding to G in $O(n)$ time.*

Clearly, having a string representation of the graph G, we can produce an interval representation satisfying the conditions of Theorem 5.1 in $O(n)$ time. Hence, we have the following corollary.

Corollary 5.2. *There is an algorithm that for a given string representation $\mathcal{S}$ of the graph G builds a corresponding* MPQ*-tree $\mathcal{T}$ in $O(n)$ time.*

Before we proceed to technical definitions and lemmas, we provide some naming conventions we are going to use in the rest of this chapter.

Node. To avoid a confusion when talking about elements of a graph and elements of a tree, we always refer elements of a graph as *vertices* and elements of a tree as *nodes*. For a vertex v of a graph G, we denote $node(v)$ the node of a corresponding *MPQ*-tree $\mathcal{T}$ such that v belongs to the set assigned to that node.

Subtree. For a node with k subtrees $T_1, \ldots, T_k$, we denote V_i the set of all vertices that are assigned to the nodes of a subtree T_i. If $V_i = \emptyset$, then we say that a subtree T_i is empty.

Endpoint. For a *Q*-node we say that a vertex v *has its left endpoint* in a section $S_{l(v)}$, if v belongs to $S_{l(v)}$ and does not belong to any other section S_b with $b < l(v)$. Analogously, we say that v *has its right endpoint* in $S_{r(v)}$, if v belongs to $S_{r(v)}$ and does not belong to any other section S_b with $b > r(v)$. Vertex v is *contained* in sections $S_a, \ldots, S_b$, if $a \leqslant l(v) < r(v) \leqslant b$.

String representation. For an *MPQ*-tree $\mathcal{T}$ we define a $2n$-element string $\mathcal{S}$ called *string representation* of $\mathcal{T}$. This string is built recursively over the structure of $\mathcal{T}$. For a *P*-node we first output all vertices that belong to that node, then recursively string representations of the children from left to right, and at the end yet again all vertices that belong to that node, but now in the reversed order. Hence, the string representation for a *P*-node with vertices $[k]$ and no children is $123\ldots(k-1)kk(k-1)\ldots321$. A string representation for a *Q*-node is a concatenation of string representations for its sections. The string for a section S_i starts with vertices that have its left endpoint in S_i, then there is a string for a subtree T_i, and finally vertices that have its right endpoint in S_i. It is easy to see that string representation of $\mathcal{T}$ is also a string representation of the graph corresponding to $\mathcal{T}$.

Normalized string representation of $\mathcal{T}$. Consider a permutation $\sigma : [n] \to [n]$, and a string $\sigma(\mathcal{S})$, which results from the application of σ to each element of $\mathcal{S}$. *Normalized string representation* is the lexicographically smallest string $\sigma(\mathcal{S})$ among all permutations σ.

Finally, we recall some properties of *MPQ*-trees produced by the Algorithm from Theorem 5.1.

Lemma 5.3 ([28, 40])**.** *In the* MPQ*-tree constructed in Theorem 5.1 for every* Q*-node with k children we have:*

a) $V_1 \neq \emptyset$ *and* $V_k \neq \emptyset$,

b) $S_1 \subseteq S_2$ *and* $S_k \subseteq S_{k-1}$,

c) $S_{i-1} \cap S_i \neq \emptyset$, *for* $2 \leqslant i \leqslant k$,

d) $S_{i-1} \neq S_i$, *for* $2 \leqslant i \leqslant k$,

e) $(S_i \cap S_{i+1}) \setminus S_1 \neq \emptyset$ *and* $(S_{i-1} \cap S_i) \setminus S_k \neq \emptyset$, *for* $2 \leqslant i \leqslant k-1$, *and*

f) $(S_{i-1} \cup V_{i-1}) \setminus S_i \neq \emptyset$ *and* $(S_i \cup V_i) \setminus S_{i-1} \neq \emptyset$, *for* $2 \leqslant i \leqslant k$.

Moreover:

g) there are no two empty P*-nodes such that one of them is a parent of the other,*

h) there is no P*-node that has only one child which root is also a* P*-node, and*

i) P*-nodes have no empty children,*

see Figure 5.2.

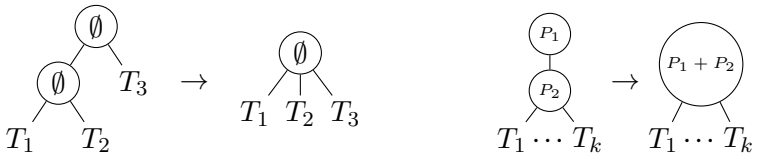

Figure 5.2: *MPQ*-trees do not contain two consecutive empty *P*-nodes (left), or a *P*-node with only one child which root is also a *P*-node (right).

5.3 Canonical *MPQ*-tree

In this section, we define a total ordering $\prec$ over *MPQ*-trees. One may notice that the lexicographical order on string representations is a total ordering on *MPQ*-trees, but for complexity reasons we introduce a different one. Denote by $|\mathcal{T}|$ the number of vertices contained in the tree $\mathcal{T}$, $c(\mathcal{T})$ the number of children of the root of $\mathcal{T}$, and $ex(\mathcal{T})$ the number of vertices that belong to the root of $\mathcal{T}$. We assign a tuple $t_{\mathcal{T}} = \langle |\mathcal{T}|, ex(\mathcal{T}), c(\mathcal{T})\rangle \in \mathbb{N}^3$ to every tree $\mathcal{T}$, and say that $\mathcal{T}_1 \prec \mathcal{T}_2$ if $t_{\mathcal{T}_1}$ is lexicographically smaller than $t_{\mathcal{T}_2}$, or $t_{\mathcal{T}_1} = t_{\mathcal{T}_2}$ and normalized string representation of $\mathcal{T}_1$ is lexicographically not greater than normalized string representation of $\mathcal{T}_2$. We say that *MPQ*-tree $\mathcal{T}$ is *in canonical form*, if for every other tree $\mathcal{T}'$ representing the same graph G we have $\mathcal{T} \prec \mathcal{T}'$. A string $\mathcal{S}$ is a *canonical string*, if it is a normalized string representation of a canonical tree. Observe that if $\mathcal{T}$ is in a canonical form, then all subtrees of $\mathcal{T}$ are in a canonical form. Clearly, if some subtree of $\mathcal{T}$ is not in a canonical form, then we may rotate it and obtain a lexicographically smaller string.

The main feature of our canonization lies in the following theorems.

Theorem 5.4. *Two interval graphs G_1 and G_2 are isomorphic if and only if their canonical strings S_1 and S_2 are equal.*

Theorem 5.5. *There is an algorithm that for every* MPQ*-tree $\mathcal{T}$ computes its canonical form in $O(n \log n)$ time.*

Proof. At the very beginning, we shall compute a function g that for every vertex v will describe its relative position among all vertices from $node(v)$. We compute this function for each *P*-node independently, and for all *Q*-nodes collectively. For a *P*-node with j vertices $z_1, \ldots, z_j$, we assign $g(z_i) = i$. Thus, we can compute the function g for all *P*-nodes in $O(n)$ time. In order to compute this function for all *Q*-nodes, at first we assign a tuple $\langle l(v), r(v)\rangle$ to each vertex v which belongs to some *Q*-node. Then we sort all tuples using radix sort algorithm, and visit vertices in the order determined by their tuples. For each *Q*-node we keep a local counter that starts with 1 and increases each time we visit a vertex from this node. Thus, because all vertices are from the set $[n]$, and each *Q*-node has a linear in terms of n number of sections, we compute this function for all vertices in $O(n)$ time.

We shall construct a function f that assigns an integer $f(\mathcal{T}') > n$ to every subtree $\mathcal{T}'$ of a tree $\mathcal{T}$ in such a way that $\mathcal{T}_1 \prec \mathcal{T}_2 \Leftrightarrow f(\mathcal{T}_1) \leqslant f(\mathcal{T}_2)$. Simultaneously we will rotate subtrees so that they are in canonical form. At first, we compute tuples $t_{\mathcal{T}'}$ for each subtree $\mathcal{T}'$ of a tree $\mathcal{T}$. Clearly, it can be easily done in $O(n)$ time. Then, we sort the tuples lexicographically in $O(n)$ using radix sort algorithm. In the next phases, we

inspect nodes of tree $\mathcal{T}$ that have the same tuple $\langle|\mathcal{T}'|, ex(\mathcal{T}'), c(\mathcal{T}')\rangle$, and we do it from the smallest tuples to the biggest ones. Observe that, when we define the value of the function f for $\mathcal{T}'$, the values for all subtrees of $\mathcal{T}'$ are already computed.

All subtrees of $\mathcal{T}'$ are in a canonical form, so in order to compute a canonical form of $\mathcal{T}'$, we need to determine the order of its children. If the root of $\mathcal{T}'$ is a *P*-node, then we use integers $f(\mathcal{T}_i)$ as keys for children, and sort them in $O(c \log c)$ time, where $c = c(\mathcal{T}')$. If the root of $\mathcal{T}'$ is a *Q*-node Q, then we may leave it in the form it is or reverse it. To decide what to do, we compute a special string representation $\mathcal{S}^*$, which is similar to the string representation, but for each vertex v that belongs to Q we put $g(v)$ instead of v, and instead of inserting the whole string for a subtree $\mathcal{T}_i$, we put a single number $f(\mathcal{T}_i)$. Hence, the produced string has length $2 * ex(\mathcal{T}') + c(\mathcal{T}')$ and is produced in time proportional to its length. We also produce similar string for a rotated node, and if that string is lexicographically smaller than the original one, then we rotate Q.

We have just computed canonical forms for all subtrees with the same tuple. For each of them we produce a special string, and sort those strings lexicographically. Finally, we assign values from the set $\{F + 1, F + 2, \ldots\}$, where F is the maximal number assigned to trees with lexicographically smaller tuples (or $F = n$ if there are no smaller). We assign those numbers according to the computed order giving the same value to the subtrees with the same special string representation, and that finishes the algorithm description.

Now, we prove that the algorithm works in the declared time. As we mentioned before, the computation of the function g is linear in time. The same applies to the computation and sorting for the node tuples. Sorting children of a *P*-node with c children takes $O(c \log c)$. Hence, because all *P*-nodes cannot have more than $O(n)$ children in total, we conclude that sorting children for all *P*-nodes takes no more than $O(n \log n)$ time. The length of a special string for a *Q*-node with j vertices and k sections is $O(j + k)$. Thus, the total processing time for all *Q*-nodes is linear in terms of n.

The only thing we have not counted yet is the time spent on sorting subtrees with the same tuple. Note that for a tuple $\langle s, e, c\rangle$, each special string has length exactly $2e + c$. Let n_{sec} be the number of subtrees having a tuple $\langle s, e, c\rangle$. Sorting process for those subtrees takes no more than $O((e + c)n_{sec} \log n_{sec})$. Thus, all sortings together take $O(\sum_{sec}(e + c)n_{sec} \log n_{sec})$. Note that $n_{sec} \leqslant n$, so we only need to show that $\sum_{sec}(e + c)n_{sec}$ is $O(n)$. But, clearly $\sum_{sec} en_{sec} = n$ since this sum counts vertices in all nodes. Similarly, $\sum_{sec} cn_{sec}$ equals the number of edges in $\mathcal{T}$, and we are done. □

5.4 Classification of Interval Edges

In this section we present a series of lemmas that characterize the interval edges for the interval graph G. Moreover, for each interval edge (x, y), we also present a linear, in terms of n, algorithm that produces a string representation for the interval graph $G - (x, y)$. For an edge (x, y) that is not an interval edge, we prove the existence of an induced chordless cycle on four vertices or an asteroidal triple in $G - (x, y)$. The characterization does not use the mere graph G, but the corresponding *MPQ*-tree $\mathcal{T}$ instead.

First, let us introduce a useful definition. We say that x *is over* y in $\mathcal{T}$, if $node(x)$ is

the lowest common ancestor of $node(x)$ and $node(y)$ in $\mathcal{T}$. Notice that, if x is over y then $(x, y) \in E(G)$. Moreover, if there is an edge (x, y) in the graph G, then x is over y, or y is over x. Now, we make an easy observation on interval edges.

Observation 5.6. *If there are at least two vertices z_1 and z_2 such that both x and y are over z_1 and z_2, and there is no edge (z_1, z_2), then (x, y) is not an interval edge.*

Proof. Vertices x, z_1, y and z_2 in that order form a cycle of length 4. We assumed that there is no edge between z_1 and z_2, so if there is no edge between x and y, then this cycle is chordless in $G - (x, y)$, see Figure 5.3. Hence, $G - (x, y)$ is not a chordal graph and so not an interval graph. □

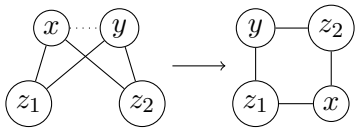

Figure 5.3: An induced C_4 after removing edge (x, y).

The above observation is just one way in showing that the edge (x, y) is not an interval edge. However, in cases when (x, y) is an interval edge, we want to show a linear time algorithm that produces a string representation for a graph $G - (x, y)$. The following lemma, we call the *swapping lemma*, comes handy when we try to produce the mentioned string. It shows when we can swap two consecutive elements in a string representation without adding or removing any edges to the represented graph.

Lemma 5.7. *Let $\mathcal{S}_1$ and $\mathcal{S}_2$ be string representations, such that $\mathcal{S}_2$ is created from $\mathcal{S}_1$ by swapping elements at positions i and $i + 1$ for some i. Denote by a the element at the i-th position in $\mathcal{S}_1$, and by b the element at the $(i + 1)$-th position ($\mathcal{S}_1 = ...ab..$ and $\mathcal{S}_2 = ...ba..$). $\mathcal{S}_1$ and $\mathcal{S}_2$ represent the same interval graph iff both elements at swapped positions represent either left endpoints or right endpoints.*

Proof. Clearly, at most one edge can be added or removed by swapping those two elements. If we swap the left endpoint of a and the right endpoint of b, then we remove an edge (a, b). If we swap the right endpoint of a and the left endpoint of b, then we add an edge (a, b) which is not present in $\mathcal{S}_1$. If both elements represent left endpoints, then right endpoints for both a and b are to the right of $i+1$, hence no edge is added or removed. Similar argument works when both elements represent right endpoints. □

Our aim is to characterize all interval edges in terms of the information provided by an *MPQ*-tree $\mathcal{T}$. Hence, as an input we are given an *MPQ*-tree $\mathcal{T}$ and some edge $(x, y) \in E(G)$. Without loss of generality, we assume that x is over y in $\mathcal{T}$, and we work under this assumption in the following subsections. We split our argument into cases according to the positions of $node(x)$ and $node(y)$ in $\mathcal{T}$.

5.4.1 Both vertices belong to the same *P*-node

Suppose both x and y belong to the same P-node P in $\mathcal{T}$. We show that under this assumption, the edge (x, y) is an interval edge if and only if P is a leaf in $\mathcal{T}$.

Lemma 5.8. *If $node(x) = node(y)$ is a* P*-node that is not a leaf, then (x, y) is not an interval edge.*

Proof. Let P be the considered common P-node, and assume it has j subtrees for some $j \geqslant 1$. If $j \geqslant 2$, then by Lemma 5.3i let $z_1 \in V_1$ and $z_2 \in V_2$. Clearly, there is no edge between z_1 and z_2 and both x and y are over z_1 and z_2. Hence, Observation 5.6 implies that (x, y) is not an interval edge. Thus, P has exactly one subtree, and according to Lemma 5.3h, its root has to be a Q-node, see Figure 5.4. Moreover, Lemma 5.3a implies, that the first and last sections of a Q-node have nonempty subtrees. Let z_1 belong to the first subtree, and z_2 belong to the last one. Yet again, assumptions of the Observation 5.6 applies, so (x, y) is not an interval edge. □

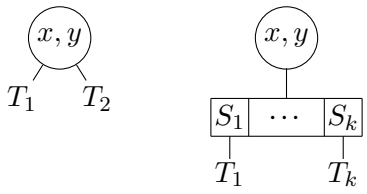

Figure 5.4: Removing an edge from a P-node that is not a leaf leads to an induced C_4 cycle.

Lemma 5.9. *If $node(x) = node(y)$ is a* P*-node that is a leaf, then (x, y) is an interval edge. Moreover, there is a linear time algorithm that produces a string representation for the graph $G - (x, y)$.*

Proof. Without loss of generality assume that $x < y$, and consider the canonical string $\mathcal{S}$ for the *MPQ*-tree $\mathcal{T}$. Clearly, $\mathcal{S}$ is of the form $\mathcal{S} = LAxByC\bar{C}y\bar{B}x\bar{A}R$, see Figure 5.5. In order to remove the edge (x, y) we find the first and the last occurrence of x in $\mathcal{S}$. Then, until x does not occupy two consecutive positions, we swap $\mathcal{S}[i]$ with $\mathcal{S}[i + 1]$, and $\mathcal{S}[j]$ with $\mathcal{S}[j - 1]$, where i denotes the first occurrence of x and j denotes the second. Next, we do the same for y, and as a result we get a string such that both y's are next to each other and are surrounded by both x's, see Figure 5.5c. Finally, we swap the first occurrence of y with the second occurrence of x, effectively removing the edge (x, y). Clearly, this procedure runs in $O(n)$ time, but we have to ensure that it does not add or remove any other edge. Note that, all modifications are performed in a substring of $\mathcal{S}$ that represents a clique, and is of the form $z_1 z_2 \dots z_k z_k \dots z_2 z_1$. Hence, each swapping operation - except the last one - swapped either two left endpoints or two right endpoints. Thus, Lemma 5.7 ensures that no edge was added or removed during this process. Finally, during the last swap x and y occupy four consecutive indexes. Hence, the only affected vertices are x and y.

□

$$\overbrace{}^{\textit{prefix}}\quad\overbrace{}^{\text{P-}\textit{node}}\quad\overbrace{}^{\textit{sufix}}$$

a) $L|AxByC\bar{C}y\bar{B}x\bar{A}|R$
b) $L|AByCxx\bar{C}y\bar{B}\bar{A}|R$
c) $L|ABCxyyx\bar{C}\bar{B}\bar{A}|R$
d) $L|ABCxxyy\bar{C}\bar{B}\bar{A}|R$

Figure 5.5: Removing an edge from a leaf *P*-node.

5.4.2 Both vertices belong to the same *Q*-node

The next case is when both vertices belong to the same Q-node. Here we show that (x, y) is an interval edge if and only if x and y have exactly one common section and the subtree of this section represents a clique (possibly empty). In case this clique is not empty this means that the subtree of the common section consists of a single P-node.

Lemma 5.10. *If $node(x) = node(y)$ is a* Q*-node, then (x, y) is not an interval edge if:*

- *x and y have more than one common section in $node(x) = node(y)$, or*
- *a subtree of the common section does not represent a clique (possibly empty).*

Proof. Assume that there is a common section S_i and its nonempty subtree T_i that does not represent a clique. Hence, there are at least two vertices z_1 and z_2 in V_i such that there is no edge between them, otherwise T_i would represent a clique. Thus, Observation 5.6 implies that (x, y) is not an interval edge.

Now, if there are two common sections S_i and S_j such that both of them have nonempty subtrees T_i and T_j respectively, then we may choose $z_1 \in V_i$ and $z_2 \in V_j$ and use the same argument. This proves that common sections have empty subtrees except at most one which represents a clique.

Finally, assume that there is more than one common section ie. $S_i, S_{i+1}, \ldots, S_{j-1}, S_j$, and without loss of generality S_i has an empty subtree. Lemma 5.3f implies that there is a vertex z_1 for which the section S_i is the last one (z_1 does not belong to sections $S_{i+1}, \ldots, S_j$). Notice that $z_1 \notin \{x, y\}$, otherwise $j = i$. If there is a nonempty subtree T_a for some $i < a \leqslant j$, then we choose $z_2 \in V_a$, and Observation 5.6 leads to an induced chordless cycle. Hence, all common subtrees are empty and Lemma 5.3f gives us a vertex z_2 for which S_j is the first section. Observation 5.6 for vertices x, y, z_1 and z_2 finishes the proof. □

Lemma 5.11. *If $node(x) = node(y)$ is a* Q*-node, x and y have exactly one common section S_i in it, and the subtree T_i represents a clique (possibly empty), then (x, y) is an interval edge. Moreover, there is a linear time algorithm that produces a string representation for the graph $G - (x, y)$.*

Proof. Again, we assume that $x < y$ and consider the canonical string $\mathcal{S}$ for the *MPQ*-tree $\mathcal{T}$, but in this case $\mathcal{S}$ has a more complex form than in the Lemma 5.9. In fact, it is of the form $\mathcal{S} = AxBL_1yL_2V_i\bar{V}_iR_1xR_2CyD$, where $L_1 \cup L_2$ represents the left endpoints of vertices from S_i, $R_1 \cup R_2$ represents the right endpoints of vertices from S_i, and $V_i \cup \bar{V}_i$

represents a clique from the subtree, see Figure 5.6. In order to remove the edge (x, y), at first we need to determine for each element in $\mathcal{S}$ whether it represents the left or the right endpoint. It can be easily done in $O(n)$, since all elements in $\mathcal{S}$ belong to the set $[n]$. The next phase swaps the first occurrence of y with its successor until the next element represents a right endpoint. Analogously, we swap the second occurrence of x with its predecessor until the next element represents a left endpoint. Clearly, because of Lemma 5.7 we did not add or remove any edge till this moment, and $\mathcal{S}$ looks like in Figure 5.6c. Finally, we can remove the edge (x, y) by swapping the first occurrence of y with the second occurrence of x, that in fact occupy consecutive positions in $\mathcal{S}$.

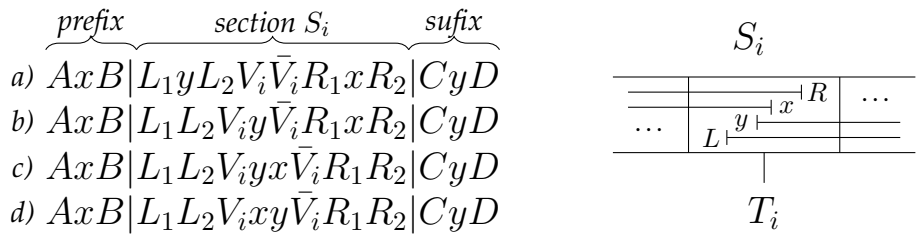

Figure 5.6: Removing an edge from the same Q-node.

□

5.4.3 Vertices in different nodes.

In the previous two subsections, we provided a full classification for the cases where x and y belong to the same node in $\mathcal{T}$. In this subsection, we consider the cases where x and y belong to different nodes in $\mathcal{T}$. Before we present our results in those cases, we introduce a terminology that allows us to describe a path between $node(x)$ and $node(y)$ in $\mathcal{T}$.

For a Q-node with k sections $S_1, \ldots, S_k$, we say that the section S_a is a *central* section if $1 < a < k$. Sections S_1 and S_k are called *non-central* sections. For a vertex v we say that the section S_a is a *v-central* section if $l(v) < a < r(v)$. Sections $S_{l(v)}$ and $S_{r(v)}$ are *v-non-central* sections. For every two vertices x and y, there is exactly one path in the tree $\mathcal{T}$ between $node(x)$ and $node(y)$. We say that this unique path is an *$\langle x, y\rangle$-tree-path* if x is over y in $\mathcal{T}$. For an $\langle x, y\rangle$-tree-path: $node(x) = n_1 - n_2 - \ldots - n_t = node(y)$, we say that this path *goes through a central section* if there is a Q-node n_i for $1 < i < t$ such that n_{i+1} belongs to a subtree of some central section of n_i, see Figure 5.10. Moreover, we say that the path *starts in a central section* if n_1 is a Q-node, and n_2 belongs to a subtree of some x-central section. Analogously, it *starts in a non-central section* if n_2 belongs to a subtree of some x-non-central section. We also say that an $\langle x, y\rangle$-tree-path *starts* in a P-node, if $node(x)$ is a P-node, and *ends* in a P-node if $node(y)$ is a P-node. Analogous definitions apply to Q-nodes. Finally, we say that an $\langle x, y\rangle$-tree-path is *almost rotable* if it does not go through a central section, and ends in a leaf (which obviously as to be a P-node). An $\langle x, y\rangle$-tree-path is *rotable* if it is almost rotable, and either starts in a P-node, or starts in a non-central section. Intuitively, if an $\langle x, y\rangle$-tree-path is almost rotable, then we are able to rotate all the nodes on the path $n_2 - \ldots - n_t$ in such a way that y is the leftmost vertex in the subtree which root is n_2.

Now we are ready to characterize interval edges in the case x and y belong to different nodes in $\mathcal{T}$. We prove that if $\langle x, y\rangle$-tree-path is rotable, then (x, y) is an interval edge. Unfortunately, the reverse implication is not true and sometimes $\langle x, y\rangle$-tree-path is not rotable, but (x, y) is still an interval edge. We shall prove that this happens only for almost rotable $\langle x, y\rangle$-tree-paths satisfying some additional, and quite technical, conditions. First, we show how to compute a string representation for an interval graph $G - (x, y)$ if $\langle x, y\rangle$-tree-path is rotable.

Lemma 5.12. *If an $\langle x, y\rangle$-tree-path is rotable, then (x, y) is an interval edge. Moreover, there is a linear time algorithm that produces a string representation for the graph $G - (x, y)$.*

Proof. In order to remove the edge (x, y), we do not produce a canonical string $\mathcal{S}$ immediately. At first, we need to adjust the tree $\mathcal{T}$ using some preprocessing. If $node(x)$ is a *Q*-node, then we rotate $node(x)$ so that $\langle x, y\rangle$-tree-path starts in the section $S_{l(x)}$. Then, we rotate $\mathcal{T}$ so that the path from $node(x)$ to $node(y)$ goes through the leftmost children of *P*-nodes and the leftmost sections of *Q*-nodes. Let $\mathcal{T}'$ be the result of the described adjustment, and let $\mathcal{S}$ be a string representation of $\mathcal{T}'$. Clearly, $\mathcal{S}$ is of form $LxAByC\bar{C}y\bar{B}DxR$, see Figure 5.7, and both occurrences of y lay in between occurrences of x in $\mathcal{S}$. Node $node(y)$ is a *P*-node that is a leaf, so we start with moving the first occurrence of y to the right and the second occurrence of y to the left until they both meet, as in the Lemma 5.9. All vertices that lay between the first occurrence of x and the first occurrence of y represent the left endpoints. Hence, we can swap the first occurrence of x with its successor until it meets the second occurrence of y. Lemma 5.7 ensures that no edge is added or removed during this process. Finally, moving x once more to the right swaps the left endpoint of x with the right endpoint of y effectively removing the edge (x, y).

$$
\begin{array}{ll}
 & \overbrace{\phantom{LxA|ByC\bar{C}y\bar{B}|}}^{prefix} \overbrace{}^{node(y)} \overbrace{}^{sufix} \\
a) & LxA|ByC\bar{C}y\bar{B}|DxR \\
b) & LxA|BCyy\bar{C}\bar{B}|DxR \\
 & \underbrace{\phantom{LA|BCyxy\bar{C}\bar{B}|}}_{prefix} \underbrace{}_{node(y)+x} \underbrace{}_{sufix} \\
c) & LA|BCyxy\bar{C}\bar{B}|DxR \\
d) & LA|BCyyx\bar{C}\bar{D}|DxR
\end{array}
$$

Figure 5.7: Removing an edge (x, y) in the case where $\langle x, y\rangle$-tree-path is rotable.

□

Now, we want to understand an $\langle x, y\rangle$-tree-path that is not rotable, but is still almost rotable. Such a path has to start in some x-central section of some *Q*-node. We denote $S_1, \ldots, S_k$ the sections of this *Q*-node, and S_a the x-central section where the $\langle x, y\rangle$-tree-path starts. As we already mentioned, sometimes in that case the edge (x, y) might be an interval edge. The next two lemmas establish the required conditions for that to happen.

Lemma 5.13. *Suppose that the $\langle x, y\rangle$-tree-path is almost rotable, starts in a central section S_a, y has no neighbor in a subtree T_a and at least one of the following holds:*

1. $\exists_{1<b\leqslant l(x)} : S_a \smallsetminus \{x\} \subseteq S_b$ *and* $S_{b-1} \cap S_b \subseteq S_a$,
2. $\exists_{r(x)\leqslant b<k} : S_a \smallsetminus \{x\} \subseteq S_b$ *and* $S_b \cap S_{b+1} \subseteq S_a$,
3. $S_a \smallsetminus \{x\} \subseteq S_1$,
4. $S_a \smallsetminus \{x\} \subseteq S_k$.

Then (x, y) is an interval edge. Moreover, there is a linear time algorithm that produces a string representation for the graph $G - (x, y)$.

Proof. Due to some symmetry our arguments for all four cases are very similar, so we show only the proof for the first case. Assume that there is a $1 < b \leqslant l(x)$ such that $S_a \smallsetminus \{x\} \subseteq S_b$ and $S_{b-1} \cap S_b \subseteq S_a$. As in the proof of Lemma 5.12, before we produce the string representation $\mathcal{S}$, we need to make some adjustments in the tree $\mathcal{T}$. At first, we insert a new section $S^* = S_a \smallsetminus \{x\}$ in between sections S_{b-1} and S_b, see Figure 5.8. Clearly, an insertion of a new section does not remove the old edges, but it might add some new ones. However, the section S^* is a subset of an already existing section, so it is not the case. Moreover, the condition that each vertex belongs to the sequence of consecutive sections is preserved. This is because we assumed that $S_{b-1} \cap S_b \subseteq S_a$ and $S_a \smallsetminus \{x\} \subseteq S_b$. Next, we remove the vertex y from the subtree T_a. We also define a subtree T^* of the section S^* to be a single P-node containing y. Note that y has no neighbor in T_a, so no edges were removed, except the edge (x, y). Thus, we obtained a tree $\mathcal{T}'$ that encodes the graph $G - (x, y)$. In order to get the string representation for the graph $G - (x, y)$ it is enough to compute the string representation for $\mathcal{T}'$. This covers the first two cases. For the third one it suffices to insert the section S^* before S_1 and in the fourth case we insert S^* after S_k.

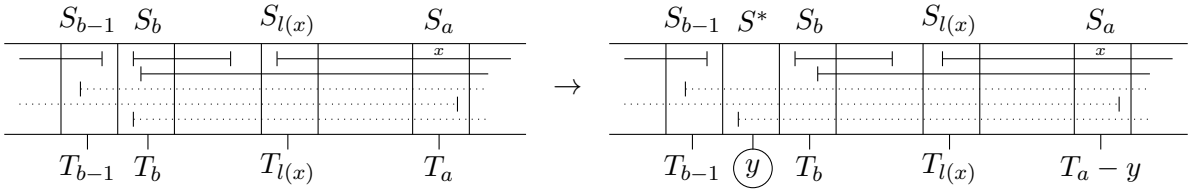

Figure 5.8: Removing an edge (x, y) in the case where $\langle x, y\rangle$-tree-path is almost rotable, starts in some x-central section S_a and y does not have a neighbor in a subtree T_a. (Case 1)

□

Lemma 5.14. *Suppose that the $\langle x, y\rangle$-tree-path is almost rotable, starts in a central section S_a, y has a neighbor in a subtree T_a and at least one of the following holds:*

1. $l(x) > 1$ *and* $S_a \smallsetminus \{x\} \subseteq S_{l(x)}$ *and* $S_{l(x)-1} \cap S_{l(x)} \subseteq S_a$,
2. $r(x) < k$ *and* $S_a \smallsetminus \{x\} \subseteq S_{r(x)}$ *and* $S_{r(x)} \cap S_{r(x)+1} \subseteq S_a$,
3. $l(x) = 1$ *and* $S_a \smallsetminus \{x\} \subseteq S_1$,

4. $r(x) = k$ *and* $S_a \setminus \{x\} \subseteq S_k$.

Then (x, y) *is an interval edge. Moreover, there is a linear time algorithm that produces a string representation for the graph* $G - (x, y)$.

Proof. This lemma is argued in a similar way to Lemma 5.13, but now y has at least one neighbor in T_a, so we cannot simply remove y from T_a. This is also the reason why conditions of this lemma are sharper than in the previous one. Yet again, we are going to prove only the first case, so assume that $S_a \setminus \{x\} \subseteq S_{l(x)}$ and $S_{l(x)-1} \cap S_{l(x)} \subseteq S_a$. Instead of inserting a new section S^*, we move the section S_a to insert it in between sections $S_{l(x)-1}$ and $S_{l(x)}$, see Figure 5.9. Clearly, no edge is added or removed, and the condition that each vertex belongs to the sequence of consecutive sections is preserved. Now, we remove x from the section S_a. This, removes the edge (x, y), but also all the edges (x, v), where $v \in V_a$. In the next phase, we are going to restore those edges. In order to do it, at first we rotate the subtree T_a in such a way that the path from $node(x)$ to $node(y)$ goes through the leftmost children of *P*-nodes and the leftmost sections of *Q*-nodes. Then, we compute the string representation $\mathcal{S}$ of the modified tree, and move the first occurrence of y to the right and the second occurrence of y to the left until both meet, as in the Lemma 5.9. After this operation is done, y is the leftmost vertex of the tree T_a - its right endpoint appears first in the string representation of T_a. In order to restore all the removed edges except (x, y), we move the first occurrence of x in $\mathcal{S}$ to the left, until its predecessor is the second occurrence of y. Clearly, this procedure restores all the removed edges except (x, y).

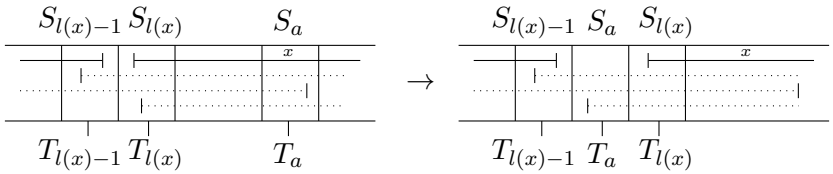

Figure 5.9: Removing an edge (x, y) in the case where $\langle x, y\rangle$-tree-path is almost rotable, starts in some x-central section S_a, and y has some neighbor in a subtree T_a. (Case 1)

□

The above two lemmas cover the only cases where the $\langle x, y\rangle$-tree-path is not rotable, but (x, y) is an interval edge. Now, we are going to show that if the $\langle x, y\rangle$-tree-path is not rotable, and the conditions of those lemmas fail, then (x, y) is not an interval edge. At first, we prove that if $\langle x, y\rangle$-tree-path does not end in a leaf, then (x, y) is not an interval edge.

Lemma 5.15. *If the* $\langle x, y\rangle$*-tree-path ends in a* Q*-node, then* (x, y) *is not an interval edge.*

Proof. If y belongs to the leftmost section of the *Q*-node, then we take an arbitrary vertex $z_1 \in V_1$. Otherwise, Lemma 5.3f supplies us with a vertex $z_1 \in \left(S_{l(y)} \cup V_{l(y)}\right) \setminus S_{l(y)+1}$. Analogously, we take a vertex $z_2 \in V_k$ if y belongs to the rightmost section, or $z_2 \in \left(S_{r(y)} \cup V_{r(y)}\right) \setminus S_{r(y)-1}$ otherwise. Clearly, there is no edge between z_1 and z_2, and both x and y are over z_1 and z_2. Thus, Observation 5.6 finishes the proof. □

Lemma 5.16. *If the $\langle x, y\rangle$-tree-path ends in a* P*-node that is not a leaf, then (x, y) is not an interval edge.*

Proof. Repeat the argument for Lemma 5.8 word by word. □

Now, we are going to prove that if the $\langle x, y\rangle$-tree-path goes through a central section, then (x, y) is not an interval edge. First, we define a *nested path*. Consider a Q-node with k sections $S_1, \ldots, S_k$. For further convenience we artificially add $S_0 = S_{k+1} = \emptyset$, and say that a set of vertices $\mathcal{P}_{i,j} = (S_i \cup \ldots \cup S_j) \setminus (S_{i-1} \cup S_{j+1})$ is a *nested path*, if for every $i \leqslant a < j$ there is at least one vertex $v_a \in \mathcal{P}_{i,j}$ that belongs to $S_a \cap S_{a+1}$. We call it a nested path, because vertices $v_i, v_{i+1} \ldots, v_{j-1}$ in that order form (possibly not simple) path in the graph G, and all of them are completely contained in the sections $S_i, \ldots, S_j$. In the following lemma, we show that if the $\langle x, y\rangle$-tree-path goes through a central section of some Q-node, then we can find an asteroidal triple in the graph $G - (x, y)$. Nested paths help us to find this triple.

Lemma 5.17. *If the $\langle x, y\rangle$-tree-path goes through a central section, then (x, y) is not an interval edge.*

Proof. Assume that the $\langle x, y\rangle$-tree-path goes through some central section S_i of a Q-node Q, and let $S_1, \ldots, S_k$ be the sections of Q. Subtrees of the first and last section are nonempty, so let $z_1 \in V_1$ and $z_2 \in V_k$. We show that vertices y, z_1 and z_2 form an asteroidal triple in the graph $G - (x, y)$.

Since the edge (x, y) has been removed, the path $z_1 - x - z_2$ avoids the neighborhood of y. To prove that there is a path between z_1 and y that avoids the neighborhood of z_2, we show that there is a nested path $\mathcal{P}_{1,k-1}$, and so the shortest path of form $z_1 - \mathcal{P}_{1,k-1} - y$ fulfill our requirements. Let $\mathcal{P}_{1,j}$ be the longest nested path, i.e. with maximal $j < k$, possible. If such a path would not exist then either S_1 is empty, or all vertices that belong to S_1 belong to all sections. In both cases, properties listed in Lemma 5.3 are violated. Moreover, if $j < k - 1$, then either $S_j \cap S_{j+1} = \emptyset$, or $S_j \cap S_{j+1} \subseteq S_k$. Again, this contradicts Lemma 5.3, and we are done. Finally a path from z_2 to y that avoids the neighborhood of z_1 is constructed in a similar way. □

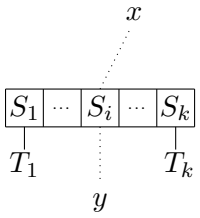

Figure 5.10: An $\langle x, y\rangle$-tree-path which goes through a central section S_i of some Q-node.

Summarizing the last three lemmas we get the following corollary.

Corollary 5.18. *If the $\langle x, y\rangle$-tree-path is not almost rotable, then (x, y) is not an interval edge.*

We are almost done with our classification. In Lemma 5.12 we proved that if the $\langle x, y\rangle$-tree-path is rotable, then (x, y) is always an interval edge. On the other hand, in

Lemmas 5.15, 5.16 and 5.17 we considered $\langle x, y\rangle$-tree-paths that are not almost rotable, and showed that for such paths (x, y) is not an interval edge. Hence, the only remaining cases are $\langle x, y\rangle$-tree-paths that are almost rotable, but not rotable. In Lemmas 5.13 and 5.14 we investigated such paths, and proved that under some additional conditions (x, y) is an interval edge. Now we show that if the $\langle x, y\rangle$-tree-path is almost rotable, but is not rotable, and conditions of Lemmas 5.13 and 5.14 are not satisfied, then (x, y) is not an interval edge. This case is the hardest one, so that we start with two auxiliary lemmas.

Lemma 5.19. *Let $S_1, \ldots, S_k$ be the sections of some* Q*-node in an* MPQ*-tree $\mathcal{T}$ (and to simplify notation add $S_0 = S_{k+1} = \emptyset$). Then for every $(a, b) \neq (1, k)$ with $a < b$ there is a vertex $v \in ((S_{a-1} \cap S_a) \setminus S_b) \cup ((S_b \cap S_{b+1}) \setminus S_a)$.*

Proof. Assume to the contrary that for some pair (a, b) as above there is no appropriate vertex, so each vertex $u \in S_a \cup \ldots \cup S_b$ is either fully contained in sections $S_a, \ldots, S_b$, or belongs to $S_a \cap \ldots \cap S_b$. Without loss of generality assume that $b < k$. Each vertex belongs to at least two sections. Hence, no vertex has its right endpoint in S_a or left endpoint in S_b, and Lemma 5.3 implies that V_a, V_b and V_k are not empty. Let $v_a \in V_a$, $v_b \in V_b$, $v_k \in V_k$, and consider some maximal cliques C_a, C_b and C_k such that $v_a \in C_a$, $v_b \in C_b$ and $v_k \in C_k$. Note that $\mathcal{T}$ encodes only those orderings of maximal cliques in which either $C_a < C_b < C_k$ or $C_k < C_b < C_a$. Now, consider a modified tree $\mathcal{T}'$ in which we reverse the order of sections $S_a, \ldots, S_b$. Clearly, because of our assumptions, $\mathcal{T}'$ represents the same interval graph as $\mathcal{T}$, but $\mathcal{T}'$ encodes an ordering $C_b < C_a < C_k$, which is not encoded by $\mathcal{T}$. Thus, $\mathcal{T}$ is not a valid *MPQ*-tree, and we are done. □

Lemma 5.20. *If the $\langle x, y\rangle$-tree-path starts in a central section S_a, and there is a vertex $q \in S_a \setminus \big(S_{l(x)} \cup S_{r(x)}\big)$, then (x, y) is not an interval edge.*

Proof. Assume to the contrary that (x, y) is an interval edge. Let P_{l_1,r_1} for $l(q) \leqslant r_1 < a$ be a nested path with the smallest l_1 possible. Analogously, let P_{l_2,r_2} for $a < l_2 \leqslant r(q)$ be a nested path with the biggest r_2 possible. Notice that these paths may not exist. For instance, if q has its right endpoint in S_a, then P_{l_2,r_2} does not exist.

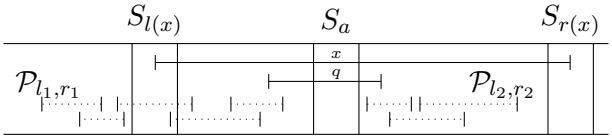

Figure 5.11: Paths $\mathcal{P}_{l_1,r_1}$ and $\mathcal{P}_{l_2,r_2}$.

At first, we consider the case where neither of these paths exist, to create an asteroidal triple in the graph $G - (x, y)$. By Lemma 5.19, without loss of generality, there is a vertex $v \in \big(S_{r(q)} \cap S_{r(q)+1}\big) \setminus S_{l(q)}$. Both paths do not exist, so v has to belong to S_a. Moreover, Lemma 5.3f gives two vertices $v_L \in \big(S_{l(q)} \cup V_{l(q)}\big) \setminus S_{l(q)+1}$, and $v_R \in \big(S_{r(q)+1} \cup V_{r(q)+1}\big) \setminus S_{r(q)}$. Thus, the paths: $v_L - x - v_R$, $v_L - q - y$, and $v_R - v - y$ certify the asteroidal triple $\{v_L, v_R, y\}$, and we are done in this case.

Now, assume that both paths exist, and put $L = \max\{l_1, l(x)\}$ and $R = \min\{r_2, r(x)\}$. Lemma 5.3f implies that there is a vertex v_L which has its right endpoint in S_L or

$v_L \in V_L$. Analogously, define a vertex v_R for the section S_R. Clearly, both v_L and v_R are neighbors of x, but not of q. Hence, $\{v_L, v_R, y\}$ is an asteroidal triple in the graph $G-(x,y)$, witnessed by the paths: $v_L - x - v_R$, $v_L - P_{l_1,r_1} - q - y$, and $v_R - P_{l_2,r_2} - q - y$. Thus, if both paths exist, then (x,y) is not an interval edge.

Without loss of generality, assume that $\mathcal{P}_{l_1,r_1}$ exists, but $\mathcal{P}_{l_2,r_2}$ does not. Lemma 5.19 supplies us with a vertex v such that $v \in (S_{l_1-1} \cap S_{l_1}) \setminus S_{r(q)}$, or $v \in \big(S_{r(q)} \cap S_{r(q)+1}\big) \setminus S_{l_1}$. Note that, a this point v and x might be the same vertex. We assumed that right path does not exist, so $S_{r(q)} \cap S_{r(q)+1} \subseteq S_a$. Moreover, l_1 is the smallest index such that there is a nested path P_{l_1,r_1} for $l(q) \leqslant r_1 < a$. Hence, $S_{l_1-1} \cap S_{l_1} \subseteq S_a$, and thus in either case v belongs to S_a. Lemma 5.3f gives three vertices: $v_L \in (S_{l_1} \cup V_{l_1}) \setminus S_{l_1-1}$, $v_R \in \big(S_{r(q)} \cup V_{r(q)}\big) \setminus S_{r(q)-1}$, and $v_{R+1} \in \big(S_{r(q)+1} \cup V_{r(q)+1}\big) \setminus S_{r(q)}$. Note that, both v_R and v_{R+1} are neighbors of x, and both v_L and v_{R+1} are not neighbors of q.

If $l_1 < l(x)$, then the paths: $v_L - P_{l_1,r_1} - x - v_{R+1}$, $v_L - P_{l_1,r_1} - q - y$, and $v_{R+1} - x - q - y$ certify the asteroidal triple $\{v_L, v_{R+1}, y\}$. Thus, we conclude that $l(x) \leqslant l_1 < l(q)$, which means that the whole path P_{l_1,r_1} is contained in x. Moreover, $v \neq x$, and v_L is a neighbor of x. If $v \in (S_{l_1-1} \cap S_{l_1}) \setminus S_{r(q)}$, then there is an asteroidal triple $\{v_L, v_R, y\}$ certified i.e. by the paths: $v_L - x - v_R$, $v_L - v - y$ and $v_R - q - y$. On the other hand, if $v \in \big(S_{r(q)} \cap S_{r(q)+1}\big) \setminus S_{l_1}$, then $\{v_L, v_{R+1}, y\}$ is an asteroidal triple certified i.e. by the paths: $v_L - x - v_{R+1}$, $v_L - P_{l_1,r_1} - q - y$, and $v_{R+1} - v - y$. Thus, in this case the edge (x,y) is also not an interval edge, and we are done. □

Finally, we are ready to prove the last two negative results on interval edges.

Lemma 5.21. *Suppose that the $\langle x,y\rangle$-tree-path starts in a central section S_a, y has a neighbor in the subtree T_a, and neither of the conditions* 1-4 *of Lemma 5.14 holds. Then (x,y) is not an interval edge.*

Proof. Assume to the contrary that (x,y) is an interval edge, and let z be the neighbor of y in T_a. Let $S_1, \ldots, S_k$ be the sections of $node(x)$, and for further convenience add $S_0 = S_{k+1} = \emptyset$. Our assumptions supplies us with $z_l \neq x$ and $z_r \neq x$ with the properties: $z_l \in \big(S_a \setminus S_{l(x)}\big) \cup \big(\big(S_{l(x)-1} \cap S_{l(x)}\big) \setminus S_a\big)$ and $z_r \in \big(S_a \setminus S_{r(x)}\big) \cup \big(\big(S_{r(x)} \cap S_{r(x)+1}\big) \setminus S_a\big)$.

First, we consider the case in which both z_l and z_r belong to the first summands. If $z_l = z_r$, then $z_l \in S_a \setminus \big(S_{l(x)} \cup S_{r(x)}\big)$ and Lemma 5.20 leads to a contradiction, hence $z_l \neq z_r$. Using the same argument, we also conclude that $z_l \in S_{r(x)}$ and $z_r \in S_{l(x)}$. Lemma 5.3 supplies us with vertices: $v_L \in \big(S_{l(x)} \cup V_{l(x)}\big) \setminus S_{l(x)+1}$ and $v_R \in \big(S_{r(x)} \cup V_{r(x)}\big) \setminus S_{r(x)-1}$. The paths: $v_L - x - v_R$, $v_L - z_r - y$, and $v_R - z_l - y$ certify the asteroidal triple $\{v_L, v_R, y\}$, and we are done in this case.

Now, without loss of generality, we assume that $z_r \in \big(S_{r(x)} \cap S_{r(x)+1}\big) \setminus S_a$. If z_l also belongs to the second summand, then consider vertices $v_L \in \big(S_{l(x)-1} \cup V_{l(x)-1}\big) \setminus S_{l(x)}$ and $v_R \in \big(S_{r(x)+1} \cup V_{r(x)+1}\big) \setminus S_{r(x)}$ provided by Lemma 5.3, and notice that the paths: $v_L - z_l - x - z_r - v_R$, $v_L - z_l - x - z - y$, and $v_R - z_r - x - z - y$ certify the asteroidal triple $\{v_L, v_R, y\}$.

Finally we are left with the case $z_l \in S_a \setminus S_{l(x)}$. Note that if $z_l \notin S_{r(x)}$, then $z_l \in S_a \setminus \big(S_{l(x)} \cup S_{r(x)}\big)$ and Lemma 5.20 leads to a contradiction. Hence, $z_l \in S_{r(x)}$ and there is an edge between z_l and z_r. Lemma 5.3f supplies us with two vertices $v_L \in \big(S_{l(x)} \cup V_{l(x)}\big) \setminus S_{l(x)+1}$ and $v_R \in \big(S_{r(x)+1} \cup V_{r(x)+1}\big) \setminus S_{r(x)}$. The paths: $v_L - x - z_r - v_R$, $v_L - x - z - y$, and $v_R - z_r - z_l - y$. certify the asteroidal triple $\{v_L, v_R, y\}$, and we are done. □

Lemma 5.22. *Suppose that the $\langle x, y\rangle$-tree-path starts in a central section S_a, y does not have a neighbor in subtree T_a, and neither of the conditions* 1-4 *of Lemma 5.13 holds. Then (x, y) is not an interval edge.*

Proof. Assume to the contrary that (x, y) is an interval edge. Let $S_1, \ldots, S_k$ be the sections of $node(x)$, and for further convenience add $S_0 = S_{k+1} = \emptyset$. Our assumptions supplies us with two sequences of vertices $z_1, \ldots, z_{l(x)}$ and $z_{r(x)}, \ldots, z_k$ with the properties: $z_i \in (S_a \setminus S_i) \cup ((S_{i-1} \cap S_i) \setminus S_a)$ for $1 \leqslant i \leqslant l(x)$, and $z_i \in (S_a \setminus S_i) \cup ((S_i \cap S_{i+1}) \setminus S_a)$ for $r(x) \leqslant i \leqslant k$. Moreover, none of these vertices is x. Let p_L be the longest sequence $z_L, z_{L+1}, \ldots, z_{l(x)}$ such that all vertices from p_L belong to the second summands. Analogously, let p_R be the longest sequence $z_{r(x)}, \ldots, z_{R-1}, z_R$ with the same property. Note that z_1 and z_k have to belong to the first summands since $S_0 \cap S_1 = S_k \cap S_{k+1} = \emptyset$. Therefore, $L > 1$ and $R < k$.

If both sequences p_L and p_R are empty (both vertices $z_{l(x)}$ and $z_{r(x)}$ do not belong to the second summands), then $z_{l(x)} \in S_a \setminus S_{l(x)}$ and $z_{r(x)} \in S_a \setminus S_{r(x)}$. The same argument as in the proof of Lemma 5.21 shows that in this case there is an asteroidal triple in the graph $G - (x, y)$. Thus, without loss of generality we assume that p_L contains at least one element ($L \leqslant l(x)$).

If p_R is also not empty, then Lemma 5.3f provides two vertices $v_L \in (S_{L-1} \cup V_{L-1}) \setminus S_L$ and $v_R \in (S_{R+1} \cup V_{R+1}) \setminus S_R$. Both sequences are the longest possible, so $z_{L-1} \in S_a \setminus S_{L-1}$ and $z_{R+1} \in S_a \setminus S_{R+1}$. The paths: $v_L - p_L - x - p_R - v_R$, $v_L - p_L - x - z_{R+1} - y$, and $v_R - p_R - x - z_{L-1} - y$ certify the asteroidal triple $\{v_L, v_R, y\}$, and we are done in this case.

The last remaining case is that p_R is empty, but p_L is not. In this case, take a vertex $v_R \in \left(S_{r(x)} \cup V_{r(x)}\right) \setminus S_{r(x)-1}$ provided by Lemma 5.3f. The sequence p_R is empty, so $z_{r(x)} \in S_a \setminus S_{r(x)}$. Yet, again we find an asteroidal triple $\{v_L, v_R, y\}$ certified by paths: $v_L - p_L - x - v_R$, $v_L - p_L - x - z_{r(x)} - y$, and $v_R - x - z_{L-1} - y$. □

This finishes our classification of interval edges. For every edge (x, y) that is not an interval edge we presented structures certifying that $G - (x, y)$ is not an interval graph, see Lemmas 5.8, 5.10, 5.15, 5.16, 5.17, 5.21 and 5.22. Moreover, for each edge (x, y) which is an interval edge, we provided a linear time algorithm that produces a string representation for the graph $G - (x, y)$, see Lemmas 5.9, 5.11, 5.12, 5.13, and 5.14. One of the consequences of all those lemmas is the following theorem.

Theorem 5.23. *There is a linear time algorithm, that for every* MPQ*-tree $\mathcal{T}$ representing graph G, and every interval edge (x, y), produces a string representation of the graph $G - (x, y)$.*

5.5 Listing Interval Edges

In this section we present an efficient algorithm that for a given *MPQ*-tree $\mathcal{T}$ lists all interval edges of the graph represented by $\mathcal{T}$.

Let $S_1, \ldots, S_k$ be the sections of a *Q*-node Q. For $b \in \{2, \ldots, k\}$, denote $f_r(b)$ the maximal index of a section such that $S_{b-1} \cap S_b \subseteq S_{f_r(b)}$. Analogously, for $b \in \{1, \ldots, k-1\}$, denote $f_l(b)$ the minimal index of a section such that $S_b \cap S_{b+1} \subseteq S_{f_l(b)}$. For every vertex x which belongs to Q, and every $l(x) \leqslant a \leqslant r(x)$ let:

$$L^*(x,a) = \begin{cases} 1 & \text{if } S_a = \{x\} \\ \max\limits_{v \in S_a \setminus \{x\}} l(v) & \text{if } S_a \neq \{x\} \end{cases} \qquad R^*(x,a) = \begin{cases} k & \text{if } S_a = \{x\} \\ \min\limits_{v \in S_a \setminus \{x\}} r(v) & \text{if } S_a \neq \{x\} \end{cases}$$

In other words, if $S_a \neq \{x\}$, then $L^*(x,a) = i$ if S_i is the rightmost section such that there is a vertex $v \neq x$ which belongs to S_a and v has its left endpoint in S_i. Now, we translate the inclusions between sections of a Q-node used in our characterization of interval edges into this new language:

i) $\forall_{1<b\leqslant a} : S_{b-1} \cap S_b \subseteq S_a \Leftrightarrow f_r(b) \geqslant a$.

ii) $\forall_{a\leqslant b<k} : S_b \cap S_{b+1} \subseteq S_a \Leftrightarrow f_l(b) \leqslant a$.

iii) $\forall_{l(x)\leqslant a\leqslant r(x)} : S_a \setminus \{x\} \subseteq S_b \Leftrightarrow b \in [L^*(x,a), R^*(x,a)]$.

Using this translation, we reformulate the conditions of Lemmas 5.13 and 5.14 into the form that will be easier to test algorithmically.

Observation 5.24. *For an $\langle x, y\rangle$-tree-path that starts in a central section S_a, the conditions* 1-4 *in Lemma 5.13 can be replaced by:*

1. $L^*(x,a) \leqslant l(x)$ *and* $\min\{l(x), R^*(x,a)\} > 1$ *and* $f_r(\min\{l(x), R^*(x,a)\}) \geqslant a$,
2. $R^*(x,a) \geqslant r(x)$ *and* $\max\{r(x), L^*(x,a)\} < k$ *and* $f_l(\max\{r(x), L^*(x,a)\}) \leqslant a$,
3. $L^*(x,a) = 1$,
4. $R^*(x,a) = k$,

respectively.

Proof. Note that, conditions (3) and (4) translate directly, and conditions (1) and (2) are symmetrical. Hence, we only show the equivalence of the first cases. At first, we show that if there is $1 < b \leqslant l(x)$ such that $S_a \setminus \{x\} \subseteq S_b$ and $S_{b-1} \cap S_b \subseteq S_a$, then $L^*(x,a) \leqslant l(x)$ and $f_r(\min\{l(x), R^*(x,a)\}) \geqslant a$. In our new language our assumptions are $b \in [L^*(x,a), R^*(x,a)]$ and $f_r(b) \geqslant a$. Thus, $L^*(x,a) \leqslant l(x)$, and $1 < b \leqslant \min\{l(x), R^*(x,a)\}$. Note that f_r is a non-decreasing function, so $f_r(b) \geqslant a$, implies $f_r(\min\{l(x), R^*(x,a)\}) \geqslant a$, and we are done.

Now, we let $b = \min\{l(x), R^*(x,a)\}$, and show that if $b > 1$, $f_r(b) \geqslant a$, and $L^*(x,a) \leqslant l(x)$, then $S_{b-1} \cap S_b \subseteq S_a$ and $S_a \setminus \{x\} \subseteq S_b$. Clearly, $1 < b \leqslant l(x)$ and $f_r(b) \geqslant a$, so $S_{b-1} \cap S_b \subseteq S_a$. Hence, the only thing we need to show is $S_a \setminus \{x\} \subseteq S_b$, which is equivalent to $b \in [L^*(x,a), R^*(x,a)]$. It is easy to see that for every x and a we have $L^*(x,a) \leqslant R^*(x,a)$. Thus, the interval $[L^*(x,a), R^*(x,a)]$ is never empty. If $b = R^*(x,a)$, then we are done, so assume that $b = l(x)$. But, in this case $L^*(x,a) \leqslant l(x) = b \leqslant R^*(x,a)$ as required. □

A very similar proof applies to reformulation of Lemma 5.14, so we leave it to the reader.

Observation 5.25. *For an $\langle x, y\rangle$-tree-path that starts in a central section S_a, the conditions* 1-4 *in Lemma 5.14 can be replaced by:*

1. $l(x) > 1$ *and* $l(x) \in [L^*(x,a), R^*(x,a)]$ *and* $f_r(l(x)) \geqslant a$,
2. $r(x) < k$ *and* $r(x) \in [L^*(x,a), R^*(x,a)]$ *and* $f_l(r(x)) \leqslant a$,
3. $l(x) = 1$ *and* $L^*(x,a) = 1$,
4. $r(x) = k$ *and* $R^*(x,a) = k$,

respectively.

As Observations 5.24 and 5.25 provide new fast and easy tests for interval edges, we are left with determining the values for all the functions l, r, f_l, f_r, L^* and R^*. To store functions l, r, f_l and f_r we use $O(n)$-element arrays of integers. Functions L^* and R^* in their explicit forms may require $O(n^2)$ space. Thus, instead of representing them as two-dimensional arrays, we replace first the function L^* by two one-dimensional functions L_1 and L_2. We set $L_1(a) = \max\{l(v) : v \in S_a\}$, and denote $v_1(a)$ a vertex for which $l(v_1(a)) = L_1(a)$. We also define $L_2(a) = \max\{l(v) : v \in S_a \setminus \{v_1(a)\}\}$, or $L_2(a) = 1$ if $S_a = \{v_1(a)\}$. Using those two functions we are able to compute the function L^* in the following way:

$$L^*(x,a) = \begin{cases} L_1(a) & \text{if } L_1(a) \neq l(x), \\ L_2(a) & \text{otherwise.} \end{cases}$$

Analogously, we can represent the function R^* using two functions R_1 and R_2. Now we are ready to show the interval edges enumeration algorithm.

Lemma 5.26. *The functions l, r, f_l and f_r for all* Q*-nodes of the tree $\mathcal{T}$ can be computed in $O(n \log n)$ time.*

Proof. We argue that computing f_r for a single Q-node, say Q_d with n_d vertices, takes $O(n_d \log n_d)$ time. Then we simply sum up all Q-nodes, and since $\sum n_d \leqslant n$ the computation of all functions f_r takes no more than $O(n \log n)$ time.

Let i be the minimum index of the section containing a right endpoint of some vertex from $S_{b-1} \cap S_b$. Clearly, $S_{b-1} \cap S_b \subseteq S_i$ and $\neg(S_{b-1} \cap S_b \subseteq S_{i+1})$. Hence, $f_r(b) = i$, and in order to compute the function f_r for all b, it is enough to scan the sections of Q_d from left to right maintaining a heap of right endpoints. When algorithm enters the section S_{b+1} it adds to the heap all right endpoints of vertices that have its left endpoint in S_b, assigns $f_r(b)$ to be the minimum value stored in the heap, and removes all the endpoints of vertices that have its right endpoint in S_b. Thus, the computation of the function f_r for Q_d takes $O(n_d \log n_d)$ time. A similar argument applies to the functions f_l, and functions l and r can be easily computed in $O(n)$ time. □

Lemma 5.27. *The functions L_1, L_2, R_1 and R_2 for all* Q*-nodes of the tree $\mathcal{T}$ can be computed in $O(n \log n)$ time.*

Proof. The proof is very similar to the proof of Lemma 5.26. Yet again, we use a heap of right or left endpoints, but now we are interested not only in the minimal element but also in the second minimal one. □

Theorem 5.28. *There is an algorithm that for a given* MPQ*-tree $\mathcal{T}$ lists all its interval edges in $O(\max\{n + m, n \log n\})$ time.*

Proof. The algorithm is pretty straightforward. At first, we compute all the functions l, r, f_l, f_r, L_1, L_2, R_1 and R_2 in $O(n \log n)$ time. Then, we inspect all P-nodes and all sections of Q-nodes listing all edges (x, y) that satisfy the conditions of Lemmas 5.9 or 5.11. For a P-node that is a leaf, we list all pairs (v_a, v_b) for $a \neq b$, while for the section S_i with subtree that is either empty or is a P-node with no children, we list the edges of the form (v_L, v_R), where v_L is a vertex that has its right endpoint in S_i, and v_R is a vertex that has its left endpoint in S_i. Note that, an *MPQ*-tree has no more than $O(n)$ nodes and sections. Moreover, we have a constant time access to all vertices v_L and v_R. Thus, this phase works in $O(n + m)$.

Now, we want to find all interval edges (x, y) such that x and y belong to different nodes in $\mathcal{T}$, and x is over y. Lemmas 5.15 and 5.16 imply that it is enough to consider only those pair of vertices (x, y), where y belongs to a leaf of $\mathcal{T}$. Hence, for each leaf L of a tree $\mathcal{T}$, we traverse a unique path between L and the root of $\mathcal{T}$, starting from L, and listing edges of the form (x, y), where x belongs to the currently visited node, and y belongs to L. We do not list all such edges, but for each candidate we need to decide whether (x, y) is an interval edge or not. In order to do it efficiently, we keep two boolean variables: 1. does y have a neighbor in the visited subtree, and 2. does the path go through some central section. Using those two variables and precomputed functions, we can decide whether (x, y) is an interval edge in a constant time thanks to Lemma 5.12, and Observations 5.24 and 5.25. Thus, the time spent by the algorithm in this phase is bounded from above by the sum of lengths of all paths we have visited plus the number of tested edges. Lemma 5.3g, implies that on those paths there are no two consecutive empty P-nodes, so we can bound the sum of lengths of those paths by $O(m)$. Thus, the time spent by the algorithm in this phase is $O(n + m)$, and the whole algorithm works in $O(\max\{n + m, n \log n\})$ time. □

5.6 The Enumeration Algorithm

In this section we present the graph enumeration algorithm. We define a parent-child relationship in the same way as authors in [44] did using the following lemma.

Lemma 5.29 ([27, 44])**.** *If $G = (V, E)$ is an interval graph which is not a clique, then there is at least one edge $e \notin E$ such that $G + e$ is also an interval graph.*

Yamazaki et al. used Lemma 5.29 and defined the parent of G to be a graph $G + e$ such that $G + e$ is an interval graph and e is lexicographically the smallest possible. They proved that for every interval graph G its parent can be computed in $O(n + m)$ time.

Theorem 5.30 ([44] Thm. 5)**.** *Let $G = (V, E)$ be any interval graph. Then its parent can be computed in $O(n + m)$ time.*

Thanks to the fact that we work with *MPQ*-trees and string representations, we get rid of the $O(m)$ summand using the algorithm from Theorem 5.1 in the way described below.

Theorem 5.31. *There is an $O(n)$ time algorithm that for every canonical* MPQ*-tree representing an interval graph G that is not a clique, produces a string representation of a graph $G + e$, where $e \notin E$ is the lexicographically smallest edge.*

Proof. Let $\mathcal{S}$ be the canonical string of the considered *MPQ*-tree. Let $12...j$ be the longest (possibly empty) prefix of $\mathcal{S}$ such that $j...21$ is a suffix of $\mathcal{S}$. Clearly, such prefix can be found in $O(n)$ time. Moreover, all vertices $1, \ldots, j$ have degree $n-1$ in the graph G. Hence there is no pair $(x, y) \notin E$ such that $x \in [j]$. We prove that there is a pair $(j+1, y)$ for some $y \geqslant j+2$ such that $G+(j+1, y)$ is an interval graph. Let y be the interval with the leftmost left endpoint that is to the right of the right endpoint of $j+1$. If such y does not exist, then to the right of the right endpoint of $j+1$ there are only right endpoints. But, this combined with the form of the canonical string contradict the maximality of the prefix $12...j$. Thus, y exists, $(j+1, k)$ is an edge for all $k < y$, and there is no edge between $j+1$ and y. So, $G+(j+1, y)$ is the parent of G. In order to produce a string representation for the graph $G+(j+1, y)$, we move the right endpoint of $j+1$ to the right until it passes the left endpoint of y. Lemma 5.7 guarantees that no edge except $(j+1, y)$ was added. □

Finally, we take all the pieces together and present the key procedure of our graph enumeration algorithm.

Theorem 5.32. *Let $\mathcal{T}$ be a canonical* MPQ*-tree representing a non-empty ($m \geqslant 1$) interval graph G. The set of canonical trees for children of G can be computed in $O(nm \log n)$ time.*

Proof. At first, we list all interval edges for the tree $\mathcal{T}$ in $O(\max\{n+m, n \log n\})$ time, see Theorem 5.28. Then, for each interval edge e, we produce a string representation $\mathcal{S}'$ of $G-e$ in $O(n)$ time, build *MPQ*-tree $\mathcal{T}'$ for the interval graph represented by $\mathcal{S}'$ in $O(n)$, and finally compute the canonical form of $\mathcal{T}'$ in $O(n \log n)$ time, see Theorems 5.23, 5.1, and 5.5. Hence, we compute all canonical strings for children candidates in $O(nm \log n)$ time. Theorem 5.4 implies that in order to check isomorphism between the children, it is enough to remove duplicates from this set. It can be easily done by storing all computed strings in a trie. Finally, we need to filter out those graphs for which G is not a parent. For each candidate's canonical string, we compute its parent's string in $O(n)$ time, see Theorem 5.31, build a canonical *MPQ*-tree in $O(n \log n)$ time and check whether its canonical string equals with a canonical string for the graph G in $O(n)$ time. Thus, we filter out non-children in $O(nm \log n)$ time, and the whole process takes $O(nm \log n)$ time. □

5.7 Performance

In the previous section we have presented theoretical analysis of our enumeration algorithm. In this section we present two lemmas, that helped us to significantly speedup the execution of our algorithm.

Lemma 5.33. *Let $v_1 \neq v_2$ be two vertices of the graph $G = (V, E)$. If $N(v_1) \setminus \{v_2\} = N(v_2) \setminus \{v_1\}$, then for every $y \notin \{v_1, v_2\}$ graphs $G-(v_1, y)$ and $G-(v_2, y)$ are isomorphic.*

Proof. One can easily check that the function $f : V \to V$ such that $f(v_1) = v_2$, $f(v_2) = v_1$ and f keeps other vertices untouched, is an isomorphism between $G-(v_1, y)$ and $G-(v_2, y)$. □

A consequence of the above lemma is that if a *P*-node contains more than one vertex namely $v_1, \ldots, v_j$, then graphs $G-(v_1, y)$ and $G-(v_a, y)$ for every $a > 1$ are isomorphic. Thus, when listing potential candidates for the children of the graph G, we may omit all edges of the form (v_a, y) for $a > 1$ and list only the edges of the form (v_1, y). The same argument applies to a *Q*-node and vertices that occupy exactly the same sections.

Lemma 5.34. *Let $\{v_1, \ldots, v_k\}$ be a clique in the graph $G = (V, E)$ such that $\forall_{i \neq j} : N(v_i) \setminus \{v_j\} = N(v_j) \setminus \{v_i\}$. All graphs $G_{i,j} = G - (v_i, v_j)$ are pairwise isomorphic.*

Proof. Consider two graphs $G_{i,j}$ and $G_{i',j'}$ and a function $f : V \to V$ such that $f(v_i) = v_{i'}$, $f(v_j) = v_{j'}$. On the other vertices f is an identity. □

A consequence of this lemma is that if a *P*-node contains more than 2 vertices, then we may omit all the edges between them, but one. The same applies to the vertices that occupy exactly the same sections of a *Q*-node.

Unfortunately, the tricks of Lemmas 5.33 and 5.34 do not improve the worst-case time delay, and it is quite easy to show an *MPQ*-tree that even with those tricks take $O(nm \log n)$ time to process, see Figure 5.12 for example.

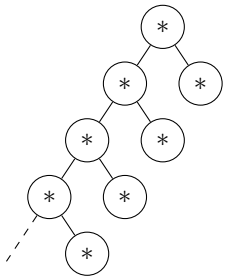

Figure 5.12: *MPQ*-trees with $\Theta(n^2)$ interval edges – each node is a *P*-node and contains exactly one vertex of the graph.

Storing results

We did implement the proposed algorithms, and generated all non-isomorphic interval graphs on n vertices for all $n \leqslant 15$. The results are available at `https://patrykmikos.staff.tcs.uj.edu.pl/graphs/`. Our algorithm is fast enough to generate all non-isomorphic interval graphs even for higher values of n in reasonable time, but unfortunately we lack enough space to efficiently store them. For instance string representations of all non-isomorphic interval graphs on 15 vertices occupies about 200 GB of disk space. Based on our results, we estimate that string representations of all non-isomorphic interval graphs on 20 vertices would occupy more than 20 PB. In order to significantly reduce the space usage, we developed a simple binary format to store the generated graphs.

Shorter canonical representation

In order to store generated string representations efficiently, at first we get rid of redundant information from the string representation. For each $2n$-element canonical string

representation we consider its prefix that ends at the first occurrence of the element n. Note that all edges of the original graph are encoded in that prefix, so there is no need to store the longer sequence. Moreover, we can also remove the element n since all elements $[n-1]$ are present in the prefix and so n is uniquely determined.

For instance the sequence $1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 8\ 9\ 7\ 10\ 10\ 9\ 6\ 5\ 4\ 3\ 2\ 1$ translates into $1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 8\ 9\ 7$. In order to convert the shortened sequence into the original one, at first we find the maximum element it contains and denote it x. Then we add $x+1$ two times and all elements that appeared exactly once in the shortened form in a decreasing order. In the sequence $1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 8\ 9\ 7$ the maximum element is 9 and elements $1\ 2\ 3\ 4\ 5\ 6\ 9$ appear exactly once. So we append a sequence $10\ 10\ 9\ 6\ 5\ 4\ 3\ 2\ 1$ at the end.

Binary format for Trie

Now we present a binary format used to store a Trie of shortened canonical representations. We assume that this Trie contains only canonical representations of graphs on n vertices for some of the ours n. We use the first byte of the file to store the information about that n. Then, we recursively store Trie nodes. For each node, we store 2 bits. First of them is 1 if Trie contains a representation that ends in this node. The second bit is 1 if the given node is a leaf in Trie. For inner nodes (the second bit is 0) we store an additional vector of $n-1$ bits, where i-th bit tells whether this node has a child labeled with i. Thus, for each Trie node that is a leaf we use exactly 2 bits, while for the inner nodes we use $n+1$ bits per each. Finally, we compress the resulting binary file using LZMA compression with parameter *–best*. The following table shows the results we achieved.

n	Number of graphs	Size of raw file	Size of binary file	Size of compressed file
1	1	5 B		
2	2	18 B		
3	4	52 B		
4	10	170 B		
5	27	567 B		
6	92	2.3 kB		
7	369	11 kB		
8	1,807	59 kB		
9	10,344	374 kB		
10	67,659	2.7 MB		
11	491,347	23 MB		
12	3,894,446	197 MB	6.7 MB	727 kB
13	33,278,992	1.9 GB	60 MB	2.3 MB
14	304,256,984	19 GB	565 MB	17.2 MB
15	2,290,093,835		5.6 GB	76.2 MB

Table 5.1: Size of files required to store all generated graphs.

Bibliography

[1] Hüseyin Acan. Counting unlabeled interval graphs, 2018.

[2] Udo Adamy and Thomas Erlebach. Online coloring of intervals with bandwidth. In *WAOA 2003: 1st International Workshop on Approximation and Online Algorithms, Budapest, Hungary, September 2003. Proceedings*, volume 2909 of *Lecture Notes in Computer Science*, pages 1–12, 2004.

[3] D. Avis and K. Fukuda. Reverse search for enumeration. *Discrete Applied Mathematics*, 65(1):21–46, 1996. First International Colloquium on Graphs and Optimization.

[4] Yossi Azar, Amos Fiat, Meital Levy, and NS Narayanaswamy. An improved algorithm for online coloring of intervals with bandwidth. *Theoretical Computer Science*, 363(1):18–27, 2006.

[5] János Balogh, József Békési, and Gábor Galambos. New lower bounds for certain classes of bin packing algorithms. *Theoretical Computer Science*, 440-441:1–13, 2012.

[6] Kenneth P. Bogart and Douglas B. West. A short proof that 'proper = unit'. *Discrete Mathematics*, 201(1):21–23, 1999.

[7] Marek Chrobak and Maciej Ślusarek. On some packing problem related to dynamic storage allocation. *Informatique théorique et applications*, 22(4):487–499, 1988.

[8] Edward G. Coffman and János Csirik. Performance guarantees for one-dimensional bin packing. In *Handbook of Approximation Algorithms and Metaheuristics*, 2007.

[9] Edward G. Coffman Jr., János Csirik, Gábor Galambos, Silvano Martello, and Daniele Vigo. *Bin Packing Approximation Algorithms: Survey and Classification*, pages 455–531. Springer New York, New York, NY, 2013.

[10] János Csirik and Gerhard J. Woeginger. *On-line packing and covering problems*, pages 147–177. Springer Berlin Heidelberg, Berlin, Heidelberg, 1998.

[11] Reinhard Diestel. *Graph Theory*. Springer Berlin Heidelberg, 2017.

[12] Leah Epstein and Meital Levy. Online interval coloring and variants. In *ICALP 2005: 32nd International Colloquim on Automata, Languages and Programming, Lisbon, Portugal, July 2005. Proceedings*, volume 3580 of *Lecture Notes in Computer Science*, pages 602–613, 2005.

[13] Leah Epstein and Meital Levy. Online interval coloring with packing constraints. In *MFCS 2005: 30th International Symposium on Mathematical Foundations of Computer Science, Gdańsk, Poland, August 2005. Proceedings*, volume 3618 of *Lecture Notes in Computer Science*, pages 295–307, 2005.

[14] P. C. Fishburn and R. L. Graham. Classes of interval graphs under expanding length restrictions. *Journal of Graph Theory*, 9(4):459–472, 1985.

[15] G. Galambos and J.B.G. Frenk. A simple proof of liang's lower bound for on-line bin packing and the extension to the parametric case. *Discrete Applied Mathematics*, 41(2):173–178, 1993.

[16] Magnús M. Halldórsson. Parallel and on-line graph coloring. *Journal of Algorithms*, 23(2):265–280, 1997.

[17] Magnús M. Halldórsson and Mario Szegedy. Lower bounds for on-line graph coloring. *Theoretical Computer Science*, 130(1):163–174, 1994.

[18] P. Hanlon. Counting interval graphs. *Transactions of The American Mathematical Society - TRANS AMER MATH SOC*, 272, 1982.

[19] N. Pippenger J.C. Yang. On the enumeration of interval graphs. *Proceedings of the American Mathematical Society Series B*, 4:1–3, 2017.

[20] D. S. Johnson. Near-optimal bin packing algorithms. *PhD Thesis*, 1973.

[21] D. S. Johnson, A. Demers, J. D. Ullman, M. R. Garey, and R. L. Graham. Worst-case performance bounds for simple one-dimensional packing algorithms. *SIAM Journal on Computing*, 3(4):299–325, 1974.

[22] K. Junosza-Szaniawski, P. Rzążewski, J. Sokół, and K. Węsek. Online coloring and L(2,1)-labeling of unit disk intersection graphs. *SIAM Journal on Discrete Mathematics*, To appear.

[23] H. A. Kierstead. The linearity of first-fit coloring of interval graphs. *SIAM Journal on Discrete Mathematics*, 1(4):526–530, 1988.

[24] H. A. Kierstead and Jun Qin. Coloring interval graphs with first-fit. *Discrete Mathematics*, 144(1):47 – 57, 1995.

[25] Henry A. Kierstead, David A. Smith, and William T. Trotter. First-fit coloring on interval graphs has performance ratio at least 5. *European Journal of Combinatorics*, 51:236–254, 2016.

[26] Henry A. Kierstead and William T. Trotter. An extremal problem in recursive combinatorics. In *12th Southeastern Conference on Combinatorics, Graph Theory and Computing, Baton Rouge, LA, USA, March 1981. Proceedings, vol. II*, volume 33 of *Congressus Numerantium*, pages 143–153, 1981.

[27] M. Kiyomi, S. Kijima, and T. Uno. Listing chordal graphs and interval graphs. In *Graph-Theoretic Concepts in Computer Science*, pages 68–77, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[28] N. Korte and R. Möhring. An incremental linear-time algorithm for recognizing interval graphs. *SIAM Journal on Computing*, 18(1):68–81, 1989.

[29] F.M. Liang. A lower bound for on-line bin packing. *Inform. Process. Lett.*, 10:76–79, 1980.

[30] G.S. Lueker and K.S. Booth. A linear time algorithm for deciding interval graph isomorphism. *J. ACM*, 26(2):183–195, 1979.

[31] N. S. Narayanaswamy and R. Subhash Babu. A note on first-fit coloring of interval graphs. *Order*, 25(1):49–53, 2008.

[32] NS Narayanaswamy. Dynamic storage allocation and on-line colouring interval graphs. In *COCOON 2004: 10th Annual International Conference on Computing and Combinatorics, Jeju Island, Korea, August 2004. Proceedings*, volume 3106 of *Lecture Notes in Computer Science*, pages 329–338, 2004.

[33] Sriram V. Pemmaraju, Rajiv Raman, and Kasturi Varadarajan. Buffer minimization using max-coloring. In *Proceedings of the Fifteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '04, pages 562–571, Philadelphia, PA, USA, 2004. Society for Industrial and Applied Mathematics.

[34] Sriram V. Pemmaraju, Rajiv Raman, and Kasturi R. Varadarajan. Max-coloring and online coloring with bandwidths on interval graphs. *ACM Transactions on Algorithms*, 7(3):35:1–35:21, 2011.

[35] F. S. Roberts. Indifference graphs. In *Proof Techniques in Graph Theory, F. Harary (Ed.)*, pages 139–146. Academic Press, 1969.

[36] T. Saitoh, Y. Otachi, K. Yamanaka, and R. Uehara. Random generation and enumeration of bipartite permutation graphs. *Journal of Discrete Algorithms*, 10:84–97, 2012.

[37] T. Saitoh, K. Yamanaka, M. Kiyomi, and R. Uehara. Random generation and enumeration of proper interval graphs. *IEICE Transactions on Information and Systems*, E93.D(7):1816–1823, 2010.

[38] J. Sylvester. On a point in the theory of vulgar fractions. *American Journal of Mathematics*, 3(4):332–335, 1880.

[39] R. Uehara T. Saitoh, M. Kiyomi. Simple efficient algorithm for mpq-tree of an interval graph. *IEICE Technical Report*, 107(127):49–54, 2007.

[40] R. Uehara. Canonical data structure for interval probe graphs. In R. Fleischer and G. Trippen, editors, *Algorithms and Computation*, pages 859–870, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[41] André van Vliet. Lower bound and upper bounds for on-line bin packing and scheduling algorithms, ph.d. thesis. *Tinbergen Institute Research Series*, 93.

[42] André van Vliet. An improved lower bound for on-line bin packing algorithms. *Inf. Process. Lett.*, 43(5):277–284, 1992.

[43] K. Yamazaki, T. Saitoh, M. Kiyomi, and R. Uehara. Enumeration of nonisomorphic interval graphs and nonisomorphic permutation graphs. In *WALCOM: Algorithms and Computation*, pages 8–19, Cham, 2018. Springer International Publishing.

[44] K. Yamazaki, T. Saitoh, M. Kiyomi, and R. Uehara. Enumeration of nonisomorphic interval graphs and nonisomorphic permutation graphs. *Theoretical Computer Science*, 2019.