

Radboud University



---

# Coloring Hyperholes and Ring Graphs

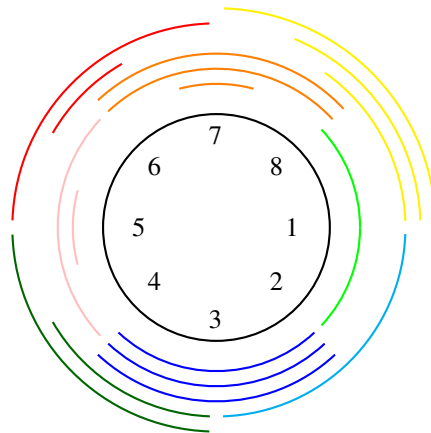
---

MASTER'S THESIS IN MATHEMATICS

*Author:*  
Els Hoekstra

*Supervisor:*  
Dr. Wieb Bosma  
Radboud University

*Second reader:*  
Prof. Dr. Kristina Vušković  
University of Leeds



November 2021

# Acknowledgements

First and foremost, I would like to thank my supervisor Wieb Bosma for the amount of time he invested in me and his support during this thesis. For the weekly meetings, which were always fun and helped me to continue with the thesis. I want to thank him for his guidance, his positive feedback towards my thesis and his willingness to read all of the versions I sent him. For his patience and understanding when I needed a break and for believing in me.

I also want to thank Kristina Vušković. Her enthusiasm for the course ‘Graph Theory: Structure and Algorithms’ led me to a subject in graph theory. Thank you for giving me the subject of this thesis, I truly enjoyed doing this research. I want to give thanks to Kristina Vušković for being a second reader, for taking the time to look at my thesis in detail and attending my presentation.

Two persons whom I am also grateful for, are my study advisors Bram Arens and Ina de Vries. To Bram Arens, who helped me go through a tough period in my study by taking the time for weekly calls. For helping me to take the step to change the subject of my thesis into one which is better suited for me. To Ina de Vries who guided me throughout the study.

A special thanks goes to my friends and family for their support. They sympathised with me in my ups and downs, they were enthusiastic in major breakthroughs and stood with me when things got rough. The endless study sessions with my friends helped me to stay focused and have a great start. Extra thanks for Jimmy who understood and supported me when I was passionate to continue with my thesis, even in the weekends. And an extra thanks for Timo, Jasper, Paul and Willemijn who willingly checked my thesis on (textual) errors.

# Contents

<b>Introduction</b>	<b>3</b>
Definitions and notation . . . . .	4
<b>1 Coloring circular-arc graphs</b>	<b>6</b>
1.1 Interval graphs . . . . .	6
1.2 Introduction to circular-arc graphs . . . . .	6
1.3 Coloring circular-arc graphs in general . . . . .	7
1.4 Proper circular-arc graphs . . . . .	9
1.5 Perfect circular-arc graphs . . . . .	11
<b>2 Chromatic polynomial for hyperholes</b>	<b>12</b>
2.1 Fundamental theorems for chromatic polynomials . . . . .	13
2.2 Hyperpaths . . . . .	14
2.3 Hyperholes . . . . .	16
2.4 Magma program and computation time . . . . .	23
2.5 Chromatic polynomial for classes of hyperholes . . . . .	24
<b>3 Coloring rings</b>	<b>27</b>
3.1 Recognizing rings . . . . .	27
3.2 Coloring even rings . . . . .	31
3.3 Coloring odd hyperholes . . . . .	32
3.4 Coloring odd rings . . . . .	34
<b>Bibliography</b>	<b>42</b>
<b>Appendix</b>	<b>44</b>
Preliminary algorithms . . . . .	44
Algorithms for hyperholes . . . . .	46
Algorithms for rings . . . . .	47

# Introduction

In 1959, S. Benzer researched the internal structure of chromosomes [1]. In this genetic analysis, the question arose whether or not mutations were in linear order. It could be tested whether two different mutations overlap and based on this an intersection graph can be constructed. If this intersection graph is a linear interval graph, then the mutations are in linear order.

V. Klee first introduced circular-arc graphs in 1969 [11]. Circular-arc graphs are intersection graphs of arcs laying on a circle. See Figure 1 for an example. A fast recognition algorithm was searched for to detect circularity in genetics. Circular-arc graphs were extensively researched by A. Tucker and others. Tucker first discovered a polynomial time recognition algorithm for circular-arc graphs in 1980 [24].

In 2014 G. Durán, L.N. Grippo and M.D. Safe wrote a survey on structural results relating to circular-arc graphs [7]. Most results and important open problems contained characterizations of circular arc graphs through forbidden induced subgraphs, none of the results or problems were related to the coloring of circular-arc graphs.

This thesis discusses the coloring of circular-arc graphs.

An application for coloring circular-arc graphs is found in scheduling of transport [23]. Every hour the schedule for busses repeats. The time it takes to complete a route can be represented as an arc on a circle. See for an example Figure 1, where route *A* is driven from a quarter past the hour till 15 minutes later. For cost efficiency we want the least possible number of bus drivers to work the shifts. Every chauffeur receives a different color and we can color the arcs on the circular-arc model by who drives the route. This coloring is proper and we want to use the least possible number of colors.

Another application is in the design of compilers where a variable is used in a certain amount of time and allocated to a register [23]. In while loops or for loops the same variables are used in every round. The least number of registers necessary is related to an optimal coloring of a circular-arc graph. Other applications are phasing of traffic lights [8] [19] and can be found in genetics [20].

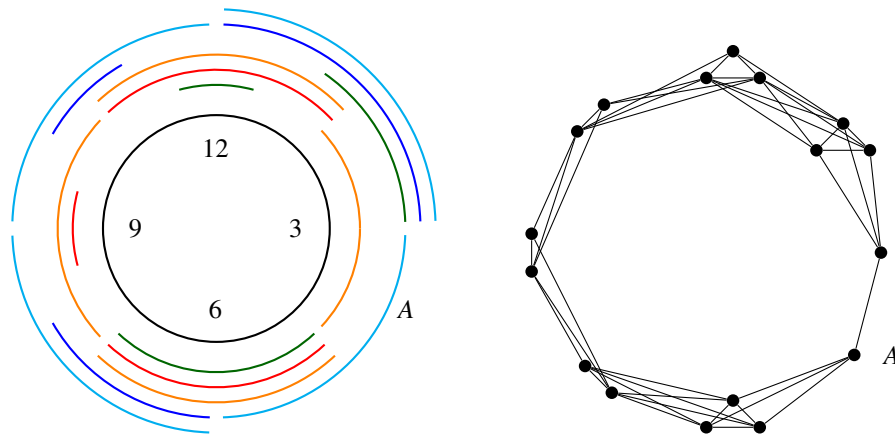


Figure 1: Bus schedule. The arcs represents a route, a color represents a bus driver. On the right its intersection graph.

In this thesis we start with an introduction to circular-arc graphs. We first prove that the coloring problem for circular-arc graphs is NP-complete [6] and then describe two polynomial time algorithms to color proper [17] and perfect [25] circular-arc graphs. This is all discussed in Chapter 1.

Then we focus on hyperholes. Hyperholes can be described as holes (or irreducible cycles of length  $\geq 4$ ) where every vertex is blown up to a clique. See Figure 2 for an example and Definition 2.12 for the definition of hyperholes.

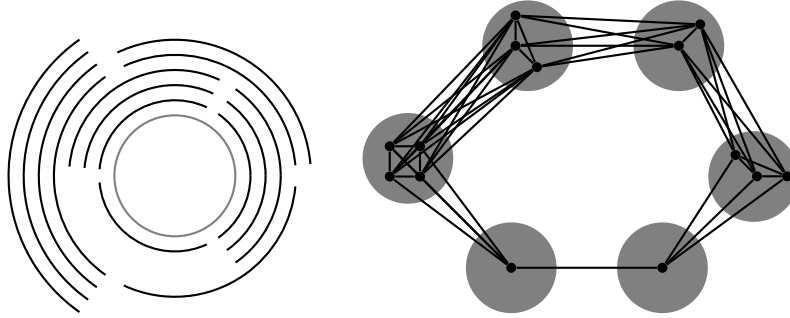


Figure 2: On the left a circular-arc model of the hyperhole on the right.

A recursive relation for the chromatic polynomial of a hyperhole is discovered and proved in Chapter 2. We describe a fast algorithm to determine the chromatic polynomial of a hyperhole in section 2.4 and give the chromatic polynomial for the class of hyperholes where at least one vertex is blown up by a clique of size one (like the hyperhole in Figure 2) in section 2.5.

In 2020, F. Maffray, I. Penev and K. Vušković [14] discovered a polynomial time algorithm to color odd rings. Rings form a subclass of circular-arc graphs and are subgraphs of hyperholes with extra conditions. See Figure 1 for an example and Definition 3.1 for the definition of a ring. The possibility to color rings in polynomial time was an answer to an open problem in [2]. We describe the algorithms from [2][14] in Chapter 3. The implementations are new, and have led to the correction of some minor errors in the description [3].

## Definitions and notation

Before we start, we need to agree on some definitions and notation. This section will provide the necessary terminology which is used throughout the paper. Some definitions, such as those of chromatic polynomials and rings, will be introduced in the designated chapters.

All the graphs in this paper are finite, nonempty, simple (i.e., unweighted, undirected, without self-loops nor multiple edges between the same pair of vertices) graphs. We denote  $V(G)$  for the set of vertices of  $G$  and  $E(G)$  for the edges. The letter  $n$  represents the number of vertices in  $G$  and the letter  $m$  the number of edges. The *complement graph* of  $G$  is denoted as  $\overline{G} = (V(G), \overline{E}(G))$ . The complement graph has all the same vertices as the graph itself and  $e$  is an edge in  $\overline{G}$  if and only if  $e$  is not an edge in  $G$ . A graph is *connected* if for every pair of vertices there exists a path connecting them. The most common finite, nonempty, simple graphs are *complete graphs* (i.e., graphs with an edge between every vertex, notation:  $K_n$ ), *edgeless graphs* ( $\overline{K}_n$ ), *paths* (i.e., graphs with vertices  $V = \{v_1, \dots, v_n\}$  and edges  $E = \{\{v_i, v_{i+1}\} : 1 \leq i \leq n - 1\}$ , notation:  $P_n$ ), *cycles* (i.e., graphs where  $(v_1, \dots, v_n)$  is a path and  $\{v_1, v_n\}$  is also an edge, notation:  $C_n$ ) and *trees* (i.e., connected graphs with no cycles).

A *coloring* for a graph  $G$  is a map from the vertices of  $G$  to  $\mathbb{N}$ . It assigns to every vertex a color. We say that  $c : V(G) \rightarrow \mathbb{N}$  is a *proper coloring* for  $G$  if and only if for all edges  $\{v, w\} \in E(G)$  it holds that  $c(v) \neq c(w)$ . From now on we only consider proper colorings. For an integer  $r \in \mathbb{N}$  we say  $G$  is  *$r$ -colorable* if there is a coloring of  $V(G)$  that uses at most  $r$  colors. The *chromatic number*  $\chi(G)$  of  $G$  is the smallest such integer.

We denote  $G[S]$  for the *subgraph induced* by the subset  $S \subseteq V(G)$ . The vertex set of  $G[S]$  is equal to  $S$  and the edges are all edges of  $G$  with both endpoints in  $S$ . The graph  $G \setminus S := G[V(G) \setminus S]$

is the graph where the vertices of  $S$  are removed. We say  $S$  is a *clique* of  $G$  if all vertices in  $G[S]$  are connected. A clique  $C$  is said to be maximal in graph  $G$  if there is no vertex  $v$  in  $G$ , such that  $G[C \cup \{v\}]$  is a clique. The size of the largest clique in  $G$  is denoted as  $\omega(G)$ . A *cutset* of a graph  $G$  is a (possibly empty) subset  $S \subseteq V(G)$  of the vertices of  $G$  such that  $G \setminus S$  is disconnected. A *clique-cutset* is a cutset  $S$  for which  $G[S]$  is a clique. We define a *clique-cut-partition* as a triple  $(A, B, C)$  such that  $A, B$  are nonempty,  $(A, B, C)$  is a partition for the vertices of  $G$ , no vertex in  $A$  is connected to a vertex in  $B$  and the last condition is that  $C$  is a clique-cutset for  $G$ .

Two graphs  $G$  and  $H$  are *isomorphic* if there exists a bijection between the vertices  $f : V(G) \leftrightarrow V(H)$  such that  $\{v, w\} \in E(G)$  if and only if  $\{f(v), f(w)\} \in E(H)$ . The notation for two isomorphic graphs  $G, H$  is  $G \cong H$ . We say that  $G$  *contains* a graph  $H$  if there is a subset  $S \subseteq V(G)$  such that  $G[S]$  and  $H$  are isomorphic.

The *neighbors* of a vertex  $v \in V(G)$  are all vertices in  $G$  that are connected to the vertex  $v$ . We will denote *the neighborhood* of vertex  $v$  in graph  $G$  by  $N_G(v)$ . So  $N_G(v) := \{w : \{v, w\} \in E(G)\}$ . We will define  $N_G[v] := N_G(v) \cup \{v\}$ ; i.e., the neighborhood of  $v$  with the vertex  $v$  itself included. Let  $S$  be a subset of  $V(G)$ , then a vertex  $v \notin S$  is *complete* to  $S$  if  $S \subseteq N_G(v)$  and *anticomplete* if  $S \cap N_G(v) = \emptyset$ . A subset  $A \subseteq V(G)$  is called complete (resp. anticomplete) to  $S$  if all vertices in  $A$  are complete (resp. anticomplete) to  $S$ .

A *simplicial vertex*  $v$  is a vertex whose neighborhood is a clique. A *simplicial elimination ordering* for a graph  $G$  is an ordering for a (subset of the) vertex-set  $\{v_1, \dots, v_t\} \subseteq V(G)$  such that vertex  $v_i$  is simplicial in  $G \setminus \{v_1, \dots, v_{i-1}\}$ . *Irreducible cycles* are cycles without chords. *Holes* are irreducible cycles of length  $\geq 4$ . The graph  $G$  is called *chordal* if it contains no holes. It is well known a graph  $G$  is chordal if and only if  $G$  has a simplicial elimination ordering that uses all the vertices (e.g. [13]).

The notation for *adding an edge*  $e_{vw} = \{v, w\} \in E(G)$  to the graph  $G$  is  $G + e_{vw}$ . It is also possible to *merge two vertices*  $v, w \in V(G)$ . The new vertex  $x_{vw}$  (or  $vw$ ) has the combined neighborhood:  $N_G(x_{vw}) := (N_G(v) \setminus \{w\}) \cup (N_G(w) \setminus \{v\})$ . The notation for merging two vertices in  $G$  is:  $G/vw$ . The vertex-set will be  $V(G/vw) = (V(G) \cup \{x_{vw}\}) \setminus \{v, w\}$ .

*Perfect graphs*  $G$  are graphs such that  $\chi(H) = \omega(H)$  for every subgraph  $H$  of  $G$ . Notice that a perfect graph  $G$  can not contain an odd cycle.

A *matching*  $M$  of a graph  $G$  is a subset of its edges such that no vertex in  $G$  is an endpoint of two or more edges in  $M$ . An *augmenting path* of a matching  $M$  in graph  $G$ , is a path  $P$  in  $G$  such that edges in  $P$  alternate being in  $M$ , starting and ending with an edge not in  $M$ .

# Chapter 1

## Coloring circular-arc graphs

### 1.1 Interval graphs

**Definition 1.1.** A graph  $G = (V, E)$  is an *interval graph* if there exist a set  $\mathcal{I} := \{I_v\}_{v \in V}$  of open intervals  $I_v = (a, b)$  on the real line with  $a < b$  and a bijection  $f : V \rightarrow \mathcal{I}$  such that for  $v \neq w$

$$\{v, w\} \in E \iff f(v) \cap f(w) \neq \emptyset.$$

We call  $\{I_v\}_{v \in V}$  an interval representation of  $G$ .

An example is given in Figure 1.1. There is a simple linear-time algorithm to recognize interval graphs [12], given by N. Korte and R. Möhring.

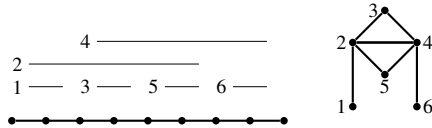


Figure 1.1: On the left an interval model of the interval graph on the right.

**Theorem 1.2.** [13] Interval graphs are chordal.

*Proof.* [13] Take an interval graph  $G = (V, E)$  with interval representation  $\{I_v\}_{v \in V}$  where  $I_v = (a_v, b_v)$ . Assume  $(1, \dots, k)$  is a hole in graph  $G$ , then without loss of generality:  $a_1 < b_1 \leq a_3 < b_3$ . Because  $I_2 \cap I_1 \neq \emptyset$  and  $I_2 \cap I_3 \neq \emptyset$ , we have  $a_2 < b_1 \wedge a_3 < b_2$ . Now make an interval

$$J := \bigcup_{i \in \{4, \dots, k\}} I_i.$$

This is an interval, because  $i$  is connected to  $i+1$ . This interval has overlap with  $I_1$  and with  $I_3$ , but not with  $I_2$  which is impossible.  $\square$

With the above theorem, we can color interval graphs in polynomial time, because we can color any chordal graphs in polynomial time (See COLORCHORDAL, Algorithm 13 in appendix). In this thesis we will focus more on circular-arc graphs.

### 1.2 Introduction to circular-arc graphs

**Definition 1.3.** A graph  $G = (V, E)$  is a *circular-arc graph* if there exist a set  $\mathcal{A} := \{A_v\}_{v \in V}$  of open arcs  $A_v = (a, b)$  on a circle with circumference  $r$  (the arc is from  $a$  to  $b$  in the clockwise direction with  $a, b \in [0, r)$ ) and a bijection  $f : V \rightarrow \mathcal{A}$  such that for  $v \neq w$

$$\{v, w\} \in E \iff f(v) \cap f(w) \neq \emptyset.$$

We call  $\{A_v\}_{v \in V}$  a circular-arc representation of  $G$ .

Interval graphs are examples of circular-arc graphs. The opposite is false, because circular-arc graphs can have holes where interval graphs must be chordal. See Figure 1.2 for an example of a circular-arc graph and its model.

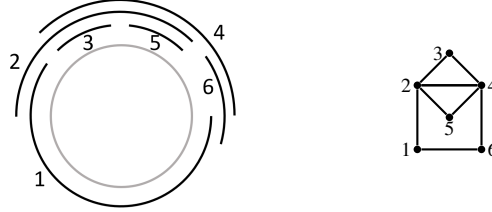


Figure 1.2: On the left a circular-arc model of the graph on the right

In 1980 a recognition algorithm was given by Tucker that could recognize circular-arc graphs in  $\mathcal{O}(n^3)$  time [24]. This algorithm has been simplified to an  $\mathcal{O}(n^2)$  time algorithm [5] and further optimized to an  $\mathcal{O}(n)$  time algorithm [15]. If a graph  $G$  is a circular-arc graph, then all algorithms will produce a circular-arc representation of  $G$ .

### 1.3 Coloring circular-arc graphs in general

The question that arises is whether we can color a circular-arc graph  $G$  in polynomial time. A vertex coloring of  $G$  corresponds uniquely to an arc coloring of its circular-arc model. We will often use an arc coloring when we want to color the vertices.

#### ARC-COLORING PROBLEM

Given: A family  $A$  of arcs and an integer  $k \in \mathbb{N}$

Question: Can we color the arcs with  $k$  colors such that intersecting arcs have different colors?

M.R. Garey, D.S. Johnson, G.L. Miller and C.H. Papadimitriou proved in 1980 that the arc-coloring problem is NP-complete:

**Theorem 1.4.** [6] The Arc-coloring problem is NP-complete.

*Proof.* We give the proof from [6] which is a reduction proof from the word problem for products of symmetric groups (WPPSG). We start by defining the WPPSG problem.

#### WPPSG PROBLEM

Given:  $K \in \mathbb{N}$ ,  $m$  subsets  $X_1, \dots, X_m \subseteq \{1, \dots, K\}$  and a permutation  $\pi \in S_K$

Question: Can  $\pi$  be written as  $\pi = \pi_m \cdot \dots \cdot \pi_1$  where  $\pi_i$  is a permutation of  $\{1, \dots, K\}$  that leave all elements in  $\{1, \dots, K\} \setminus X_i$  fixed.

The WPPSG problem is proven to be NP-complete in the same article [6].

Recall  $\pi_{i+1} \cdot \pi_i$  is the permutation that first applies  $\pi_i$  and then  $\pi_{i+1}$ . Let  $X_{l_i[1]}, \dots, X_{l_i[r(i)]}$  be all the sets that contain  $i$ , i.e. there are  $r(i)$  different sets that contain  $i$ . First, we define a map  $f$  from the WPPSG problem to the Arc-Coloring problem:

$$\left\{ \begin{array}{l} K \in \mathbb{N} \\ \pi \in S_K \\ X_1, \dots, X_m \\ \text{with } \forall i \exists j [i \in X_j] \end{array} \right. \mapsto \left\{ \begin{array}{l} K \text{ colors} \\ \text{Arcs } F := \bigcup_{i=1}^K (F_i \cup C_i) \text{ with} \\ F_i := (i, K + l_i[1]) \cup \{(K + l_i[j - 1], K + l_i[j]) : 2 \leq j \leq r(i)\} \\ C_i := (K + l_i[r(i)], \pi^{-1}(i)). \end{array} \right.$$



The arcs in  $F$  are in the interval  $[0, K+m)$ . If there exists an  $i$  such that  $i \notin X_j$  for all  $j$ , then either  $\pi(i) \neq i$ , we make  $f(K, X_1, \dots, X_m, \pi) = (\cup_{i=1}^2(0, 1), k=1)$  for which the answer is ‘no’. Or,  $\pi(i) = i$  so as new input, we decrease all integers larger than  $i$  with 1.

We can compute  $f$  in polynomial time. What is left to do, is to prove

$$\pi = \pi_m \cdot \dots \cdot \pi_1 \iff f(K, X_1, \dots, X_m, \pi) \text{ is } K\text{-colorable.}$$

Let  $K \in \mathbb{N}$ ,  $X_1, \dots, X_m \subseteq \{1, \dots, k\}$ ,  $\pi \in S_K$  be the input of  $f$  such that  $F, K$  is the output. An example where some arcs are already colored, can be found in Figure 1.3. We want to see when we can color this graph with  $K$  colors.

First we make a model  $F'$  of arcs that will help us to properly define a permutation. This model  $F'$  will have a one-to-one correspondence with  $F$ . The arcs in  $C$  will be split into two parts and placed in  $F'$ :  $F'_i := F_i \cup (K + l_i[r(i)], 0) \cup (0, i)$  and  $F' := \cup_{i=1}^K F'_i$ . With this construction, for  $0 \leq p < K+m$ , the point  $p + \frac{1}{2}$  is in  $K$  arcs, none of those arcs are in the same set  $F'_i$ . Now we can define properly  $\sigma_p(j) = i$  to be the  $j$ -th color assigned to the unique arc in  $F'_i$  that contains the point  $p + \frac{1}{2}$ . For every  $p$ ,  $\sigma_p$  is a map from  $\{1, \dots, K\}$  to itself and therefore a permutation. Without loss of generality, for  $i \leq K$  we set  $\sigma_0(i) = i$ . To be able to go back to  $F$ , we need

$$\sigma_{K+m}(i) = \pi(i). \tag{1.1}$$

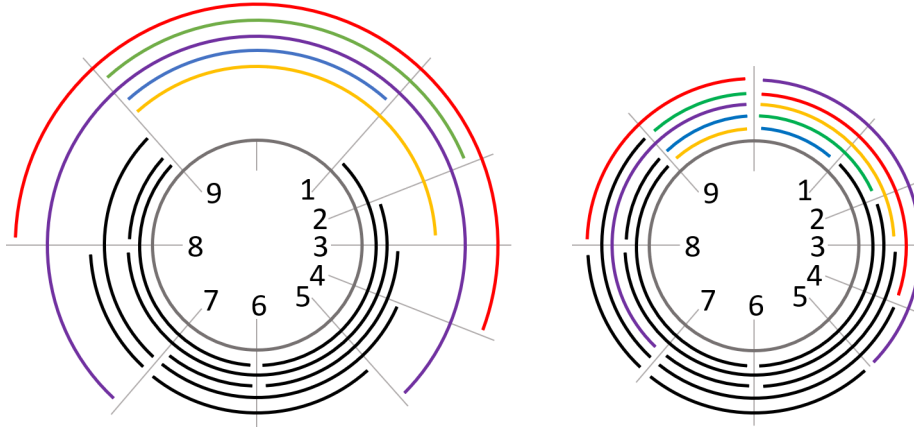


Figure 1.3: Example [6].  $K = 5$ ,  $\pi = (12453)$ ,  $X_1 = \{1, 3\}$ ,  $X_2 = \{3, 4, 5\}$ ,  $X_3 = \{2, 5\}$ ,  $X_4 = \{1, 2, 4\}$ ,  $m = 4$ . On the left  $F = \cup_{i=1}^5 (F_i \cup C_i)$ , on the right  $F'$

Notice that for the points  $p \leq K$  the permutation  $\sigma_p$  is already fixed:  $\sigma_p(i) = \sigma_0(i) = i$ . This can be easily seen with induction to  $p$ . If  $\sigma_p$  is known, we can make  $\sigma_{p+1}$ . Sometimes an arc continues and therefore the assigned color is fixed, i.e.  $\sigma_p(j) = \sigma_{p+1}(j)$ . At other times we can choose between the colors that are not assigned by the obligation. We have

$$\sigma_{p+1}(j) \begin{cases} = \sigma_p(j) & \text{if } p, p+1 \in A \in F'_{\sigma_p(j)} \text{ for some } A, \\ \in \{\sigma_p(j) \mid (a, p+1) \in F'_{\sigma_p(j)} \text{ for some } a\} & \text{otherwise.} \end{cases}$$

By construction, when  $p \in \{K+1, \dots, K+m\}$ , the set of integers to choose from in the second case is always  $X_{p-K}$ . Therefore,  $\sigma_{p+1} = \pi_{p-K} \cdot \sigma_p$ . All others should stay the same. In our example,  $X_{6-5} = X_1 = \{1, 3\}$  and thus  $\sigma_6(1) \in \{1, 3\}$ . In total, we have

$$\pi(i) \stackrel{(1.1)}{=} \sigma_{K+m}(i) = \pi_m \cdot \dots \cdot \pi_1 \cdot id_{\{1, \dots, K\}}.$$

And therefore

$$\pi = \pi_m \cdot \dots \cdot \pi_1 \iff f(K, X_1, \dots, X_m, \pi) \text{ is } K\text{-colorable.}$$

The arc-coloring problem is in NP, because in  $\mathcal{O}(n^2)$  time, we can check if the colors assigned to the endpoints of each edge in  $G$  are different. We keep track of the used colors in linear time. Both tests are polynomial and if successful the graph can be colored with  $K$  colors.  $\square$

Theorem 1.4 tells us a polynomial time algorithm to color circular-arc graphs in general is not known. However, through the WPPSG-problem, there exists an  $\mathcal{O}(n \cdot k! \cdot k \log k)$  time algorithm to answer the question of whether a circular-arc graph with  $n$  vertices is  $k$ -colorable [3]. This algorithm is polynomial (even linear) if  $k$  is fixed. The following problem is therefore in P:

#### ARC- $K$ -COLORING PROBLEM

Given: A family  $A$  of arcs.

Question: Can we color the arcs with  $K$  colors such that intersecting arcs have different colors?

Some subclasses of circular-arc graphs have polynomial time algorithms to color the graphs from the subclasses optimally where the number of colors needed is not fixed. In the next two sections, we will discuss examples of such subclasses.

## 1.4 Proper circular-arc graphs

**Definition 1.5.** A graph  $G = (V, E)$  is a *proper circular-arc graph* if there exists a circular-arc representation  $A = \{A_1, \dots, A_n\}$  of  $G$  such that no arc in  $A$  contains another arc, i.e.

$$\forall A_i, A_j \in A \ [i \neq j \Rightarrow \neg(A_i \subseteq A_j \vee A_j \subseteq A_i)].$$

There is also a notion of *unit circular-arc graphs* in the literature (first introduced by Tucker [22]). These are graphs that have a circular-arc representation where all arcs have the same length. All unit circular-arc graphs are proper circular-arc graphs.

There exists a  $\mathcal{O}(n^2)$  time algorithm for recognizing proper circular-arc graphs and creating an associated proper circular-arc representation [21].

J.B. Orlin, M.A. Bonuccelli and D.P. Bovet gave a  $\mathcal{O}(n^2 \log n)$  time algorithm to optimally color proper circular-arc graphs [17]. First they translate the problem of proper coloring a circular-arc graph  $G$  with  $k$  colors (if possible) to a linear integer programming problem. Normally, there is no polynomial time algorithm known to determine a feasible solution for a linear integer programming problem. With this set of equations we *can* have a  $\mathcal{O}(n^2)$  time algorithm to solve it. A short description how this is done, can be found in Theorem 1.7. After a solution to the system is given, we can color the vertices.

**Theorem 1.6.** [17] Let  $G = (V, E)$ ,  $n := \#V$  be a proper circular-arc graph with an associated proper circular-arc representation  $A = \{A_1, \dots, A_n\}$  such that  $A_i = [a_1, b_i]$  and  $a_1 < \dots < a_n$ . Then  $G$  may be  $k$ -colored if and only if there is a feasible solution to the following system with  $r := n \bmod k$  and  $x_0, \dots, x_n$  are integer valued variables:

$$\begin{aligned} & \text{for } i = 1, \dots, n \quad \begin{cases} 0 \leq x_i - x_{i-1} \\ x_0 = 0 \\ x_n = r \cdot \lceil n/k \rceil \end{cases} \\ & \text{for } S := \{A_{i+1} \dots A_j\} \text{ is a maximal} \quad \begin{cases} x_j - x_i \leq r \\ (j-i) - (x_j - x_i) \leq k - r \end{cases} \\ & \text{clique in } G \text{ and } i < j. (|S| = j - i) \\ & \text{for } S := \{A_{i+1} \dots A_j\} \text{ is a maximal} \quad \begin{cases} x_n + x_j - x_i \leq r \\ (n+j-i) - (x_n + x_j - x_i) \leq k - r \end{cases} \\ & \text{clique in } G \text{ and } i > j. (|S| = n + j - i) \end{aligned}$$

*Proof.* This proof is inspired by and sometimes incorporates proofs of [17]. First we notice that every maximal clique  $S$  of  $G$  is of the form  $S = \{A_i, A_{i+1}, \dots, A_j\}$  for some  $i, j$ . This holds because

a maximal clique has one or more common points on the circle and therefore, it is easy to see that  $a_j := \max_l \{a_l : A_l \in S\}$  or  $b_i := \min_l \{b_l : A_l \in S\}$  is a common point that induces the clique:

$$a_j \in \bigcap_{A \in S} A \quad \vee \quad b_i \in \bigcap_{A \in S} A$$

for otherwise  $A_i$  and  $A_j$  do not overlap. If this common point is equal to 0, then  $S = \{A_i : a_i > b_i \text{ or } a_i = 0\}$ . If this common point is greater than 0, we can rotate the circle and fix the arcs such that the common point is equal to 0 and apply the same argument.

( $\Leftarrow$ )

Assume there exists a feasible solution to the system described in Theorem 1.6. We define the set

$$W := \{A_i : x_i - x_{i-1} = 1\}.$$

We will interpret for  $i < j$ ,  $x_j - x_i = |W \cap \{A_{i+1}, \dots, A_j\}|$  and for  $i > j$ ,  $x_i - x_j = |W \cap \{A_{i+1}, \dots, A_j\}|$ . Thus  $|W| = x_n - x_0 = r \cdot \lceil n/k \rceil - 0$ . We use  $l(i)$  to denote the integer such that  $W$  has  $i-1$  arcs  $A_j$  with  $a_j < a_i$ . The intuition is that the arc  $A_i$  is the  $l(i)$ -th arc in  $W$ . Consider the coloring  $c$  of  $G[W]$  with the colors  $1, \dots, r$  such that  $c(A_i) := l(i) \bmod r$ . Thus, we sequentially color the arcs. The ‘last’ arc in  $W$  has color  $r$ . Let  $c(A_i) = c(A_j)$  with  $i < j$ .

- i If  $b_i \in A_j$ , then  $A_{l(i)}, A_{l(i+1)}, \dots, A_{l(j)}$  is in a maximal clique  $S$  with  $|S \cap W| \geq r+1$  for otherwise  $c(A_i) \neq c(A_j)$ . But this is in contradiction with the assumption we had a feasible solution ( $|S \cap W| = x_j - x_{i-1} \leq r$  must hold). Thus  $b_i \notin A_j$ .
- ii If  $a_j \in A_i$ , then  $A_j, \dots, A_i$  is in a maximal clique. An analogous argument will conclude  $a_j \notin A_i$ .

Therefore,  $A_i \cap A_j = \emptyset$ , so the coloring  $c$  is proper. Analogously we can color  $G[V \setminus W]$  with  $k-r$  colors.

( $\Rightarrow$ )

Assume the graph  $G$  can be colored with  $k$  colors.

**Claim 1.**  $G$  can be colored with  $k$  colors in such a way that each color class has either  $\lceil n/k \rceil$  or  $\lfloor n/k \rfloor$  colors. *Proof of Claim 1:* [17] We will prove this with induction to  $k$ . Assume  $k=2$ . If  $n$  is even, then we can color the arcs of  $G$  alternately. If  $n$  is odd and a color class  $C$  is of size  $\geq (n+2)/2$ , then there exists an  $i$  such that  $A_i, A_{i+1} \in C$ . It follows  $C' = \{\dots, A_{i-4}, A_{i-2}, A_i, A_{i+1}, A_{i+3}, \dots\}$  nor  $A \setminus C'$  has two overlapping arcs and consequently are proper coloring classes of size  $\lceil n/2 \rceil$  and  $\lfloor n/2 \rfloor$  respectively. If  $k > 2$  we can iterative recolor  $G[C \cup D]$  with  $C, D$  the color classes of smallest and largest size.

Let  $c$  be a  $k$ -coloring of  $G$  such that every color class has  $\lceil n/k \rceil$  or  $\lfloor n/k \rfloor$  colors and let  $r = n \bmod k$ . If  $r = 0$ , then  $x_i = 0$  for every  $i$ , is a feasible solution. If  $r > 0$ , then we take a union of coloring classes  $W := \bigcup_{|C|=\lceil n/k \rceil} C$  and assign  $x_i := |\{A_1, \dots, A_i\} \cap W|$ . This is a feasible solution.  $\square$

**Theorem 1.7.** [17] A proper circular-arc graph  $G$  can be colored with  $\chi(G)$  colors in  $\mathcal{O}(n^2 \log n)$  steps.

*Proof.* Let  $G$  be a graph. In  $\mathcal{O}(n^2)$  time an associated proper circular-arc representation is given if possible [22]. If such representation does not exist, it will output false. Otherwise, for a given  $k$ , we can create the constraints as described in Theorem 1.6. A feasible solution to the system can be determined in  $\mathcal{O}(n^2)$  steps: First a directed, weighted graph  $G'$  is made with the vertices  $v_0, \dots, v_{n-1}$  and for every constraint  $x_j - x_i \leq w_{ij}$  we construct the edges are  $(v_i, v_j)$  with weight  $w_{ij}$ . If for every  $i$ , the shortest distance from  $v_0$  to  $v_i$  is positive, then we assign the distance  $d_{0i}$  the the variable  $x_i$ . Otherwise, there is no feasible solution to the system. There are at most  $2n$  points that induce different cliques and therefore at most  $5n$  edges. Thus, these distances can be computed in  $\mathcal{O}(n^2)$  time by the Bellman-Ford method [17].

Using binary search, we can determine  $k$  in  $\mathcal{O}(\log n)$  steps. At every step, we compute if the solutions for a  $k$ . In total, we need  $\mathcal{O}(n^2 \log n)$  steps.

When we have a feasible solution of minimal  $k$  to the system, we can determine the coloring of  $G$  in  $\mathcal{O}(n)$  steps. First we color  $G[\{A_i : x_i - x_{i-1} = 1\}]$  with  $1, \dots, (n \bmod k)$  colors by sequentially color its arcs. We color all other arcs sequentially with the colors  $(n+1 \bmod k), \dots, k$ .  $\square$

## 1.5 Perfect circular-arc graphs

Perfect circular-arc graphs are circular-arc graphs which are perfect, i.e.  $\chi(H) = \omega(H)$  for every induced subgraph  $H$ . These graphs can be recognised in  $\mathcal{O}(mn \log \log n + m^2)$  time [4]. If the graph is a perfect circular-arc graph, we will also obtain a circular-arc model representing the graph. For perfect circular-arc graphs there also exists a polynomial time algorithm to compute its coloring [25]. We will only give a short description of the algorithm.

**Theorem 1.8.** Perfect circular-arc graphs can be colored in  $\mathcal{O}(n^2m)$  time.

*Proof.* The following is a short description of the algorithm given in [25]. Let  $G$  be a perfect circular-arc graph and  $A = \{A_1, \dots, A_n\}$  the associated arc model such that  $A_i = (a_i, b_i)$  and the length of arc  $A_i$  is greater than or equal to the length of arc  $A_j$  for all  $j > i$ . Recall a circular-arc representation of  $G$  can be found in  $\mathcal{O}(n)$  time [15] and an ordering can be realized in  $\mathcal{O}(n \log n)$  time (e.g. by merge sort). We will color the arcs one at a time.

Assume the algorithm already gave a coloring  $c$  for the vertices  $\{v_1, \dots, v_l\}$  and now we want to create a coloring  $c'$  for the vertices  $\{v_1, \dots, v_{l+1}\}$ . We define  $H := G[\{v_1, \dots, v_l\}]$  and  $H_0 := G[N_H(v_{l+1})]$ .

All arcs that have overlap with  $A_{l+1}$  contain point  $a_{l+1}$  or  $b_{l+1}$ , because the length of the arc  $A_{l+1}$  is smaller than or equal to those of the arcs representing  $H$  (and also  $H_0$ ). Therefore  $H_0$  can be covered by two cliques (namely, the cliques induced by  $a_{l+1}$  or  $b_{l+1}$ ) and so  $\overline{H_0}$  is bipartite.

We construct a matching  $M$  in  $\overline{H_0}$  by  $\{v, w\} \in M$  if and only if  $c(v) = c(w)$ . This is a correct matching, because  $c$  is a proper coloring of  $H_0$  and  $H_0$  can be covered by two cliques, so every color in  $c$  is assigned to at most two vertices. Hsu [10] proved the following statement:

**Claim 1.** If  $H_0$  has used  $\omega(G)$  colors, then  $M$  is not a maximum matching in  $\overline{H_0}$ .

In  $\mathcal{O}(n^2)$  steps we can find an augmenting path (or correctly state  $M$  is maximal) [25]. If  $M$  is maximal, we assign a random color to  $v_{l+1}$  not assigned yet to its neighbors. Thus  $c'|_{V(H)} = c$  and  $c'(v_{l+1}) \notin c(V(H_0))$ . If  $M$  is not maximal, we found  $P$ : an augmenting path.

If this path is an edge  $\{v, w\} \in E(\overline{H_0})$ , then  $c_v := c(v) \neq c(w) =: c_w$  and no other vertices in  $H_0$  are colored with  $c_v$  or  $c_w$ . Define

$$T_{H,w}^{c_v c_w} := \{u : (c(u) = c_v \text{ or } c(u) = c_w) \text{ and } \exists P_{wu} = (w, p_2, \dots, u) \text{ a path from } w \text{ to } u\}.$$

The new coloring  $c'$  will be equal to  $c$ , but with one small change: the colors of the component  $T_{H,w}^{c_v c_w}$  are switched in  $c'$ . If  $c'(v) \neq c(v)$ , then there exists a shortest path  $P_{vw}$  from  $w$  to  $v$  and  $P_{vw} \cup \{v_{l+1}\}$  is an odd cycle. This contradicts  $G$  being perfect. Therefore, we can expand the coloring  $c'$  by assigning  $c'(v_{l+1}) := c(w)$ .

If this augmenting path  $P$  consists of more than one edge, we can shorten the path in  $\mathcal{O}(n+m)$  steps by switching colors and changing the matching  $M$  until we have an edge. There are two color switch possibilities. It depends on the edges in  $\overline{H_0}$  which one we apply. We need to do this at most  $\mathcal{O}(n)$  times.

The details and the complete proof can be found in [25]. □

## Chapter 2

# Chromatic polynomial for hyperholes

This chapter is all about finding the chromatic polynomial of a hyperhole. Therefore, we need to know what a chromatic polynomial is and what a hyperhole is. The definitions are in line with [18] and [14].

**Definition 2.1.** The *chromatic function* of a graph  $G$  is the function  $P_G : \mathbb{N} \rightarrow \mathbb{N}$  that gives the number of ways of coloring the vertices of a graph  $G$  with input the number of available colors.

Notice that  $P_G(\chi(G)) > 0$  and for all  $x \in \{0, \dots, \chi(G) - 1\}$  we have  $P_G(x) = 0$ . The vertices in the graph are interpreted as fixed on a piece of paper or as labeled. In Figure 2.1 we see two different colorings of the graph using the colors {red, blue, yellow}.

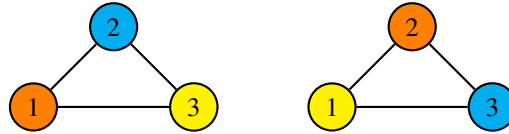


Figure 2.1: Two different colorings of the same graph

There are three important families of graphs where the chromatic function is known that we are going to use to calculate the chromatic function of a hyperhole. The chromatic function can be computed for these families using induction on the number of vertices. We will only give a proof sketch.

**Lemma 2.2.** The chromatic function of an edgeless graph  $\overline{K}_n$  on  $n$  vertices is

$$P_{\overline{K}_n}(\lambda) = \lambda^n.$$

*Proof.* The graph is edgeless, so if we want to color a vertex it does not matter how another vertex is colored for it to be a proper coloring. Therefore  $P_{\overline{K}_n}(\lambda) = \lambda^n$ .  $\square$

**Lemma 2.3.** The chromatic function of a path  $P_n$  on  $n$  vertices is

$$P_{P_n}(\lambda) = \lambda(\lambda - 1)^{n-1}.$$

*Proof.* Take a path  $P_n = (v_1, \dots, v_n)$ . For the first vertex  $v_1$  we have  $\lambda$  options to color this vertex. The second vertex  $v_2$  has only  $\lambda - 1$  options to be colored, because it cannot be colored with the same color as the one before. Do this inductively and we see that  $P_{P_n}(\lambda) = \lambda(\lambda - 1)^{n-1}$ .  $\square$

**Lemma 2.4.** The chromatic function of a complete graph  $K_n$  on  $n$  vertices is

$$P_{K_n}(\lambda) = \lambda(\lambda - 1) \cdots (\lambda - (n - 1)) = \prod_{i=0}^{n-1} (\lambda - i).$$

*Proof.* All vertices have to be colored differently. At the first vertex we can choose from  $\lambda$  colors, the second from  $\lambda - 1$  colors, etcetera.  $\square$

## 2.1 Fundamental theorems for chromatic polynomials

A trick can be used to calculate a chromatic function for any graph. The trick is to add or delete edges from the graph until we find a graph of which we know the chromatic function.

**Theorem 2.5.** Let  $G = (V, E)$  be a graph and  $u, v \in V$  be two vertices with no edge between them. Then

$$P_G(\lambda) = P_{G+e_{uv}}(\lambda) + P_{G/uv}(\lambda).$$

*Proof.* The number of colorings in graph  $G$  is equal to the number of colorings of graph  $G$  where the vertices  $u$  and  $v$  have a different color *plus* the case where the vertices  $u$  and  $v$  have the same color. The first one is exactly  $P_{G+e_{uv}}(\lambda)$ . By adding an edge, the colors of  $u$  and  $v$  have to be different. The second one is  $P_{G/uv}(\lambda)$ . Merging the vertices  $u$  and  $v$  is analogous to assigning the same color to  $u$  and  $v$ .  $\square$

**Corollary 2.6.** Let  $G = (V, E)$  be a graph and  $u, v \in V$  be neighbors. Then

$$P_G(\lambda) = P_{G-e_{uv}}(\lambda) - P_{G/uv}(\lambda).$$

*Proof.* We will use Theorem 2.5 on the graph  $G - e_{uv}$ :

$$\begin{aligned} P_{G-e_{uv}}(\lambda) &= P_{G-e_{uv}+e_{uv}}(\lambda) + P_{G/uv}(\lambda) \\ P_{G-e_{uv}}(\lambda) &= P_G(\lambda) + P_{G/uv}(\lambda) \\ P_{G-e_{uv}}(\lambda) - P_{G/uv}(\lambda) &= P_G(\lambda). \end{aligned}$$

$\square$

**Theorem 2.7.** The chromatic function of a simple graph is a polynomial with integer coefficients:  $P_G(\lambda) = \lambda^n - a_{n-1}\lambda^{n-1} + a_{n-2}\lambda^{n-2} + \dots + (-1)^n a_2 \lambda^2 - (-1)^n a_1 \lambda$  where  $n$  is the number of vertices in  $G$  and for all  $i \leq n$  we have  $a_i \in \mathbb{N}_{\geq 0}$ .

*Proof.* We will give a proof with induction to the number of edges. The theorem holds for an edgeless graph, because  $P_{K_n}(\lambda) = \lambda^n$ . Assume it holds for a simple graph with  $m$  edges. Take a simple graph  $G$  with  $m+1$  edges and  $n$  vertices. We use Corollary 2.6 to delete an edge  $e_{uv} \in E(G)$ . Note  $G/uv$  has strictly fewer edges than  $G$  and hence we can apply the induction hypothesis two times:

$$\begin{aligned} P_G(\lambda) &\stackrel{2.6}{=} P_{G-e_{uv}}(\lambda) - P_{G/uv}(\lambda) \\ &\stackrel{IH}{=} \lambda^n - a_{n-1}\lambda^{n-1} + \dots + (-1)^{n+1} a_1 \lambda - (\lambda^{n-1} - a'_{n-2}\lambda^{n-2} + \dots + \lambda^2 + (-1)^n a'_1 \lambda) \\ &= \lambda^n - (a_{n-1} + 1)\lambda^{n-1} + (a_{n-2} + a'_{n-2})\lambda^{n-2} + \dots + (-1)^n (a_1 + a'_1)\lambda \end{aligned}$$

All  $a_i$  and  $a'_i$  are in  $\mathbb{N}$ . With induction we have proven our theorem.  $\square$

From now on we say chromatic polynomial instead of chromatic function.

When a graph has a clique-cut-partition  $(A, B, C)$ , we can use it to calculate the chromatic polynomial of the whole graph by calculating the chromatic polynomial of two parts  $G[A \cup C]$  and  $G[B \cup C]$ . Recall,  $C$  is a clique and  $A$  is anticomplete to  $B$ .

**Theorem 2.8.** If  $(A, B, C)$  is a clique-cut-partition of a graph  $G$ , then

$$P_G(\lambda) = \frac{P_{G[A \cup C]}(\lambda) \cdot P_{G[B \cup C]}(\lambda)}{P_{G[C]}(\lambda)}.$$

*Proof.* There are  $P_{G[C]}(\lambda)$  number of ways to color the common part  $G[C]$ . If we fix the colors of this part, we have  $P_{G[A \cup C]}(\lambda)/P_{G[C]}(\lambda)$  ways to color  $G[A]$  and  $P_{G[B \cup C]}(\lambda)/P_{G[C]}(\lambda)$  for  $G[B]$ . It brings a number of colorings equal to

$$P_G(\lambda) = P_{G[C]}(\lambda) \cdot \frac{P_{G[A \cup C]}(\lambda)}{P_{G[C]}(\lambda)} \cdot \frac{P_{G[B \cup C]}(\lambda)}{P_{G[C]}(\lambda)} = \frac{P_{G[A \cup C]}(\lambda) \cdot P_{G[B \cup C]}(\lambda)}{P_{G[C]}(\lambda)}$$

$\square$

To simplify notation, we will remove the ‘ $P$ ’ to denote the chromatic polynomial of a graph  $G$ . From now on, we will write  $G(\lambda)$  for the chromatic polynomial of  $G$  with variable  $\lambda$ . The notation for  $G$  can be a drawing of the graph or letters that represent a special graph.

## 2.2 Hyperpaths

We can visualise a hyperpath by taking a path and blowing up each vertex to a nonempty clique of arbitrary size.

**Definition 2.9.** A *hyperpath*  $G = (V, E)$  is a graph whose vertex set can be partitioned into  $k \geq 3$  nonempty cliques  $X_1, X_2, \dots, X_k$  such that for all  $i \in \{2, \dots, k-1\}$ ,  $X_i$  is complete to  $X_{i-1} \cup X_{i+1}$ , anticomplete to all  $X_j$  with  $j \notin \{i-1, i, i+1\}$  and  $X_1$  is anticomplete to  $X_k$ . We say that  $G$  is a  $k$ -*hyperpath* and the *length* of  $G$  is equal to  $k$ . Another notation for  $G$  is  $[X_1, \dots, X_k]_P$  where  $X_i$  are subsets of  $V(G)$ , or as  $[x_1, \dots, x_k]_P$  where  $x_i := \#X_i$ .

Figure 2.2 contains an example of a hyperpath. The hyperpath is equal to  $[1, 4, 3, 2, 3, 1]_P$ . Notice hyperpaths are examples of interval-graphs and an interval model of the hyperpath is given on the left of Figure 2.2. To simplify our drawings, we will use a circle with a number  $n$  to denote a clique of size  $n$ . We will use an ‘ $=$ ’-sign to denote that two (sub)graphs are complete to each other.

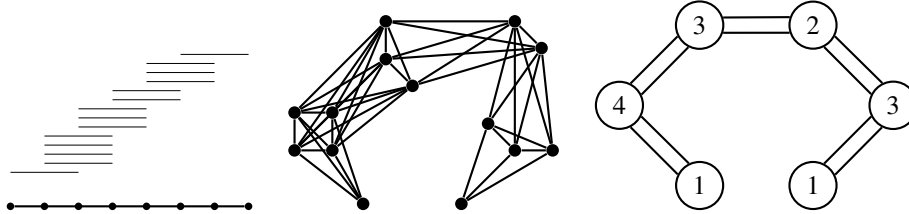


Figure 2.2: On the left an interval-graph model of the hyperpath  $[1, 4, 3, 2, 3, 1]_P$  as shown in the middle. On the right a schematic model of the same hyperpath. The length of the hyperpath is equal to 6.

With the following proposition (Proposition 2.10), we can calculate that its chromatic polynomial is

$$\begin{aligned} & [1, 4, 3, 2, 3, 1]_P(\lambda) \\ &= \lambda(\lambda-1)(\lambda-2)(\lambda-3)(\lambda-4)(\lambda-4)(\lambda-5)(\lambda-6) \\ & \quad \cdot (\lambda-3)(\lambda-4)(\lambda-2)(\lambda-3)(\lambda-4)(\lambda-3) \\ &= \lambda(\lambda-1)(\lambda-2)^2(\lambda-3)^4(\lambda-4)^4(\lambda-5)(\lambda-6). \end{aligned}$$

**Proposition 2.10.** Let  $G = [x_1, \dots, x_k]_P$  be a hyperpath. To simplify notation, we define  $x_0 := 0$ . The chromatic polynomial is

$$[x_1, \dots, x_k]_P(\lambda) = \prod_{i=1}^k \prod_{j=0}^{x_i-1} (\lambda - x_{i-1} - j).$$

*Proof.* We will prove it with induction on the length of the path.

BASE:

Let  $G = [X_1, X_2, X_3]_P$  be a hyperpath of length three and define  $x_i := \#X_i$ , then  $(X_1, X_3, X_2)$  is a clique-cut-partition of  $P$ . Theorem 2.8 gives us

$$G(\lambda) = \frac{G[X_1, X_2](\lambda) \cdot G[X_2, X_3](\lambda)}{G[X_2](\lambda)}.$$

Because  $X_1$  is complete to  $X_2$  and  $X_2$  is complete to  $X_3$ , we know that  $G[X_1, X_2], G[X_2, X_3], G[X_2]$  are cliques. Lemma 2.4 gives us

$$\begin{aligned}
\frac{G[X_1, X_2](\lambda) \cdot G[X_2, X_3](\lambda)}{G[X_2](\lambda)} &= \frac{\left(\prod_{j=0}^{x_1+x_2-1} (\lambda - j)\right) \left(\prod_{j=0}^{x_2+x_3-1} (\lambda - j)\right)}{\prod_{j=0}^{x_2-1} (\lambda - j)} \\
&= \frac{\left(\prod_{j=0}^{x_1+x_2-1} (\lambda - j)\right) \left(\prod_{j=0}^{x_2-1} (\lambda - j)\right) \left(\prod_{j=x_2}^{x_2+x_3-1} (\lambda - j)\right)}{\prod_{j=0}^{x_2-1} (\lambda - j)} \\
&= \left(\prod_{j=0}^{x_1-1} (\lambda - j)\right) \left(\prod_{j=x_1}^{x_1+x_2-1} (\lambda - j)\right) \left(\prod_{j=x_2}^{x_2+x_3-1} (\lambda - j)\right) \\
&= \prod_{i=1}^3 \prod_{j=x_{i-1}}^{x_{i-1}+x_i-1} (\lambda - j) = \prod_{i=1}^3 \prod_{j=0}^{x_i-1} (\lambda - x_{i-1} - j).
\end{aligned}$$

INDUCTION STEP:

We want to calculate the chromatic polynomial of  $G = [X_1, \dots, X_{k+1}]_P$ . We define  $x_i := \#X_i$ . Assume the chromatic polynomial of  $[X_1, \dots, X_k]_P$  is equal to  $\prod_{i=1}^k \prod_{j=0}^{x_i-1} (\lambda - x_{i-1} - j)$  where  $x_i := \#X_i$  and  $\#X_0 := 0$ . We know that  $([X_1, \dots, X_{k-1}]_P, X_k, X_{k+1})$  is a clique-cut-partition. The following steps are analogous to the base case, so we will withhold some steps.

$$\begin{aligned}
[X_1, \dots, X_{k+1}]_P(\lambda) &\stackrel{[2.8]}{=} \frac{[X_1, \dots, X_k]_P(\lambda) \cdot G[X_k, X_{k+1}](\lambda)}{G[X_k](\lambda)} \\
&\stackrel{IH}{=} \frac{\left(\prod_{i=1}^k \prod_{j=0}^{x_i-1} (\lambda - x_{i-1} - j)\right) \cdot G[X_k, X_{k+1}](\lambda)}{G[X_k](\lambda)} \\
&= \left(\prod_{i=1}^k \prod_{j=0}^{x_i-1} (\lambda - x_{i-1} - j)\right) \cdot \prod_{t=x_k}^{x_k+x_{k+1}-1} (\lambda - t) \\
&= \prod_{i=1}^{k+1} \prod_{j=0}^{x_i-1} (\lambda - x_{i-1} - j)
\end{aligned}$$

□

**Corollary 2.11.** The chromatic number of a hyperpath  $[X_1, \dots, X_k]_P$  is equal to the maximum clique-size:

$$\chi([X_1, \dots, X_k]_P) = \max_{i < k} (\{x_i + x_{i+1}\}) \quad \text{with } x_i := \#X_i.$$

*Proof.* The proof is derived from colorings of even rings (Lemma 3.2 of [14]). Let  $m := \max_{i < k} (\{x_i + x_{i+1}\})$  and  $c := \{1, \dots, m\}$ . Vertices of  $X_i$  can be colored with the colors  $\{1, \dots, x_i\}$  if  $i$  is odd and with  $\{m - x_i + 1, \dots, x_i\}$  if  $i$  is even. This is a proper coloring, because  $x_i < m - x_{i+1} + 1$  and  $x_{i+1} < m - x_i + 1$  as from how  $m$  is defined. By Proposition 2.10 the graph  $[X_1, \dots, X_k]_P$  can not be colored with less colors. If  $m$  is maximal at  $x_i + x_{i+1}$ , then  $(\lambda - x_i - (x_{i+1} - 1)) = (\lambda - (m - 1))$  is a factor of  $[X_1, \dots, X_k]_P$ . Therefore,  $[X_1, \dots, X_k](m - 1) = 0$ . □

In Figure 2.3 we color the hyperpath  $[1, 4, 3, 2, 3, 1]_P$  with the colors {blue, light blue, green, dark green, purple, yellow, orange}

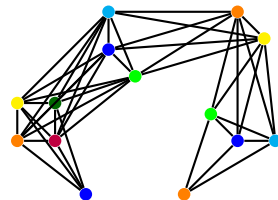


Figure 2.3: A coloring of the hyperpath  $[X_1, \dots, X_k]_P$



## 2.3 Hyperholes

Like hyperpaths we have hyperholes, i.e., a hole (or irreducible cycle of length  $\geq 4$ ) where the vertices are blown up to cliques [2]. For these types of graphs, we wish to determine the chromatic polynomial.

**Definition 2.12.** A *hyperhole*  $G = (V, E)$  is a graph whose vertex set can be partitioned into  $k \geq 4$  nonempty cliques  $X_1, X_2, \dots, X_k$  such that for all  $i \in \{1, \dots, k\}$ ,  $X_i$  is complete to  $X_{i-1} \cup X_{i+1}$  and anti-complete to all  $X_j$  with  $j \notin \{i-1, i, i+1\}$ . Here,  $i$  is taken modulo  $k$ . We say that  $G$  is a  $k$ -*hyperhole* and the *length* of  $G$  is equal to  $k$ . Another notation for  $G$  is  $[X_1, \dots, X_k]_H$  where  $X_i$  are subsets of  $V(G)$ , or as  $[x_1, \dots, x_k]_H$  where  $x_i := \#X_i$ .

From now on when discussing hyperholes we will take indices modulo  $k$  for  $k$  the length of that hyperhole. Notice hyperholes have circular-arc models, see Figure 2.4 for an example of a hyperhole and a circular-arc model of the hyperhole. How such model can be constructed, is discussed in Chapter 3.

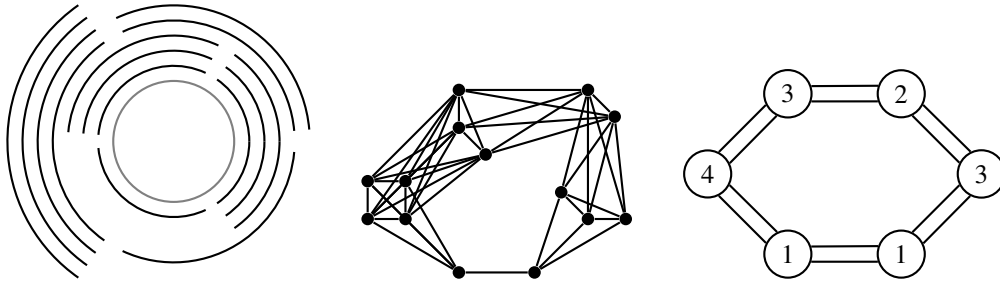


Figure 2.4: On the left a circular-arc model of the hyperhole in the middle. On the right a schematic model of the same hyperpath of length 6.

The final symbol we want to introduce is the ' $\rightsquigarrow_l$ '-symbol. Like the '='-symbol, this new symbol is used between two cliques. For cliques  $X_i$  and  $X_j$  we write  $X_i \rightsquigarrow_l X_j$  if exactly  $l$  vertices in  $X_i$  are complete to  $X_j$  and all other vertices of  $X_i$  are anticomplete to  $X_j$ . See Figure 2.5 for an example where  $X_i \cong K_4$  and  $X_j \cong K_3$  and  $l = 1$ .

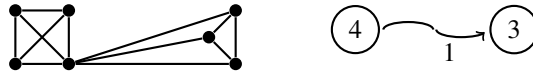


Figure 2.5: Schematic figure on the right represents the graph on the left.

The remainder of this section is dedicated to proving the Chromatic Theorem for hyperholes (Theorem 2.13), stated below.

**Theorem 2.13.** Let  $G = [x_1, \dots, x_k]_H$  be a hyperhole of length  $k \geq 4$ . Without loss of generality, assume  $x_k \geq x_1$ . Indices are not taken modulo  $k$  and we define  $x_0 := 0$ . Then the chromatic polynomial of  $G$  can be calculated recursively:

$$\begin{aligned}
 [x_1, \dots, x_k]_H(\lambda) &= \left( \prod_{i=1}^k \prod_{j=0}^{x_i-1} (\lambda - x_{i-1} - j) \right) \\
 &+ \left( \sum_{i=1}^{x_1} (-1)^i \binom{x_1}{i} \left( \prod_{j=0}^{i-1} (x_k - j) \right) \left( \prod_{j=x_2+i}^{x_1+x_2-1} (\lambda - j) \right) \right. \\
 &\quad \left. \cdot \left( \prod_{j=x_{k-1}+i}^{x_{k-1}+x_k-1} (\lambda - j) \right) \cdot [i, x_2, \dots, x_{k-1}]_H(\lambda) \right)
 \end{aligned}$$

In order to prove the Chromatic Theorem for hyperholes, we wish to prove a lemma that will allow us to remove multiple edges at the same time. Recall that we denote  $e_{uv}$  for an edge between the vertices  $u$  and  $v$ . The notation for removing an edge  $e_{uv}$  of the graph  $G$  is  $G - e_{uv}$  and for contracting the vertices  $u, v$  is  $G/uv$ .

**Lemma 2.14.** Let  $G$  be a graph and let  $X = \{x_1, \dots, x_r\}$  be a clique in  $G$  with vertices that have the same neighbors:  $N_G[x_1] = \dots = N_G[x_r]$ . Let  $v \in N_G[x_1] \setminus X$ , then

$$G/vx_1 \cong (G - e_{vx_1})/vx_2 \cong \dots \cong (G - e_{vx_1} - \dots - e_{vx_{r-1}})/vx_r.$$

*Proof.* We will denote  $V(G/vx_i) := V(G) \cup \{vx_i\} \setminus \{v, x_i\}$ . First we prove  $G/vx_1 \cong (G - e_{vx_1})/vx_2$  by making a bijection  $f$  between the vertices. We take  $f|_{V(G/vx_1) \setminus \{vx_1, x_2\}} := id$ ; the identity map. And  $f(vx_1) = vx_2, f(x_2) = x_1$ . This is a bijection and we have:

$$\{w, vx_1\} \in E(G/vx_1) \Leftrightarrow \{w, v\} \in E(G) \vee \{w, x_1\} \in E(G) \quad (2.1)$$

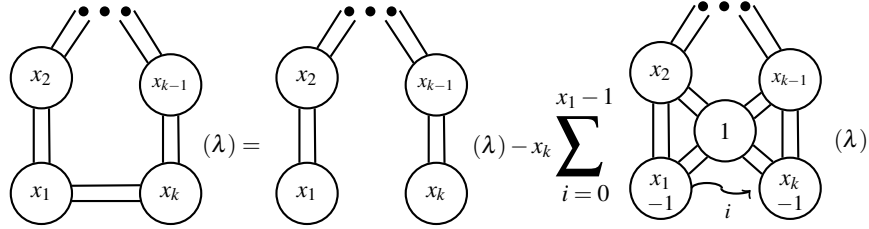
$$\Leftrightarrow \{w, v\} \in E(G) \vee \{w, x_2\} \in E(G) \quad (2.2)$$

$$\Leftrightarrow \{w, vx_2\} \in E((G - e_{v, x_1})/vx_2) \quad (2.3)$$

The second line (2) holds because  $N_G[x_1] = N_G[x_2]$ . The last line (3) holds because  $x_1$  is connected to  $x_2$  in  $G$ , so the removing of the line  $\{v, x_1\}$  does not affect the edges in  $(G - e_{v, x_1})/vx_2$ . For every other edge  $\{s, t\} \in E(G/vx_1)$  it is clear that it is also an edge in  $(G - e_{v, x_1})/vx_2$ . Thus  $f$  is an isomorphism. The proof of the other isomorphisms is analogously.  $\square$

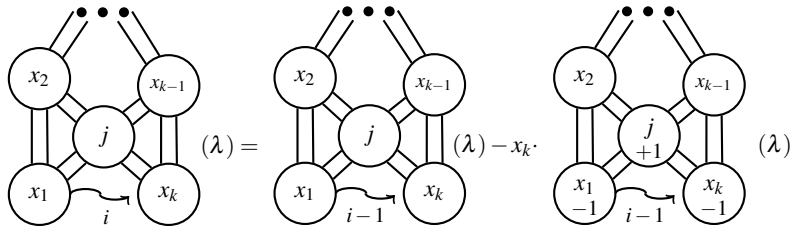
The next proposition is the first concrete step towards proving Theorem 2.13.

**Proposition 2.15.** Let  $H = [x_1, \dots, x_k]_H$  with  $x_i \in \mathbb{N}$  be a hyperhole of length  $k \geq 4$ . The following holds:



Before we prove Proposition 2.15, we will prove the following lemma.

**Lemma 2.16.** The following holds, where  $x_i = \#X_i$ :



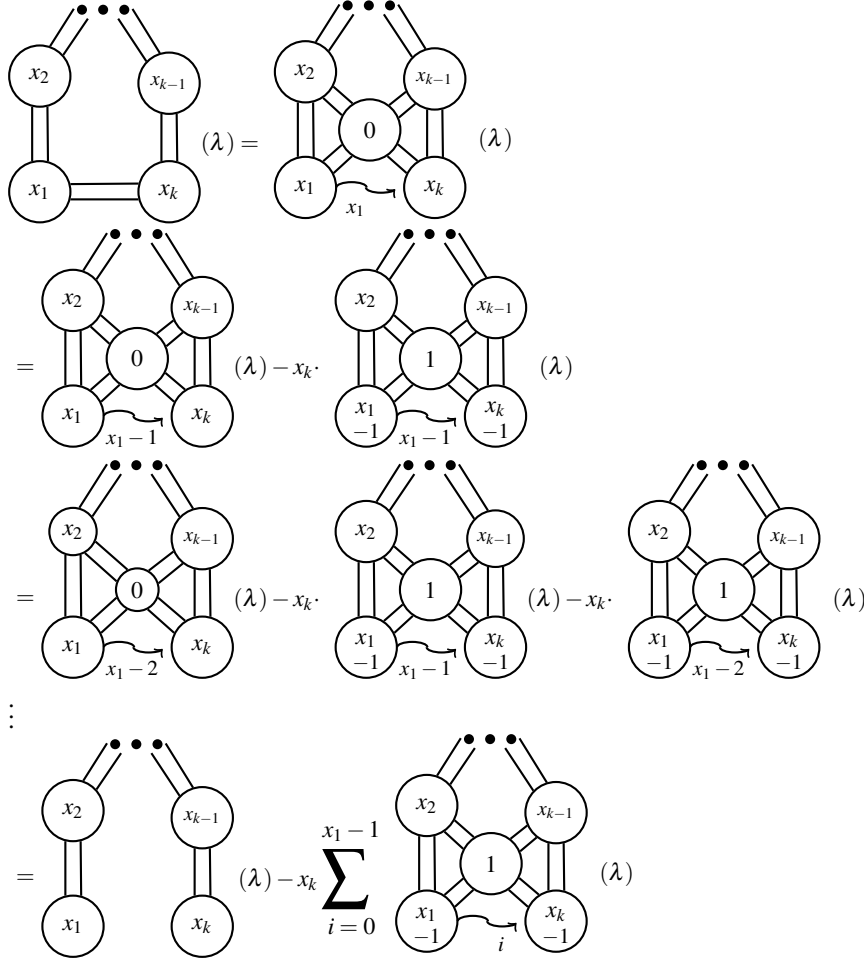
*Proof.* Consider the graph  $G$  visualised on the left side of the equation and a vertex  $v$  of  $X_1$  that is complete to  $X_k = \{w_1, \dots, w_t\}$ . We use Corollary 2.6 to remove the edges one by one between  $v$  and  $X_k$ , starting at  $w_1$ .

$$\begin{aligned} G(\lambda) &= (G - e_{vw_1})(\lambda) - (G/vw_1)(\lambda) \\ &= (G - e_{vw_1} - e_{vw_2})(\lambda) - \left( (G/vw_1)(\lambda) + ((G - e_{vw_1})/vw_2)(\lambda) \right) \\ &\vdots \\ &= (G - e_{vw_1} - \dots - e_{vw_t})(\lambda) - \left( (G/vw_1)(\lambda) + \dots + ((G - e_{vw_1} - \dots - e_{vw_{t-1}})/vw_t)(\lambda) \right) \\ &\stackrel{2.14}{=} (G - e_{vw_1} - \dots - e_{vw_t})(\lambda) - x_k \cdot (G/vw_1)(\lambda) \end{aligned}$$

We can use Lemma 2.14, because  $X_k$  is a clique and the neighbors of two vertices in  $X_k$  are the same. This is exactly the equation in the Lemma.  $\square$

We can now prove Proposition 2.15.

*Proof.* Take a hyperhole  $H = [x_1, \dots, x_k]_H = [X_1, \dots, X_k]$  with  $x_i = \#X_i \in \mathbb{N}$  of length  $k \geq 4$ . Notice that the leftmost graph  $G$  of Lemma 2.16 is the same as hyperhole  $H$  if we take  $i = x_1$  and  $j = 0$ . For the next part of the proof, we will use Lemma 2.16 for  $x_1$  number of times. Every time on the left-most graph:



$\square$

We already know the chromatic polynomial of any hyperpath. Therefore, the next step is to simplify the right side of the equation in Lemma 2.15. To do so, we use a property from Pascal's Triangle.

**Lemma 2.17.** For all  $n \in \mathbb{N}_{\geq 0}$  and for all  $0 \leq k \leq n$  this holds:

$$\binom{n+1}{k+1} = \sum_{i=k}^n \binom{i}{k}$$

*Proof.* We will prove this with induction to  $n$ .

BASE:

Assume  $n = 0$ , then  $k = 0$  as well. And  $\binom{1}{1} = \binom{0}{0} = \sum_{i=0}^0 \binom{i}{0}$ . Therefore the base case holds.

INDUCTION STEP:

Assume for some  $n$  we have that  $\binom{n+1}{k+1} = \sum_{i=k}^n \binom{i}{k}$  holds for every  $k \leq n$ . If  $k = 0$ , then  $\sum_{i=0}^{n+1} \binom{i}{0} = (n+1) \cdot 1 = \binom{n+1}{1}$ ; if  $k = n+1$ , then  $\binom{n+2}{n+2} = \sum_{i=n+2}^{n+2} \binom{i}{n+2}$ . Now take  $k > 0$  and  $k < n+1$ , then

$$\begin{aligned} \binom{n+2}{k+1} &= \binom{n+1}{k+1} + \binom{n+1}{k} \\ &\stackrel{IH}{=} \sum_{i=k}^n \binom{i}{k} + \sum_{i=k-1}^n \binom{i}{k-1} \\ &= \left( \sum_{i=k}^n \binom{i}{k} + \binom{i}{k-1} \right) + \binom{k-1}{k-1} \\ &= \left( \sum_{i=k}^n \binom{i+1}{k} \right) \binom{k-1}{k-1} \\ &= \left( \sum_{i=k+1}^{n+1} \binom{i}{k} \right) \binom{k}{k} \\ &= \sum_{i=k}^{n+1} \binom{i}{k} \end{aligned}$$

So the property stated in the lemma holds for every  $n \geq 0$  and every  $0 \leq k \leq n$ . □

**Proposition 2.18.** Let  $x_k \geq x_1$  and  $x_i = \#X_i$ , then the following holds:

$$-x_k \sum_{i=0}^{x_1-1} \text{Graph}_1(\lambda) = \left( \sum_{l=1}^{x_1} (-1)^l \left( \prod_{j=0}^{l-1} (x_k - j) \right) \binom{x_1}{l} \right) \text{Graph}_2(\lambda)$$

*Proof.* The following proof is a twofold. We will first prove another claim before we finish the proof itself.

**Claim 1.** The following holds where  $x_i = \#X_i$ :

$$\text{Graph}_1(\lambda) = \left( \sum_{l=0}^i (-1)^l \left( \prod_{j=0}^{l-1} (x_k - j) \right) \binom{i}{l} \right) \text{Graph}_2(\lambda)$$

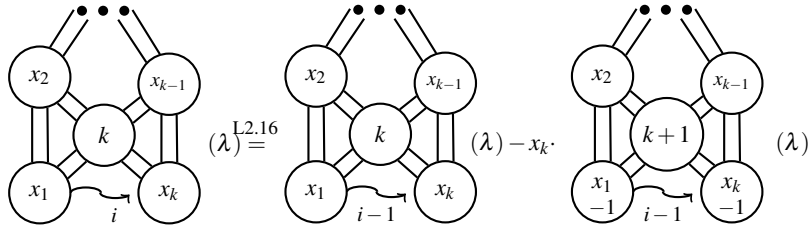
*Proof of Claim 1:* We will prove this with induction to  $i$ : the number of vertices in  $X_1$  that are complete to  $X_k$ . The proof has the same idea as the inductive proof for the Binomial Theorem [9].

BASE:

Assume  $i = 0$ , so no vertices are complete to  $X_k$ . This is trivial as the left part of the equation is the same as the right side.

INDUCTION STEP:

Assume the Claim 1 holds for all graphs that have exactly  $i-1$  vertices in  $X_1$  that are complete to  $X_k$ . Consider a graph that has exactly  $i$  vertices in  $X_1$  complete to  $X_k$ . The first step is to use Lemma 2.16:

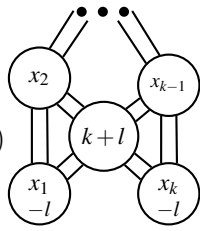


$$\begin{aligned} & \mathbb{H} \left( \sum_{l=0}^{i-1} (-1)^l \left( \prod_{j=0}^{l-1} (x_k - j) \right) \binom{i-1}{l} \right) \text{Diagram}(k+l, \lambda) \\ & - x_k \cdot \left( \sum_{l=0}^{i-1} (-1)^l \left( \prod_{j=0}^{l-1} (x_k - 1 - j) \right) \binom{i-1}{l} \right) \text{Diagram}(k+l+1, \lambda) \end{aligned}$$

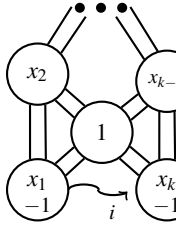
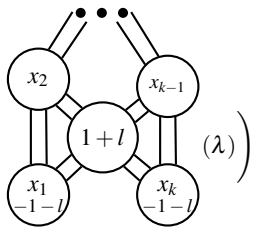
$$\begin{aligned} & = \left( \sum_{l=0}^{i-1} (-1)^l \left( \prod_{j=0}^{l-1} (x_k - j) \right) \binom{i-1}{l} \right) \text{Diagram}(k+l, \lambda) \\ & + \left( \sum_{l=1}^i (-1)^l \left( \prod_{j=0}^{l-1} (x_k - j) \right) \binom{i-1}{l-1} \right) \text{Diagram}(k+l, \lambda) \end{aligned}$$

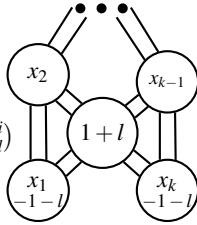
$$= \left( \sum_{l=1}^{i-1} (-1)^l \left( \prod_{j=0}^{l-1} (x_k - j) \right) \binom{i-1}{l} \right) \text{Diagram}(k+l, \lambda)$$

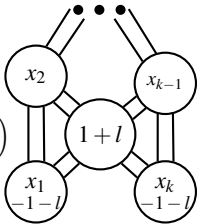
$$+ \text{Diagram}(k, \lambda) + (-1)^i \left( \prod_{j=0}^{i-1} (x_k - j) \right) \text{Diagram}(k+i, \lambda)$$

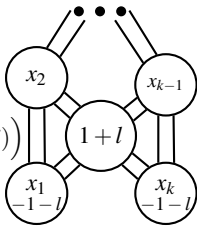
$$= \sum_{l=0}^i (-1)^l \left( \prod_{j=0}^{l-1} (x_k - j) \right) \binom{i}{l} (\lambda).$$


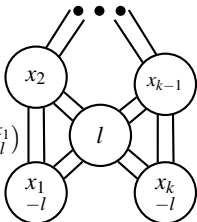
With the principle of induction, we have proven the claim. Now that we have proven the claim, we can use it for the first line of the following equation.

$$-x_k \sum_{i=0}^{x_1-1} \binom{x_1-1}{i} \binom{i}{1} (\lambda) \stackrel{c1}{=} -x_k \left( \sum_{i=0}^{x_1-1} \sum_{l=0}^i (-1)^l \left( \prod_{j=0}^{l-1} (x_k - 1 - j) \right) \binom{i}{l} \right) (\lambda)$$



$$= - \sum_{i=0}^{x_1-1} \sum_{l=0}^i (-1)^l \left( \prod_{j=0}^l (x_k - j) \right) \binom{i}{l} (\lambda)$$


$$= - \sum_{l=0}^{x_1-1} \left( (-1)^l \left( \prod_{j=0}^l (x_k - j) \right) \binom{l}{l} (\lambda) \cdot \sum_{i=l}^{x_1-1} \binom{i}{l} \right)$$


$$\stackrel{L2.17}{=} - \sum_{l=0}^{x_1-1} \left( (-1)^l \left( \prod_{j=0}^l (x_k - j) \right) \binom{l}{l} (\lambda) \cdot \binom{x_1-1}{l+1} \right)$$


$$= \sum_{l=1}^{x_1} (-1)^l \left( \prod_{j=0}^{l-1} (x_k - j) \right) \binom{x_1}{l} (\lambda)$$


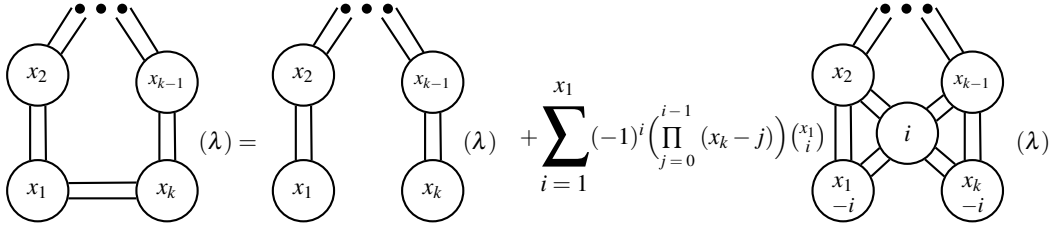
□

At the moment, we have created a lot of cliques, so we can use the clique-cut-partition theorem for chromatic polynomials (Theorem 2.8) to simplify the right side of the equation in Lemma 2.18 even more. We can now prove the Chromatic Theorem for hyperholes. For convenience the statement of the Chromatic Theorem for hyperholes is stated below.

**Theorem 2.13.** Let  $G = [x_1, \dots, x_k]_H$  be a hyperhole of length  $k \geq 4$ . Without loss of generality, assume  $x_k \geq x_1$ . Indices are not taken modulo  $k$  and we define  $x_0 := 0$ . Then the chromatic polynomial of  $G$  can be calculated recursively:

$$\begin{aligned}
[x_1, \dots, x_k]_H(\lambda) &= \left( \prod_{i=1}^k \prod_{j=0}^{x_i-1} (\lambda - x_{i-1} - j) \right) \\
&\quad + \left( \sum_{i=1}^{x_1} (-1)^i \binom{x_1}{i} \left( \prod_{j=0}^{i-1} (x_k - j) \right) \left( \prod_{j=x_2+i}^{x_1+x_2-1} (\lambda - j) \right) \right. \\
&\quad \quad \left. \cdot \left( \prod_{j=x_{k-1}+i}^{x_{k-1}+x_k-1} (\lambda - j) \right) \cdot [i, x_2, \dots, x_{k-1}]_H(\lambda) \right)
\end{aligned}$$

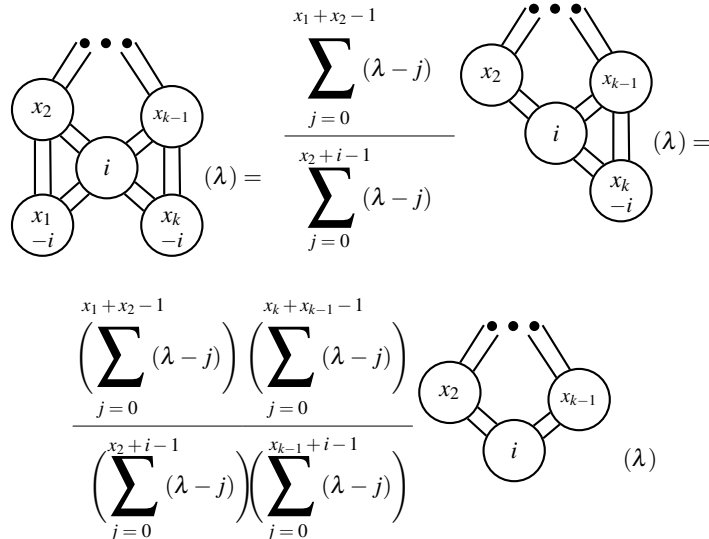
*Proof.* Take a hyperhole  $G = [X_1, \dots, X_k]_H = [x_1, \dots, x_k]_H$  of length  $k \geq 4$ . With help of Lemma 2.15 and Lemma 2.18, we know



Using Proposition 2.10 we calculate the following where  $x_0 := 0$ :

$$[x_1, \dots, x_k]_P(\lambda) := \prod_{i=1}^k \prod_{j=0}^{x_i-1} (\lambda - x_{i-1} - j).$$

Let  $G'$  be the graph on the right-most side of the next equation,  $Y$  be the  $i$  vertices in the middle of the graph and  $X'_1 := \{v_1^1, \dots, v_1^{x_1-i}\}$ . Then  $(X'_1, V(G) \setminus (X'_1 \cup X_2 \cup Y), X_2 \cup Y)$  is a clique-cut-partition. We apply the Clique-cut-partition Theorem for chromatic polynomials (Theorem 2.8) two times and insert the chromatic polynomial for a complete graph (Lemma 2.4):



With one more step, we have proven our statement. □

## 2.4 Magma program and computation time

When we find the representation  $X := [X_1, \dots, X_k]_H$  of a hyperhole, we can convert Theorem 2.13 into a Magma program.

**Algorithm 1.** CHROMATICPOLYNOMIALHYPERHOLE( $X$ ).

```

1  #INPUT: Representation of a hyperhole in integers: X := [|X1|, ..., |Xk|]
2  #OUTPUT: The chromatic polynomial of the hyperhole
3
4  P<x> := PolynomialRing(Integers());
5
6  ChromaticPolynomialHyperhole := function(X);
7      #Rotate the graph such that the smallest clique is X1
8      min, i := Min(X);
9      Rotate(~X, -i+1);
10
11     if |X| eq 3 then
12         #The hyperhole with vertex partition X is a complete graph
13         g := 1;
14         for i:=0 to X[1]+X[2]+X[3]-1 do
15             g*:= (x-i);
16         end for;
17         return g;
18     else
19         f:= 1;
20
21         #Chromatic Polynomial for hyperpath [X1, ..., Xk]P
22         for i:=1 to |X| do
23             if i eq 1 then
24                 n:=0;
25             else
26                 n:=X[i-1];
27             end if;
28             for j:=n to (n+X[i]-1) do
29                 f*:= (x-j);
30             end for;
31         end for;
32
33         for i:=1 to X[1] do
34             h := (-1)i * Binomial(X[1], i);
35             for j := 0 to (i-1) do
36                 h*:= (X[|X|] - j);
37             end for;
38
39             for j := X[2]+i to X[1]+X[2]-1 do
40                 h*:= (x-j);
41             end for;
42
43             for j := X[|X|-1]+i to X[|X|]+X[|X|-1]-1 do
44                 h*:= (x-j);
45             end for;
46
47             Y := X;
48             Exclude(~Y, Y[1]);

```



```

49         Y[|Y|] := i;
50         h1 := $$ (Y);           #Recursive call to the algorithm
51
52         f += h*h1;
53     end for;
54     return f;
55 end if;
56 end function;

```

We can calculate the chromatic polynomial of a hyperhole by using Corollary 2.6 recursively. Let  $G = [X_1, \dots, X_k]_H$  be a hyperhole of length  $k$  and  $e_{uv}$  an edge. We define  $x_i := X_i$ . Then

$$P_G(\lambda) = P_{G-e_{uv}}(\lambda) + P_{G/uv}(\lambda)$$

The standard algorithm in Magma uses this principle. In total  $G$  has  $\sum_{i=1}^k \frac{x_i^2 - x_i}{2}$  edges in all cliques  $X_1, \dots, X_k$  combined and  $\sum_{i=1}^k x_i \cdot x_{i+1}$  edges between the cliques. Notice that the algorithm in Magma uses a lot of memory as for every edge in  $G$  two graphs must be saved. As an example we take the graph  $G := [1, 2, 1, 3, 5, 2, 1]_H$ , drawn in Figure 2.6.

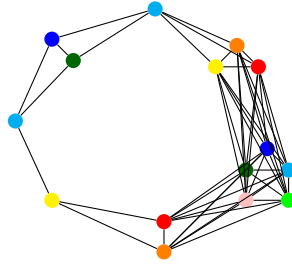


Figure 2.6: Graph  $[1, 2, 1, 3, 5, 2, 1]_H$  colored with  $\chi(H) = 8$  colors.

The chromatic polynomial for  $G$  is equal to

$$\begin{aligned}
& [1, 2, 1, 3, 5, 2, 1]_H(\lambda) = \\
& \lambda^{15} - 50\lambda^{14} + 1138\lambda^{13} - 15626\lambda^{12} + 144640\lambda^{11} - 955238\lambda^{10} + 4646270\lambda^9 \\
& - 16935214\lambda^8 + 46589063\lambda^7 - 96557200\lambda^6 + 148924952\lambda^5 - 166283680\lambda^4 \\
& + 127253616\lambda^3 - 59634432\lambda^2 + 12821760\lambda.
\end{aligned}$$

The standard Magma algorithm needed 328 seconds to compute the chromatic polynomial, while CHROMATICPOLYNOMIALHYPERHOLE needed less than a second. In fact, the first hyperhole we could find of length seven that needed one second for CHROMATICPOLYNOMIALHYPERHOLE to compute its chromatic polynomial, was  $[17, 16, 18, 17, 19, 16, 21]_H$ . This polynomial has degree 124 and the largest coefficient is approximately  $5,6 \cdot 10^{173}$ .

## 2.5 Chromatic polynomial for classes of hyperholes

As discussed before: a cycle is an example of a hyperhole. Let us calculate the chromatic polynomial of a cycle  $C_n$  with Theorem 2.13. For all  $i$  it is  $\#X_i = 1$ :

$$\begin{aligned}
C_n(\lambda) &= \prod_{i=1}^n \prod_{j=0}^{i-1} (\lambda - 1 \cdot \mathbf{1}_{\mathbb{N} \setminus \{1\}}(i-j)) + \\
& + \sum_{i=1}^1 (-1)^i \binom{1}{i} \left( \prod_{j=0}^{i-1} (1-j) \right) \left( \prod_{j=1+i}^1 (\lambda-j) \right) \left( \prod_{j=1+i}^1 (\lambda-j) \right) \cdot C_{n-1}(\lambda) \\
& = \lambda(\lambda-1)^{n-1} - C_{n-1}(\lambda)
\end{aligned}$$

This is the same we get if we use deleting of edges (Theorem 2.6) when we remember  $P_n(\lambda) = \lambda(\lambda - 1)^{n-1}$ . Using this, we can prove the chromatic polynomial of a cycle.

**Lemma 2.19.** The chromatic polynomial of a cycle  $C_n$  on  $n$  vertices is

$$P_{C_n}(\lambda) = (\lambda - 1)^n + (-1)^n(\lambda - 1).$$

*Proof.* We will prove it with induction to the number of vertices in the cycle.

BASE: Assume  $n = 3$ , then  $C_3 = K_3$ , thus

$$\begin{aligned} P_{C_3}(\lambda) &= \lambda(\lambda - 1)(\lambda - 2) = (\lambda - 1)(\lambda^2 - 2\lambda) \\ &= (\lambda - 1)((\lambda - 1)^2 - 1) = (\lambda - 1)^3 + (-1)^3(\lambda - 1) \end{aligned}$$

INDUCTION STEP: Assume for an  $n$  we know  $P_{C_n}(\lambda) = (\lambda - 1)^n + (-1)^n(\lambda - 1)$ . Using Theorem 2.13:

$$\begin{aligned} P_{C_{n+1}}(\lambda) &\stackrel{[2.13]}{=} \lambda(\lambda - 1)^n - C_n(\lambda) \\ &\stackrel{IH}{=} \lambda(\lambda - 1)^n - (\lambda - 1)^n - (-1)^n(\lambda - 1) \\ &= (\lambda - 1)^{n+1} + (\lambda - 1)^n - (\lambda - 1)^n + (-1)^{n+1}(\lambda - 1) \\ &= (\lambda - 1)^{n+1} + (-1)^{n+1}(\lambda - 1) \end{aligned}$$

□

When a hyperhole  $[x_1, \dots, x_k]_H$  with  $k \geq 4$  has  $x_i = 1$  for an  $i$ , we can calculate its chromatic polynomial. Without loss of generality, we say  $x_1 = 1$ . The next theorem calculates  $[1, x_2, \dots, x_k]_H$ .

**Theorem 2.20.** Let  $G = [1, x_2, \dots, x_k]_H$  be a hyperhole of length  $k \geq 4$ . Then the chromatic polynomial of  $G$  is equal to

$$G(\lambda) = \left( \prod_{i=2}^k \prod_{j=x_{i-1}+1}^{x_{i-1}+x_i-1} (\lambda - j) \right) \left( \left( \prod_{i=1}^k (\lambda - x_i) \right) + (-1)^k \cdot \left( \prod_{i=1}^k x_i \right) (\lambda - 1) \right)$$

*Proof.* We will prove this with induction to  $k$ : the length of a hyperhole.

BASE:

Consider a hyperhole  $[1, x_2, x_3, x_4]_H$ . We apply Theorem 2.13 and recall that  $[1, x_2, x_3]_H = K_{1+x_2+x_3}$ ; a complete graph. We define  $x_0 := 0$ .

$$\begin{aligned} &[1, x_2, x_3, x_4]_H(\lambda) \\ &\stackrel{[2.13]}{=} \left( \prod_{i=1}^4 \prod_{j=x_{i-1}+1}^{x_{i-1}+x_i-1} (\lambda - j) \right) - x_4 \left( \prod_{j=x_3+1}^{x_3+x_4-1} (\lambda - j) \right) \left( \prod_{j=0}^{x_2+x_3} (\lambda - j) \right) \\ &= \left( \prod_{i=2}^4 \prod_{j=x_{i-1}+1}^{x_{i-1}+x_i-1} (\lambda - j) \right) \left( \lambda \left( \prod_{i=1}^3 (\lambda - x_i) \right) - x_4 \lambda \cdot (\lambda - 1)(\lambda - (x_2 + x_3)) \right) \\ &= \left( \prod_{i=2}^4 \prod_{j=x_{i-1}+1}^{x_{i-1}+x_i-1} (\lambda - j) \right) \left( \left( \prod_{i=1}^4 (\lambda - x_i) \right) + x_4 x_3 x_2 (\lambda - 1) \right) \end{aligned}$$

Therefore, the base-case is true.

INDUCTION:

Assume for every  $x_2, \dots, x_k \in \mathbb{N}$  a hyperhole  $G' = [1, x_2, \dots, x_k]_H$  of length  $k$  the formula

$$G(\lambda) = \left( \prod_{i=2}^k \prod_{j=x_{i-1}+1}^{x_{i-1}+x_i-1} (\lambda - j) \right) \left( \left( \prod_{i=1}^k (\lambda - x_i) \right) + (-1)^k \cdot \left( \prod_{i=1}^k x_i \right) (\lambda - 1) \right)$$

holds. Consider a hyperhole  $G = [1, x_2, \dots, x_{k+1}]$  of length  $k + 1$ . Notate  $x_0 := 1$ . With the recursive formula for hyperholes, we know

$$\begin{aligned}
& [1, x_2, \dots, x_{k+1}]_H(\lambda) \\
\stackrel{[2.13]}{=} & \left( \prod_{i=1}^{k+1} \prod_{j=x_{i-1}}^{x_{i-1}+x_i-1} (\lambda - j) \right) - x_{k+1} \left( \prod_{j=x_k+1}^{x_k+x_{k+1}-1} (\lambda - j) \right) [1, x_2, \dots, x_k]_H(\lambda) \\
\stackrel{[IH]}{=} & \left( \prod_{i=1}^{k+1} \prod_{j=x_{i-1}}^{x_{i-1}+x_i-1} (\lambda - j) \right) - x_{k+1} \left( \prod_{j=x_k+1}^{x_k+x_{k+1}-1} (\lambda - j) \right) \cdot \left( \prod_{i=2}^k \prod_{j=x_{i-1}+1}^{x_{i-1}+x_i-1} (\lambda - j) \right) \\
& \cdot \left( \left( \prod_{i=1}^k (\lambda - x_i) \right) + (-1)^k \cdot \left( \prod_{i=1}^k x_i \right) (\lambda - 1) \right) \\
= & \left( \prod_{i=2}^{k+1} \prod_{j=x_{i-1}+1}^{x_{i-1}+x_i-1} (\lambda - j) \right) \\
& \cdot \left( \lambda \left( \prod_{i=1}^k (\lambda - x_i) \right) - x_{k+1} \left( \left( \prod_{i=1}^k (\lambda - x_i) \right) + (-1)^k \cdot \left( \prod_{i=1}^k x_i \right) (\lambda - 1) \right) \right) \\
= & \left( \prod_{i=2}^{k+1} \prod_{j=x_{i-1}+1}^{x_{i-1}+x_i-1} (\lambda - j) \right) \left( \left( \prod_{i=1}^{k+1} (\lambda - x_i) \right) + \left( (-1)^{k+1} \cdot \left( \prod_{i=1}^{k+1} x_i \right) (\lambda - 1) \right) \right)
\end{aligned}$$

This completes the proof. □

# Chapter 3

## Coloring rings

### 3.1 Recognizing rings

One way to introduce rings is through circular-arc graphs. Rings form a special case of those graphs. Recall that circular-arc graphs are intersection graphs of open intervals  $(a, b)$  on a circle with a circumference  $r$ . The arcs are from  $a$  to  $b$  in a clockwise direction and  $a, b \in [0, r)$ .

**Definition 3.1.** A *Ring*  $R$  is an intersection graph of a circular-arc model for which the following requirements hold:

1. The arcs can be partitioned into  $k$  nonempty sets  $X_1, \dots, X_k$  where  $k \geq 4$  is the circumference of the circular arc model such that (2.) is satisfied;
2. All arcs in set  $X_i$  are of the form  $(i - a_i^j, i + b_i^j)$  with the extra restrictions:

$$0 < a_i^j, b_i^j \leq 1 \tag{3.1}$$

$$a_i^1 \geq \dots \geq a_i^{\#X_i} \tag{3.2}$$

$$b_i^1 \geq \dots \geq b_i^{\#X_i} \tag{3.3}$$

$$a_i^1 = 1 \wedge b_i^1 = 1. \tag{3.4}$$

We say  $R$  is a  $k$ -ring and the length of  $R$  is equal to  $k$ .

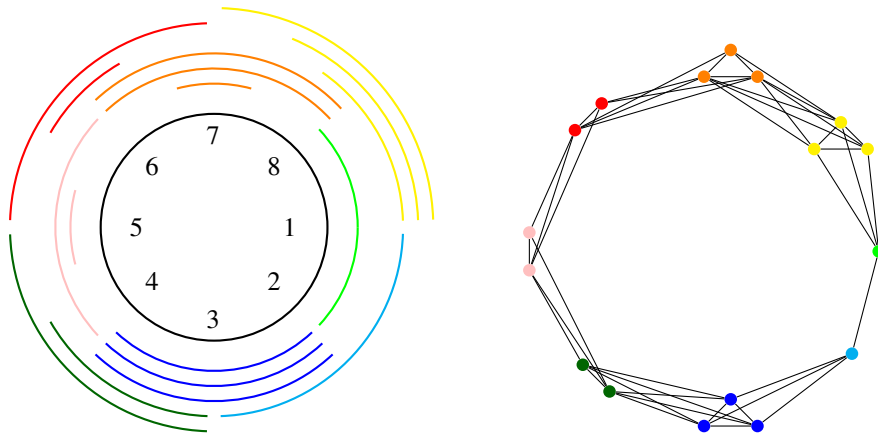


Figure 3.1: Ring of length 8. On the left the circular-arc model. On the right its intersection graph.

The model on the right of Figure 3.1 looks similar to a hyperhole. However, some edges from a hyperhole may be missing. That is another way to introduce rings: hyperholes are a subclass of rings. From the definition of hyperholes we can gain a characterization for a ring by making the requirement that every vertex  $v$  in  $X_i$  is complete to  $X_{i-1} \cup X_i \setminus v \cup X_{i+1}$  less strict.

**Lemma 3.2.** The graph  $R$  is a ring of length  $k \geq 4$  if and only if there exists a partition of the vertex set into  $k \geq 4$  nonempty sets  $X_1, X_2, \dots, X_k$ , such that for all  $1 \leq i \leq k$  the set  $X_i$  can be ordered as  $X_i = \{u_i^1, \dots, u_i^{\#X_i}\}$  and

$$X_i \subseteq N_R[u_i^{\#X_i}] \subseteq \dots \subseteq N_R[u_i^1] = X_{i-1} \cup X_i \cup X_{i+1}.$$

*Proof.* First we will prove the lemma from right to left. Assume graph  $R$  has a partition  $X_1, \dots, X_k$  for  $k \geq 4$  of the vertices  $V(R)$  where for every  $i$  there is an ordering  $X_i = \{u_i^1, \dots, u_i^{\#X_i}\}$  such that  $X_i \subseteq N_R[u_i^{\#X_i}] \subseteq \dots \subseteq N_R[u_i^1] = X_{i-1} \cup X_i \cup X_{i+1}$ . For every vertex  $u_i^j$ , we define the arc representing  $u_i^j$  to be  $A_i^j := (i - a_i^j, i + b_i^j)$  with

$$a_i^j := \frac{\#X_i - j + 1}{\#X_i}$$

$$b_i^j := \frac{l}{\#X_{i+1}} \text{ with } l \text{ the largest integer satisfying } \{v_{i+1}^1 \dots v_{i+1}^l\} \subseteq N_R(v_i^j).$$

Now we want to prove the two conditions of Definition 3.1 are met and that the model is a circular-arc model for  $R$ . The partition of the arcs will be analogue to the given vertex partition, i.e., we have an arc partition  $X_1^A, \dots, X_k^A$  with  $X_i^A = \{A_i^1, \dots, A_i^{\#X_i}\}$ . We will prove the second condition is also satisfied in Claim 1. Let  $X_i^A$  be a set of arcs.

**Claim 1.** The conditions (3.1)–(3.4) are satisfied for  $X_i^A$ .

*Proof of Claim 1:* By construction the conditions (3.1)–(3.3) and  $a_i^1 = 1$  are met. By assumption, we know  $N_R[u_i^1] = X_{i+1} \cup X_i \cup X_{i+1}$ . Thus  $l = \#X_{i+1}$  and the claim holds.

**Claim 2.** The set of open arcs  $X_1^A \cup \dots \cup X_k^A$  form a circular-arc model for  $R$ .

*Proof of Claim 2:* Let  $v, w$  be two neighbours in  $R$ . Assume  $v$  and  $w$  are from the same set  $X_i$ , i.e.,  $v = u_i^j$ ,  $w = u_i^h$  for an  $j, h$ , then  $i \in A_i^j \cap A_i^h$ . Therefore, their representing arcs intersect. Assume otherwise, so assume  $v$  and  $w$  are not from the same set  $X_i$  for any  $i$ . Without loss of generality,  $v = u_i^j$  and  $w = u_{i+1}^h$  for an  $i, j, h$ . We prove their representing arcs intersect, i.e.,  $i + b_i^j > i + 1 - a_{i+1}^h$  is satisfied:  $v$  and  $w$  are neighbors, therefore  $u_i^j \in N_R(u_{i+1}^h) \subseteq \dots \subseteq N_R(u_{i+1}^1)$  and thus

$$b_i^j \geq \frac{h}{\#X_{i+1}} > 1 - 1 + \frac{h-1}{\#X_{i+1}} = 1 - \frac{\#X_{i+1} + h - 1}{\#X_{i+1}} = 1 - a_{i+1}^h.$$

Therefore, their representing arcs intersect.

Let  $v, w$  be two vertices in  $R$  with no edge between the vertices. Assume  $v \in X_i$  and  $w \in V(R) \setminus (X_{i-1} \cup X_i) \cup X_{i+1}$ . Then it is trivial their representing arcs do not intersect. Assume without loss of generality  $v = u_i^j$  and  $w = u_{i+1}^h$  for an  $i, j, h$ . We prove their representing arcs do not intersect, i.e.,  $i + b_i^j \leq i + 1 - a_{i+1}^h$  is satisfied:  $v$  and  $w$  are no neighbors, therefore  $u_{i+1}^h \notin N_R(u_i^j)$  and thus

$$b_i^j \leq \frac{h-1}{\#X_{i+1}} = 1 - 1 + \frac{h-1}{\#X_{i+1}} = 1 - \frac{\#X_{i+1} + h - 1}{\#X_{i+1}} = 1 - a_{i+1}^h.$$

The intervals are open, so their representing arcs do not intersect.

Using Claim 1 and Claim 2 we know there exists a circular-arc model of every graph  $R$  that meets the requirements of Definition 3.1. Thus  $R$  is a ring.

We will also prove from left to right. Assume we have a ring  $R$ . This is an intersection graph of a circular-arc model with circumference  $k \geq 4$  where the arcs are partitioned into  $k$  nonempty sets  $Y_1, \dots, Y_k$  and all arcs in  $Y_i$  are of the form  $(i - a_i^j, i + b_i^j)$  that satisfy some additional restrictions (3.1)–(3.4). Let  $X_i$  be the set of vertices that represent  $Y_i$ . The first requirement ensures arcs in  $Y_i$  only intersect with arcs in  $Y_{i-1} \cup Y_i \cup Y_{i+1}$  and that all arcs in  $Y_i$  intersect each other. Therefore,  $X_i$  is a clique and anticomplete to all  $X_j$  with  $j \notin \{i-1, i, i+1\}$ . The second and third requirement guarantee the existence of an order in  $X_i := \{u_i^1, \dots, u_i^{\#X_i}\}$  such that  $N_R[u_i^{\#X_i}] \subseteq \dots \subseteq N_R[u_i^1]$ . The last requirement tells us  $N_R[u_i^1] = X_{i-1} \cup X_i \cup X_{i+1}$ .  $\square$

In literature, the characterization of Lemma 3.2 is mostly used for defining a ring [2]. From now on, when we discuss a ring, we often assume a ring partition  $X_1, \dots, X_k$  of vertices is given. We will consider the indices to be taken modulo  $k$  and we obtain an ordering of the set  $X_i = \{u_i^1, \dots, u_i^{\#X_i}\}$  such that

$$X_i \subseteq N_R[u_i^{\#X_i}] \subseteq \dots \subseteq N_R[u_i^1] = X_{i-1} \cup X_i \cup X_{i+1}.$$

We define  $s_i := u_i^1$  and  $t_i := u_i^{\#X_i}$ .

**Corollary 3.3.** All hyperholes are rings.

*Proof.* Let  $G = [X_1, \dots, X_k, X_1]_H$  be a hyperhole of length  $k \geq 4$ . All vertices  $v$  in  $X_i$  are complete to  $X_{i-1} \cup (X_i \setminus \{v\}) \cup X_{i+1}$  and anticomplete to all other vertices, so

$$X_i \subseteq N_G[u_i^{\#X_i}] = \dots = N_G[u_i^1] = X_{i-1} \cup X_i \cup X_{i+1}.$$

Therefore,  $G$  is a hyperhole. □

Using Lemma 3.2, we can program a recognition algorithm for rings that is described in Lemma 8.14 in [2]. This implementation is slightly different from the one given in [2]. The difference are the extra checks at the end of the algorithm. When we want to check whether we can sort the vertices in a clique  $X_i = \{u_i^1, \dots, u_i^{\#X_i}\}$  such that  $N_R[u_i^{\#X_i}] \subseteq \dots \subseteq N_R[u_i^1]$  hold, we use Algorithm 16 SORT-CLIQUE( $G, X_i$ ) to sort  $X_i$  from vertices with the most neighbors to vertices with the least neighbors, as described in the Appendix. It then checks if  $N_R[u_i^{\#X_i}] \subseteq \dots \subseteq N_R[u_i^1]$ . The running time of this algorithm is the same as the running time of the algorithm that determines if a graph is chordal.

The algorithm ISRING( $G$ ) starts with a vertex  $v$  with the largest number of neighbors, this will become vertex  $s_1$ . The first clique  $X_1$  is defined as all the vertices with its neighborhood contained in  $N_G[s_1]$  (lines 7–13). For the second clique  $X_2$ , we start with a random vertex  $x \in N_G(s_1) \setminus X_1$  (lines 22–27). We set  $X_2 := (N_G[x] \cap N_G(s_1)) \setminus X_1$  (lines 28–29). The other cliques are made recursively:  $X_i := N_G(s_{i-1}) \setminus (X_1 \cup \dots \cup X_{i-1})$ . At the end of the algorithm, there are some checks (lines 56–74) for example that  $G[\{s_1, \dots, s_k\}]$  is a hole or that  $s_i$  is complete to  $X_{i+1}$ . After we give the algorithm itself, we will prove the checks are sufficient to determine whether  $G$  is a ring or not.

**Algorithm 2.** ISRING( $G$ ).

```

1  #INPUT: Graph G
2  #OUTPUT: True together with its good partition if G is a ring. False otherwise.
3  #RUNNING TIME:  $\mathcal{O}(n^2)$ 
4  IsRing := function(G);
5      #Make X1
6      d,v := Maxdeg(G);
7      X1 := [v];
8      Neighbor := Include(Neighbors(v),v);
9      for w in Neighbors(v) do
10         if Neighbors(w) subset Neighbor then
11             Include(~X1, w);
12         end if;
13     end for;
14     b, X1 := SortClique(G,X1);
15     if not b then
16         "X1 could not be ordered properly";
17         return false, [];
18     end if;
19
20     #Make X2: First find an x in X2
21     x:=0;
22     for w in Neighbor do
23         if not w in X1 then
24             x:=w;
25             break;

```

```

26         end if;
27     end for;
28     X2:=[y : y in Neighbors(x) | y in Neighbors(X[1][1]) and not y in X1 ];
29     Append(~X2,x);
30     b, X2 := SortClique(G,X2);
31     if not b then
32         "X2 could not be orderd properly";
33         return false, [];
34     end if;
35
36     cupX:=X1 cat X2;
37     X:= [X1,X2];
38
39     #Make all other Xi
40     while true do
41         Xi := [x : x in Neighbors(X[|X|][1]) | not x in cupX];
42         if |Xi| eq 0 then
43             break;
44         end if;
45         b,Xi := SortClique(G,Xi);
46         if not b then
47             "Xi could not be orderd properly";
48             return false, [];
49         end if;
50
51         #Update X and cupX
52         Append(~X, Xi);
53         cupX := cupX cat Xi;
54     end while;
55
56     #Last Checks:
57     RemainingVertices := [y : y in Vertices(G) | not y in cupX];
58     if |RemainingVertices| ne 0 or |X| le 3 then
59         "k<=3 or V(X) ne V(G)";
60         return false, [];
61     elif not IsPolygon(sub<G|{X[i][1] : i in [1..|X|]}>) then
62         "x1,x2,...,xk,x1 is not a hole";
63         return false, [];
64     end if;
65     for i:= 1 to |X|-1 do
66         if not X[i] subset Neighbors(X[i+1][1]) then
67             return false
68         end if;
69     end for;
70     if not X[1] subset Neighbors(X[|X|][1]) then
71         return false;
72     elif not X[|X|] subset Neighbors(X[1][1]) then
73         return false;
74     end if;
75
76     return true, X;
77 end function;

```

*Proof.* To check if this algorithm is correct, we use the characterization of a ring given in Lemma 3.2. We immediately can see that  $X = [X_1, \dots, X_k]$  is a partition of the vertices in  $G$  into  $k \geq 4$  nonempty sets from the way  $X$  is defined and from the final checks. With SORTCLIQUE, we know that for every  $i$ , we have an ordered  $X_i := [x_i^1, \dots, x_i^{\#X_i}]$  such that  $N_R[u_i^{\#X_i}] \subseteq \dots \subseteq N_R[u_i^1]$ . We only need to check

that

$$X_i \subseteq N_R[u_i^{\#X_i}] \quad (3.5)$$

and

$$N_R[u_i^1] = X_{i-1} \cup X_i \cup X_{i+1} \quad (3.6)$$

is true for every  $i \leq k$ .

The first one, (3.5), is true because in SORTCLIQUE we check if  $N_R[u_i^{\#X_i}] \subseteq \dots \subseteq N_R[u_i^1]$ . In particular, we have  $N_R[u_i^{\#X_i}] \subseteq N_R[u_i^j]$  for every  $u_i^j \in X_i$  and thus  $u_i^{\#X_i} \in N_R[u_i^j]$ . For the second one, (3.6), we need to verify that two things are satisfied:

- ( $\subseteq$ ) Assume  $N_R[u_i^1] \not\subseteq X_{i-1} \cup X_i \cup X_{i+1}$ . Then there exists vertices  $u_i^s, u_j^t$  such that these vertices are connected and  $j \notin \{i-1, i, i+1\}$ . Then the vertices  $u_i^1, u_j^1$  are also connected, for otherwise one or both checks  $N_R[u_i^1] \subseteq N_R[u_i^1]$  and  $N_R[u_j^1] \subseteq N_R[u_j^1]$  fail. But then  $G[\{u_1^1, \dots, u_k^1\}]$  is not a hole. So the algorithm verifies whether or not  $N_R[u_i^1] \subseteq X_{i-1} \cup X_i \cup X_{i+1}$ .
- ( $\supseteq$ ) All vertices in  $X_i$  are in  $N_R[u_i^1]$ , because  $u_i^1 \in X_i$  and  $X_i$  is a clique. For every  $i < \#X$  we have  $X_{i+1} \subseteq N_G(u_i^1)$  by the way  $X_{i+1}$  is defined through  $u_i^1$ . The rest is checked in the last checks. □

**Lemma 3.4.** [14] Let  $R$  be a ring of length  $k \geq 4$ ,  $S$  a subset of its vertices and  $[v_1, \dots, v_t]$  a maximal sequence of pairwise distinct vertices of  $R \setminus S$  such that  $v_i$  is simplicial in  $R \setminus (S \cup \{v_1, \dots, v_{i-1}\})$ . Then either  $V(R) = S \cup \{v_1, \dots, v_t\}$  or  $R \setminus (S \cup \{v_1, \dots, v_t\})$  is a  $k$ -ring.

*Proof.* This proof is obtained from [14]. For a ring  $R$  with ring partition  $X_1, \dots, X_k$ ,  $S$  a subset of its vertices and  $[v_1, \dots, v_t]$  a maximal simplicial elimination ordering in  $R \setminus S$ , we set  $R' := R \setminus (S \cup \{v_1, \dots, v_t\})$  and  $Y_i := X_i \cap R'$ . Assume  $V(R') \neq \emptyset$  for otherwise the lemma is trivial. If for an  $i$  we have  $Y_i = \emptyset$ , then  $R'$  is chordal and there exist a simplicial vertex which is in contradiction with the maximality of the sequence  $[v_1, \dots, v_t]$ . We write  $Y_i = \{y_i^1, \dots, y_i^{\#Y_i}\}$  and by definition of a ring  $R$  we have

$$Y_i \subseteq N_{R'}[y_i^{\#Y_i}] \subseteq \dots \subseteq N_{R'}[y_i^1] \subseteq Y_{i-1} \cup Y_i \cup Y_{i+1}.$$

If  $N_{R'}[y_i^1] \not\subseteq Y_{i-1}$ , then  $y_{i-1}^{\#Y_{i-1}}$  is a simplicial vertex of  $R'$  contradicting the maximality of  $[v_1, \dots, v_t]$ . Analogously  $N_{R'}[y_i^1] \not\subseteq Y_{i+1}$  contradicts the maximality. Therefore  $R'$  is a ring with ring partition  $Y_1, \dots, Y_k$ . □

## 3.2 Coloring even rings

The chromatic number of a graph is always greater than or equal to the clique number of the graph, because all vertices in a clique have to be colored differently in order to obtain a proper coloring. For even rings it is an equality. A maximal clique  $C$  is always a subset  $X_i \cup X_{i+1}$  for some  $i$ , and if  $k$  is even, we can alternate the coloring tactics between the sets of the clique partition  $X_1, \dots, X_k$ . We use Algorithm CliqueSizeRing CLIQUESIZERING( $R, X$ ) to find  $\omega(R)$  in  $\mathcal{O}(n^3)$  time. We follow Lemma 3.2 of [14] to color an even ring  $R$  with the colors  $\{1, \dots, \omega(R)\}$ :

$$c(u_i^j) := \begin{cases} j & \text{if } i \text{ is odd;} \\ \omega(R) - j + 1 & \text{otherwise.} \end{cases}$$

**Algorithm 3.** COLOREVENRING( $R, X$ )



```

1  #INPUT: An even ring R together with its ring partition X.
2  #OUTPUT: A proper coloring that uses  $\chi(R)$  colors.
3  #RUNNING TIME:  $\mathcal{O}(n^3)$ 
4  ColorEvenRing := function(R,X);
5      w := CliqueSizeRing(R,X);
6      Parent := Parent({X[1][1]});
7      Colours := [Parent | {} : i in [1..w]];
8      for i:=1 to |X| do
9          if IsEven(i) then
10             for j:=1 to |X[i]| do
11                 Include(~Colours[j],X[i][j]);
12             end for;
13         else
14             for j:=1 to |X[i]| do
15                 Include(~Colours[w-j+1],X[i][j]);
16             end for;
17         end if;
18     end for;
19     return Colours;
20 end function;

```

The proof is straightforward [14]. The algorithm will color the vertices  $u_i^j$  and  $u_i^l$  differently. If  $u_i^j$  and  $u_{i+1}^l$  are neighbors, then  $R[\{u_i^1, \dots, u_i^j\} \cup \{u_{i+1}^1, \dots, u_{i+1}^l\}]$  is a clique of size  $l + j \leq \omega(R) = w$  by construction of a ring. Therefore it is not possible that  $u_i^j$  and  $u_{i+1}^l$  receive the same color. By construction of a ring, these are all possible connected vertices.

Using this algorithm, we can color the 8-ring of Figure 3.1. The maximal clique-size is 5, so we need five colors: {blue, dark blue, red, yellow, green} represent {1,2,3,4,5}.

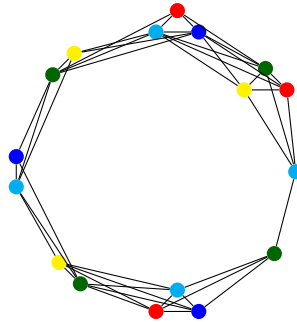


Figure 3.2: Following Algorithm 3 to color the ring of Figure 3.1 optimally

### 3.3 Coloring odd hyperholes

Only two colors are needed to color even cycles. To color an odd one, we need an extra color since we can not simply alternate the colors any more. The same holds for odd rings and odd hyperholes. A coloring algorithm for hyperholes is straightforward, given and proven as Theorem 3.2 of [16]. First, we want to determine its chromatic number.

**Theorem 3.5.** (Theorem 3.2 of [16]) Let  $G := [x_1, \dots, x_k]_H$  be a hyperhole of odd length  $k \geq 5$ , then

$$\chi(G) = \max \left( \omega(G), \left\lceil \frac{V(G)}{\alpha(G)} \right\rceil \right)$$

The algorithms, given as part of the proof [16], to color an odd hyperhole are below. The clique-size of a hyperhole is denoted as wMAX in Algorithm 4 and wTOT denotes the number of vertices. The idea is to determine how many colors are needed and then to color vertices contiguous with the

colors  $\{1, \dots, \chi(R)\}$  until we can color every vertex  $v$  of  $X_i$  for an even  $i$  with the colors  $\{1, \dots, \#X_i\}$ . At that point, we color the graph like it is an even ring. Notice that  $c(X[1]) = \{1, \dots, x_1\}$ .

**Algorithm 4.** CHROMHYPERHOLE( $G$ ):

```

1  #INPUT: An enumerated sequence that represents a hyperhole.
2  #OUTPUT: The chromatic number.
3  #RUNNING TIME:  $\mathcal{O}(n)$ 
4  ChromHyperhole := function(X);
5      wTOT := 0;
6      wMAX := 0;
7      for i:= 1 to |X| do
8          wTOT += X[i];
9          if X[i] + X[i mod |X| +1] gt wMAX then
10             wMAX := X[i] + X[i mod |X| +1];
11         end if;
12     end for;
13     if IsEven(|X|) then
14         return wMAX;
15     else
16         m := |X| div 2;
17         return Max(wMAX, Ceiling(wTOT/m));
18     end if;
19 end function;

```

**Algorithm 5.** COLORODDHYPERHOLE( $H$ ):

```

1  #INPUT: A hyperhole  $H$  of odd length represented by vertices and edges
2  #OUTPUT: A coloring of the graph, using at most  $\chi(H)$  colors.
3  #RUNNING TIME:  $\mathcal{O}(n+m)$ 
4  ColorOddHyperhole := function(G);
5      b,X:= IsRing(G);
6      x := [|X[i]| : i in [1..|X|]];
7      colors := [Parent({X[1][1]}) | {} : i in [1..ChromHyperhole(x)]];
8
9      sum := x[1];
10     k:=1;
11     c:=1;
12
13     #if  $\sum_{i=1}^k \leq (k \text{ div } 2) * |\text{colors}|$  for an odd  $k \geq 3$ , then we can color a
14     #vertex in  $X_{k+1}$  with the same color as  $X[1][1]$ . Untill then,
15     #we color contiguous
16     while sum gt (k div 2)*|colors| do
17         for i:=k to k+1 do
18             for j:=1 to x[i] do
19                 l := (c-1) mod |colors| + 1;
20                 Include(~colors[l],X[i][j]);
21                 c+=1;
22             end for;
23         end for;
24
25         sum += x[k]+x[k+1];
26         k+=2;
27     end while;
28
29     #Color the vertices with the same tactics on how to color even rings.
30     for i:=k to |X| do
31         if IsEven(i) then
32             for j:=1 to x[i] do

```

```

33         Include(~colors[j],X[i][j]);
34     end for;
35     else
36         for j:=1 to x[i] do
37             Include(~colors[|colors-j+1|], X[i][j]);
38         end for;
39     end if;
40 end for;
41 return colors;
42 end function;

```

In Figure 3.3, an example is given. We start coloring at the top clique of size 3 (with vertex  $u_1^1$ ). The colors are  $c = \{\text{dark blue, light blue, green, dark green, yellow, orange, purple}\}$ . At the bottom clique of size 3 ( $X_6$ ), we can finally color a vertex in an even set with dark blue (the same color as  $u_1^1$ ).

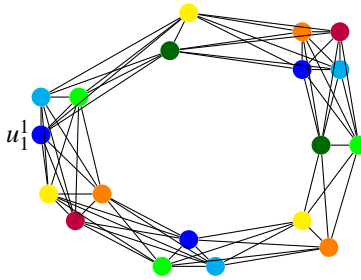


Figure 3.3: Coloring of the hyperhole  $[3, 2, 4, 2, 2, 3, 3]_H$

### 3.4 Coloring odd rings

In 2020, F. Maffray, I. Penev and K. Vušković [14] discovered a polynomial time algorithm to color odd rings. They first introduced the concept of unimprovable coloring for a subgraph  $R \setminus \{t_2\}$  of a ring  $R$ . Recall  $t_2$  is defined as the vertex  $u_2^{\#X_2}$ , a vertex with least number of neighbours in  $X_2$ . If  $R \setminus \{t_2\}$  is colored properly with coloring  $c$ , we can make the coloring unimprovable and color the whole ring using no more than  $\max\{\chi(R), r\}$  colors where  $r$  is the number of colors used to color  $R \setminus \{t_2\}$ .

**Definition 3.6.** For an odd ring  $R$ , a proper coloring  $c$  of  $R \setminus \{t_2\}$  is called *unimprovable* if for all colors  $a \in c(V(R) \setminus \{t_2\})$  such that  $a \neq c(s_1)$ , and for all components  $Q$  of  $R[\{v \in V(R \setminus \{t_2\}) : c(v) = a \vee c(v) = c(s_1)\}]$  that do not contain  $s_1$ , both the following are satisfied:

- For all odd  $i \in \{3, \dots, k\}$  such that  $Q \cap X_i \neq \emptyset$ , then  $a \notin c(X_i)$  or there exists indices  $j < l$  such that  $c(u_i^j) = c(s_1)$  and  $c(u_i^l) = a$ ;
- For all even  $i \in \{4, \dots, k-1\}$  such that  $Q \cap X_i \neq \emptyset$ , then  $c(s_1) \notin c(X_i)$  or there exists indices  $j > l$  such that  $c(u_i^j) = a$  and  $c(u_i^l) = c(s_1)$ .

For an odd ring  $R$  we define a proper coloring  $c$  on  $R \setminus \{t_2\}$  to be *improvable*, if  $c$  is not unimprovable.

The following is a property of unimprovable colorings:

**Lemma 3.7.** Let  $R$  be an odd ring and  $c$  an unimprovable coloring for  $R \setminus \{t_2\}$ , then for all vertices  $v \in V(R) \setminus \{t_2\}$  that are not colored with color  $c(s_1)$  and are not connected to  $s_1$  by a path containing only vertices that are colored with the color  $c(s_1)$  or  $c(v)$ , we have:

$$\begin{aligned}
 &v = u_i^l \text{ for some odd } i \text{ and } (\exists j < l [c(u_i^j) = c(s_1)]) \quad \text{or} \\
 &v = u_i^l \text{ for some even } i \text{ and } (\exists j > l [c(u_i^j) = c(s_1)] \vee \forall u \in X_i [c(u) \neq c(s_1)])
 \end{aligned}$$

*Proof.* Let  $R$  be a ring of odd length  $k \geq 5$ ,  $c$  an unimprovable coloring for  $R \setminus \{t_2\}$  and  $v \in V(R) \setminus \{t_2\}$  colored with color  $a \neq c(s_1)$ . If  $v$  is not connected to  $s_1$  by a path containing only vertices that are colored with the color  $c(s_1)$  or  $c(v)$ , then  $v$  is in a component  $Q$  of  $R[\{v \in V(R \setminus \{t_2\}) : c(v) = a \vee c(v) = c(s_1)\}]$  such that  $s_1 \notin Q$ . Let  $v = u_i^l$  for an  $i, l$ .

If  $i$  is odd, then by assumption of  $c$  being unimprovable, there exists an index  $j < l$  such that  $c(u_i^j) = c(s_1)$ . Therefore,  $v$  satisfies the condition in the Lemma 3.7.

If  $i$  is even, then by assumption of  $c$  being unimprovable, either  $c(s_1) \notin c(X_i)$ , so  $\forall u \in X_i [c(u) \neq c(s_1)]$ , or there exists an index  $j > l$  such that  $c(u_i^j) = c(s_1)$ . Therefore,  $v$  satisfies the condition in Lemma 3.7.

The property described in the lemma holds. □

An example of an unimprovable coloring of an odd ring is given in Figure 3.4. To check if the conditions hold for the color pink, we only need to check the vertices of the tree  $R[16, 17, 19]$  and this holds.

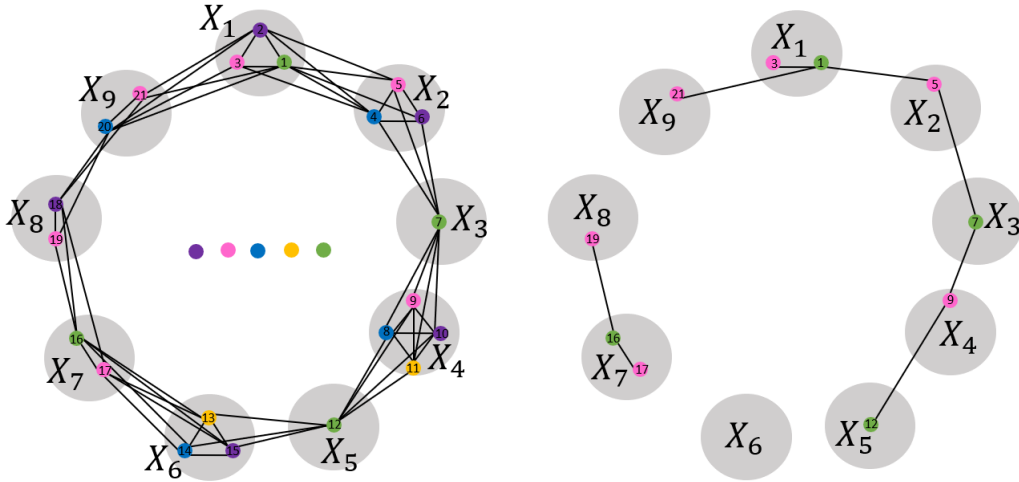


Figure 3.4: Ring of length 9 with unimprovable coloring on the left. On the right the forest  $F_{R \setminus t_2}^{\text{pink}}$ .

The following lemma is inspired by Claim 3 from Lemma 3.5 [14].

**Lemma 3.8.** Let  $R$  be an odd ring and  $c$  an unimprovable coloring of  $R \setminus \{t_2\}$ , then the color  $c(s_1)$  is assigned to  $\alpha(R) = \frac{k-1}{2}$  number of vertices.

*Proof.* Consider an odd length ring  $R$  with its ring partition  $X_1, \dots, X_k$ , an unimprovable coloring  $c$  of  $R \setminus \{t_2\}$  and let  $l$  be the first odd integer such that  $c(s_l) \neq c(s_1)$ . Such  $l$  exists, because  $k$  is odd and  $s_k$  is connected to  $s_1$ . The coloring  $c$  is unimprovable, so Lemma 3.7 suggests there should be a path from  $s_l$  to  $s_1$  from vertices containing only colors  $c(s_1)$  or  $c(s_l)$ . Notice that no vertex in  $X_{l-1}$  is colored with the color  $c(s_1)$ , because  $s_{l-2}$  is complete to  $X_{l-1}$  and the color  $c(s_1)$  is assigned to  $s_{l-2}$  by the choice of  $l$ . Thus there exists a path  $(s_l, p_{l+1}, \dots, p_k, s_1)$  with  $p_i$  in  $X_i$  and for even  $i$  we must have  $c(p_i) = c(s_1)$ . In total there are  $\frac{k-1}{2}$  vertices colored with the same color as  $s_1$ . □

An unimprovable coloring need not be optimal and an optimal coloring is not always unimprovable. In Figure 3.5 we see counterexamples of rings that are colored optimal or unimprovable, but not both. Unimprovable colorings are colorings such that the color assigned to  $s_1$  is optimally distributed throughout the ring. Sometimes it depends on the ring partition  $X_1 \dots, X_k$  (or even the order of the vertices in  $X_i$ ) whether or not a coloring is unimprovable. The right graph in Figure 3.5 is colored unimprovable for  $R \setminus \{t_2\}$  if  $X_1 = \{s_3\}$  and  $X_k = \{s_1\}$ , but improvable if  $X_1 = \{s_1\}$  and  $X_k = \{s_k\}$ .

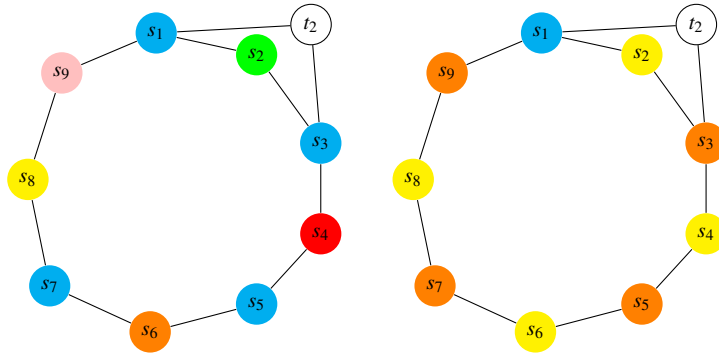


Figure 3.5: Let  $R$  be a ring with  $R \setminus \{t_2\} \cong C_9$ . On the left: an unimprovable coloring. On the right: an optimal coloring of  $R \setminus \{t_2\}$

To check whether a coloring  $c$  is unimprovable for  $R \setminus \{t_2\}$  where  $R$  is an odd ring, we will check for every color  $a \neq c(s_1)$  if the vertices that are colored with color  $a$  satisfy the conditions in Definition 3.1. To do so, for every color  $a \neq c(s_1)$  we make a forest

$$F_{R \setminus t_2}^a := R[\{v \in V(R \setminus t_2) : c(v) = a \vee c(v) = c(s_1)\}]$$

and check the vertices of every tree on this forest.

The following algorithm checks if a given coloring is unimprovable [14]. We use Algorithm 14 INDEXSET(Set,  $v$ ) of the Appendix. It returns the index  $i$  such that  $v$  is in Set[ $i$ ].

**Algorithm 6.** ISUNIMPROVABLE( $R, X, c$ )

```

1  #INPUT: An odd ring R together with its ring partition X and a coloring c
2  #      on V(R) \ {t2}
3  #OUTPUT: false, colors a and c(s1) and subtree of F_R^a where it is not unimprovable.
4  #RUNNING TIME: O(n^3)
5  IsUnimprovable := function(R,X,c);
6      c1 := c[IndexSet(c,X[1][1])];
7      X2 := [{v : v in X[i]} : i in [1..|X|]];
8      for a in Exclude(c,c1) do
9          #Make the vertices of the forrest F.
10         Fvertices:= a join c1;
11
12         while |Fvertices| gt 0 do
13             #Make a subtree of forrest F beginning at t
14             t := Random(Fvertices);
15             Tcheck := [t];
16             Exclude(~Fvertices, t);
17             T := {};
18             while |Tcheck| ge 1 do
19                 v := Tcheck[1];
20                 Exclude(~Tcheck, v);
21                 Include(~T, v);
22
23                 for w in Fvertices meet Neighbors(v) do
24                     Exclude(~Fvertices, w);
25                     Include(~Tcheck, w);
26                 end for;
27             end while;
28
29             #Check subtree T
30             if not {X[1][1]} subset T then
31                 for i:=1 to |X| do

```

```

32     Y:= [v : v in X2[i] meet T];
33     if |Y| eq 1 then
34         if IsEven(i) and {Y[i][1]} subset c1 then
35             return false, a, c1, T;
36         elif not IsEven(i) and {Y[i][1]} subset a then
37             return false, a,c1, T;
38         end if;
39     elif |Y| eq 2 then
40         m := Index(X[i],Y[i][1]);
41         n := Index(X[i],Y[i][2]);
42         if m gt n and {Y[i][1]} subset a and IsEven(i) then
43             return false, a, c1, T;
44         elif n gt m and {Y[i][2]} subset a and IsEven(i) then
45             return false, a, c1, T;
46         elif m gt n and {Y[i][1]} subset c1 and not IsEven(i) then
47             return false, a, c1, T;
48         elif n gt m and {Y[i][2]} subset c1 and not IsEven(i) then
49             return false, a, c1, T;
50         end if;
51     end if;
52     end for;
53     end if;
54     end while;
55     end for;
56 end function;

```

Now we can make any proper coloring of an odd ring an unimprovable coloring. Whenever a coloring is improvable at color  $a$  and tree  $T \subseteq F_{R \setminus t_2}^a$ , we will switch the colors given to the vertices in that tree. And we will continue to do so until the coloring is unimprovable. This algorithm will hold after at most  $n$  iterations where  $n$  is the number of vertices in  $R$ . The intuition why this algorithm stops after  $n$  iterations, is that every tree will switch colors at most one time. The proof and algorithm can be found in [14].

**Algorithm 7.** MAKEUNIMPROVABLE( $R, X, c$ )

```

1  #INPUT: An odd ring R together with its ring partition X and a coloring c
2  # on V(R) \ {t2}
3  #OUTPUT: false, color a and subtree of R where it is not unimprovable and c(s1).
4  # true, color c1 if R \ t2 is colored unimprovable
5  #RUNNING TIME: O(n^4)
6  MakeUnimprovable := function(R,X,c);
7     b,a,c1,T := IsUnimprovable(R,X,c);
8     while not b do
9         i := Index(c,a);
10        j := Index(c,c1);
11        for t in T do
12            if {t} subset a then
13                Exclude(~c[i],t);
14                Include(~c[j],t);
15            else
16                Exclude(~c[j],t);
17                Include(~c[i],t);
18            end if;
19        end for;
20        b,a,c1,T := IsUnimprovable(R,X,c);
21    end while;
22    return c,c1;
23 end function;

```

The next theorem connects rings to hyperholes. The notion of unimprovable colorings plays an

important role in the proof given by F. Maffray, I. Penev and K. Vušković [14], because it helps creating hyperholes with a lot of vertices that are contained in the ring.

**Theorem 3.9.** Let  $k \geq 4$  be an integer, and let  $R$  be a  $k$ -ring. Then

$$\chi(R) = \max\{\chi(H) \mid H \text{ is a } k\text{-hyperhole in } R\}.$$

*Proof.* This proof is an overview of the proofs from Lemma 3.5 and Theorem 1.2 of [14]. We will prove it with induction to the number of vertices in a  $k$ -ring  $R$ . Assume for every  $k$ -ring  $R'$  with fewer vertices than  $R$  it contains a  $k$ -hyperhole  $H'$  with  $\chi(H') = \chi(R')$ .

First assume  $\chi(R) = \omega(R)$ . Without loss of generality  $X_1 \cup X_2$  is a clique of size  $\omega(R)$ . Then

$$H := X_1 \cup X_2 \cup \{v_i^1 : 3 \leq i \leq k\}$$

is a hyperhole in  $R$  of the same chromatic number as the ring  $R$  itself. Recall that for  $k$  even, we always have  $\chi(R) = \omega(R)$ .

Second, we take a ring  $R$  of odd length  $k \geq 5$ , with  $\chi(R) > \omega(R)$ . We will construct a hyperhole  $H$  such that  $\chi(H) = \chi(R)$  from an optimal unimprovable coloring  $c$  of  $R \setminus \{t_2\}$ . An optimal coloring exists and earlier we showed how to make an unimprovable coloring  $c'$  for  $R \setminus \{t_2\}$  from a proper coloring  $c$  for  $R \setminus \{t_2\}$  using no more colors than used in  $c$  with Algorithm 7 MAKEUNIMPROVABLE( $R, X, c$ ). We define  $S := \{v \in R : c(v) = c(s_1) \wedge v \neq t_2\}$  and  $r := |c| = \chi(R \setminus \{t_2\})$ ; the number of colors used in  $c$ . We prove three claims; together they cover all possibilities.

**Claim 1.** If in addition to  $k \geq 5, \chi(R) > \omega(R)$ , also  $\omega(R \setminus S) = r$  is satisfied, then  $R$  contains a hyperhole  $H$  such that  $\left\lceil \frac{V(H)}{\alpha(H)} \right\rceil = r + 1$ .

*Proof of Claim 1.* We know  $\omega(R) \leq r$  for otherwise  $\omega(R) = \chi(R)$  and this contradicts our assumption. Thus  $\omega(R) = r = \omega(R \setminus S) = \chi(R \setminus \{t_2\})$  and from this, we can deduce  $X_2 \cup (N_R(t_2) \cap X_3)$  is the unique clique of size  $r$  in  $R \setminus S$ . We can conclude no vertices in this maximal clique are colored with color  $c(s_1)$ . The color  $c$  is unimprovable and  $i = 3$  is odd, so all vertices  $v \in N_R(t_2) \cap X_3$  are connected to  $s_1$  with a path  $(v, u_4^l, \dots, u_k^l, s_1)$  containing only vertices that are colored with the colors  $c(s_1)$  or  $c(v)$ . For odd  $i \geq 5$ , let  $h_i$  be the maximal integer such that  $u_i^{h_i} \in X_i$  is adjacent to two vertices in  $X_{i-1}$  resp  $X_{i+1}$  colored with color  $c(s_1)$ . The construction of  $H = [Z_1, \dots, Z_k]_H$  is as follows:

- $Z_1 := \{s_1\}, Z_2 := X_2, Z_3 := N_R(t_2) \cap X_3$ ;
- for all even  $i \geq 4$  define  $Z_i := \{u_i^l : \exists j \geq l [c(u_i^j) = c(s_1)]\}$
- for all odd  $i \geq 5$  define  $Z_i := \{u_i^l : l \leq h_i\}$  For all colors  $\mathcal{C} \in c(Z_3)$ , there is a vertex in  $Z_i$  that received color  $\mathcal{C}$ .

This is a hyperhole of  $R$ . Notice  $|\{c(s_1) \cup c(Z_2 \setminus t_2) \cup c(Z_3)\}| = r$  and in [14] it is proven that every color from  $\{c(s_1) \cup c(Z_2 \setminus t_2) \cup c(Z_3)\}$  appears on  $\frac{k-1}{2}$  vertices of  $H \setminus \{t_2\}$  and therefore  $|V(H) \setminus \{t_2\}| \geq \frac{k-1}{2} \cdot r$ , thus  $\left\lceil \frac{V(H)}{\alpha(H)} \right\rceil = r + 1$ . Therefore, the ring  $R$  contains a hyperhole  $H$  of the same chromatic number.

**Claim 2.** If in addition to  $k \geq 5, \chi(R) > \omega(R)$ , also  $\omega(R \setminus S) \leq r - 1$  and  $\chi(R \setminus S) \leq r - 1$  are satisfied, then  $\chi(R) = r$  and  $R$  contains a hyperhole  $H$  such that  $\chi(H) = r$ .

*Proof of Claim 2.* The fact  $\chi(R) = r$  follows from the assumption  $\chi(R \setminus S) \leq r - 1$  and  $c$  is an optimal coloring on  $R \setminus \{t_2\}$  using  $r$  colors. The set  $X_2 \setminus \{t_2\}$  is nonempty, because otherwise  $R \setminus \{t_2\}$  is chordal and  $r = \chi(R \setminus \{t_2\}) = \omega(R \setminus \{t_2\}) \leq \omega(R) < \chi(R) = r$ . Every vertex in  $X_2 \setminus \{t_2\}$  has a greater than or equal to neighborhood  $t_2$ . The graph  $R \setminus \{t_2\}$  is a  $k$ -ring, so we can apply the induction hypothesis to gain a hyperhole  $H$  that is contained in  $R \setminus \{t_2\}$  with chromatic number  $\chi(H) = \chi(R \setminus \{t_2\}) = \chi(R)$ .

**Claim 3.** If in addition to  $k \geq 5$ ,  $\chi(R) > \omega(R)$ , also  $\omega(R \setminus S) \leq r - 1$  and  $\chi(R \setminus S) = r$  are satisfied, then  $R$  contains a hyperhole  $H$  such that  $\left\lceil \frac{V(H)}{\alpha(H)} \right\rceil = r + 1$ .

*Proof of Claim 3.* The detailed and complete proof given in [14] is elaborate, so we only show the highlights. Let  $[v_1, \dots, v_t]$  be a maximal simplicial sequence of  $R \setminus S$ . The graph  $R \setminus S$  can not be chordal, because  $\chi(R \setminus S) \neq \omega(R \setminus S)$ . Thus  $R \setminus (S \cup \{v_1, \dots, v_t\})$  is a nonempty  $k$ -ring and by induction hypothesis contains a hyperhole  $H'$  with the same chromatic number. It can be proven that  $\left\lceil \frac{V(H')}{\alpha(H')} \right\rceil = r$ . With this hyperhole  $H'$  and an unimprovable coloring  $c$ , we may construct a hyperhole  $H$  contained in  $R$  such that  $\left\lceil \frac{V(H)}{\alpha(H)} \right\rceil = r + 1$ . Let  $h_i$  be the largest integer such that  $u_i^{h_i} \in X_i \cap V(H')$  and  $l$  the largest odd index such that in every odd  $i \leq l$  the set  $X_i$  contains a vertex colored with the color  $c(s_1)$ . The construction of  $H = [Z_1, \dots, Z_k]$  is as follows:

- for  $i \leq l + 2$  define  $Z_i := \{u_i^j : j \leq h_i\}$ ;
- for even  $i \geq l + 3$  define  $Z_i := \{u_i^j : \exists j' \geq j [c(u_i^{j'}) = c(s_1)] \vee j \leq h_i\}$ ;
- for odd  $i \geq l + 4$  define  $Z_i := \{u \in X_i : \exists v \in X_{i-1} [c(v) = c(s_1) \wedge e_{uv} \in E(R)]\}$ .

This  $H$  is proven to be a hyperhole of size  $V(H) \geq |V(H')| + \frac{k-1}{2}$  and therefore,  $\chi(H) \geq \left\lceil \frac{|V(H)|}{\alpha(H)} \right\rceil \geq \left\lceil \frac{|V(H')|}{\alpha(H')} \right\rceil + 1 = r + 1$ .

We conclude that the ring  $R$  contains a hyperhole  $H$  of the same chromatic number in every case.  $\square$

The following lemma is not an immediate consequence of the previous theorem, but follows directly from the given proof and the observation that Claim 1 holds if  $\omega(R) \leq r$  regardless of whether the condition  $\omega(R) < \chi(R)$  is met or not, and Claim 3 also holds when  $\omega(R) = \chi(R)$ .

**Lemma 3.10.** [14] Let  $R$  be an odd ring. Let  $c$  be an unimprovable coloring of  $R \setminus t_2$ , let  $S = \{x \in V(R) \mid x \neq t_2, c(x) = c(u_1^1)\}$ . Then either  $\chi(R \setminus S) \leq \#c - 1$  or  $\chi(R) = \#c + 1$ .  $\square$

Observe that  $S$  is a stable set. Therefore if  $\chi(R \setminus S) \leq r - 1$ , then  $\chi(R) \leq r$ . So, this lemma gives a tactic on how to color an odd ring  $R$ . Namely, if we can color the graphs  $R \setminus t_2$  and  $R \setminus S$ , we can either expand the coloring of  $R \setminus t_2$  with a new color for  $t_2$  or the coloring  $R \setminus S$  with a new color for the stable set  $S$ . We will use the first expansion if the number of colors used in  $\chi(R \setminus S) \geq r$  (thus  $\chi(R) = r + 1$ ) and the second expansion if otherwise.

The algorithm given in [14] works as follows. The input is an odd ring  $R$  with  $(X_1, \dots, X_k)$  as ring partition and a proper coloring on  $R \setminus t_2$ . First we make the coloring  $c$  unimprovable and define the set  $S := \{x \in V(R) \mid x \neq t_2, c(x) = c(u_1^1)\}$  (lines 8–10). Then we obtain  $R_2 := R \setminus S$  (lines 14–15) and  $R_3 := R \setminus (S \cup \{v_1, \dots, v_t\})$  with  $[v_1, \dots, v_t]$  a maximal simplicial elimination ordering of  $R_2$  (lines 36–39). Recall that  $R_3$  is empty or a ring and we can greedy color  $R_2$  from an optimal coloring of  $R_3$  by reversed simplicial elimination order (line 20 or 49). We define a coloring  $c_1$  on  $R_3 \setminus \{t_2\}$  such that  $c_1 = c|_{V(R_3) \setminus \{t_2\}}$  (lines 22–33). If  $R_3$  is a ring, we check if  $t_2$  is a vertex in  $R_3$  (line 41). If this is not the case, then  $c_1$  is already an optimal coloring for  $R_3$  (line 42). If it is, we can find an optimal coloring for  $R_3$  by making a recursive call with input  $R_3$  and  $c_1$  (lines 44–47). After this, we can easily check if  $\chi(R \setminus S) \leq |c| - 1$  by counting the number of colors used to optimally color  $R_2$  (line 52). A more detailed proof and complexity analysis can be found in [14].

**Algorithm 8.** EXTENDCOLORING( $R, X, c$ )

```

1  #INPUT: An odd ring R together with its ring partition X and a coloring c
2  #      on V(R) \ {t2}
3  #OUTPUT: Proper coloring on R using at most max(r, chi(R)) colors where r = #c
4  #RUNNING TIME: O(n^5)
5  ExtendColoring := function(R, X, c);
6      #Make the coloring c unimprovable.
7      #Make stable set S := v in V(R \ t2) : c(v) = c(s1)

```



```

8      c,S:=MakeUnimprovable(R,X,c);
9      t2:= X[2][|X[2]|];
10     Exclude(~S,t2);
11     r := |c|;
12
13     #Make R2:=R\S
14     W:= {v : v in Vertices(R) | not {v} subset S };
15     R2 := sub<R | W>;
16
17     #Make proper coloring for R2:=R\S
18     simp := MaxSimplicial(R2);
19     if |simp| eq |Vertices(R2)| then
20         c2:= ColorGreedy(R2, [],simp);
21     else
22         c1 := c;
23         simp2 := {v : v in simp};
24         for v in simp2 join S join {t2} do
25             i:= IndexSet(c1,v);
26             if i ne 0 then
27                 if {v} eq c1[i] then
28                     Exclude(~c1,{v});
29                 elif {v} subset c1[i] then
30                     Exclude(~c1[i], v);
31                 end if;
32             end if;
33         end for;
34
35         #Making R3 := R \ (S ∪ simp)
36         for s in simp do
37             Exclude(~W,s);
38         end for;
39         R3 := sub<R | W>;
40
41         if Index(simp,t2) gt 0 then
42             c3 := c1;
43         else
44             b,X3 := IsRing(R3);
45             j := IndexSet(X3,t2);
46             Rotate(~X3,-j+2);
47             c3 := $$ (R3,X3,c1);
48         end if;
49         c2 := ColorGreedy(R2,c3,simp);
50     end if;
51
52     if |c2| le r-1 then
53         c2 := Parent([S]) ! c2;
54         return Include(c2,S);
55     else
56         return Include(c,{t2});
57     end if;
58 end function;

```

We can finally color a ring  $R$  with even and with odd length. Let  $G$  be a graph such that  $G \setminus \{v_1, \dots, v_t\}$  is a ring with  $[v_1, \dots, v_t]$  a maximal simplicial elimination ordering. The following algorithm will color every such graph  $G$  optimally [14]. It is a recursive algorithm that first computes a maximal simplicial elimination ordering  $A$  of  $G$  (line 5). If this ordering includes every vertex of  $G$ , we can simply color  $G$  greedily (lines 6–7). If it is a part of the vertices of  $G$ , then  $G \setminus A$  is a ring (Lemma 3.4) and we make a recursive call with input  $G \setminus A$  and greedily expand the coloring using a reversed simplicial elimination ordering (lines 8–11). If there are no simplicial vertices in  $G$ , then we

check if  $G$  is a ring and obtain its ring partition (line 13). If it is an even ring, we can use Algorithm  $\text{COLOREVENRING}(G, X)$  to color  $G$  (line 18). If it is an odd ring, we can make a recursive call with input  $G \setminus \{t_2\}$  and extend the coloring with  $\text{EXTENDCOLORING}(G, X, c)$ . A more detailed proof and complexity analysis can be found in [14].

**Algorithm 9.**  $\text{COLORRING}(R)$

```

1  #INPUT: A ring R
2  #OUTPUT: Proper coloring on R that uses  $\chi(R)$  colors.
3  #RUNNING TIME:  $\mathcal{O}(n^6)$ 
4  ColorRing := function(G);
5      simp := MaxSimplicial(G);
6      if |simp| eq |Vertices(G)| then
7          return ColorGreedy(G, [], simp);
8      elif |simp| ge 1 then
9          G2 := G - {v : v in simp};
10         c := $$ (G2);
11         return ColorGreedy(G, c, simp);
12     else
13         b, X := IsRing(G);
14         if not b then
15             return false;
16         end if;
17         if IsEven(|X|) then
18             return ColorEvenRing(G, X);
19         else
20             G2 := G - {X[2][|X[2]|]};
21             c := $$ (G2);
22             c := Parent([Vertices(G)[1]]) ! c;
23             return ExtendColoring(G, X, c);
24         end if;
25     end if;
26 end function;

```

# Bibliography

- [1] Benzer, S. (1959). On the topology of the genetic fine structure. *Proceedings of the National Academy of Sciences of the United States of America*, 45(11), 1607–1620.
- [2] Boncompagni, V & Penev, I. & Vušković, K. (2019). Clique-cutsets beyond chordal graphs. *Journal of Graph Theory*, 91(2), 192–246.
- [3] Bosma, W. & Cannon, J. & Playoust, J. (1997). The Magma algebra system. *I. The user language*, *J. Symbolic Comput.*, v. 24, 5017 pages.
- [4] Cameron K. & Eschen E.M. & Hoàng C.T. & Sritharan R. (2006). Recognition of Perfect Circular-arc Graphs. *In: Bondy A., Fonlupt J., Fouquet J.L., Fournier J.C., Ramírez Alfonsín J.L. (eds) Graph Theory in Paris. Trends in Mathematics. Birkhäuser Basel.*, p97–108.
- [5] Eschen, E.M. & Spinrad, J.P. (1993). An  $\mathcal{O}(n^2)$  algorithm for circular-arc graph recognition *SODA*, p128–137
- [6] Garey, M.R. & Johnson, D.S. & Miller, G.L. & Papadimitriou, C.H. (1980). The complexity of coloring circular arcs and chords *SIAM J. Alg. Disc. Math.*, Vol.1, No.2
- [7] Guillermo, D & Grippo, L.N. & Safe, M.D. (2014). Structural results on circular-arc graphs and circle graphs: a survey and the main open problems. *Discrete Applied Mathematics*, 164, 427–443.
- [8] Golumbic, M. C. (1980). Algorithmic graph theory and perfect graphs *New York: Academic Press*
- [9] Hamming, R. W. (2004). Methods of mathematics applied to calculus, probability, and statistics. (*Dover, Ser. Dover books on mathematics*). *Dover Publications*.
- [10] Hsu, W.L. (1981). How to color claw-free perfect graphs. *Ann. Discrete Math.*, vol. 11, 189–197.
- [11] Klee, V. (1969). What Are the Intersection Graphs of Arcs in a Circle? *The American Mathematical Monthly*, vol. 76(7), 810–813.
- [12] Korte, N. & Möhring, R. (1986). A Simple Linear-Time Algorithm to Recognize Interval Graphs. *Conference Paper*, 1–16.
- [13] Lekkerkerker, C. & Boland, J. (1962). Representation of a finite graph by a set of intervals on the real line. *Fundamenta Mathematicae* vol. 51(1), 45–64.
- [14] Maffray, F. & Penev, I. & Vušković, K. (2021). Coloring rings. *Journal of Graph Theory*, 96(4), 642–683.
- [15] McConnell, R (2001). Linear-Time Recognition of Circular-Arc Graphs. *Computer Science Technical Reports*.
- [16] Narayanan, L. & Shende, S. M. (2001). Static frequency assignment in cellular networks. *Algorithmica : An International Journal in Computer Science*, 29(3), 396–409.
- [17] Orlin, J.B. & Bonuccelli, M.A. & Bovet, D.P. (1981). An  $\mathcal{O}(n^2)$  Algorithm for Coloring Proper Circular Arc Graphs. *SIAM Journal on Algebraic and Discrete Methods*, vol. 2(2), 88–93.

- [18] Read, R.C. (1968). An introduction to chromatic polynomials. *Journal of Combinatorial Theory*, 4(1), 52-71.
- [19] Stoffers, K.E. (1968). Scheduling of traffic lights—a new approach. *Transportation Research*, 2(3), 199–234.
- [20] Stahl, F.W. (1967). Circular genetic maps. *Journal of Cellular Physiology*, 70(S1), 1–12.
- [21] Tucker, A. (1971). Matrix characterizations of circular-arc graphs. *Pacific J. Math*, 39(2), 535–545.
- [22] Tucker, A. (1974). Structure theorems for some circular-arc graphs. *Discrete Mathematics*, 7(1), 167–195.
- [23] Tucker, A. (1975). Coloring a Family of Circular Arc. *SIAM Journal on Applied Mathematics*, 29(3), 493-502.
- [24] Tucker, A. (1980). An efficient test for circular-arc graphs. *SIAM J. Comput.* Vol.9, No.1
- [25] Chen, X. & Hu, Z. & Zang, W. (2005). Perfect Circular Arc Coloring. *Journal of Combinatorial Optimization*, 9, 267–280.

# Appendix

## Preliminary algorithms

The first algorithm is to find a maximal elimination ordering for a graph  $G$  [14]. The idea behind this algorithm is the definition of a simplicial vertex. A vertex is simplicial if and only if its neighborhood is a clique. For a graph  $G$ , we calculate

$$\text{Diff}_G(x,y) := \begin{cases} \#(N_G[x] \setminus N_G[y]) & \text{if } x,y \text{ are connected} \\ 0 & \text{if otherwise} \end{cases}.$$

A vertex  $x$  is simplicial iff  $\text{Diff}_G(x,y) = 0$  for all vertices  $y$ . If we found a simplicial vertex  $x$ , then we remove it from the graph, include it in the list, set  $\text{Diff}_{G \setminus x}(x',y')$  correct for all other vertices and repeat the algorithm to look for another simplicial vertex.

**Algorithm 10.** MAXSIMPLICIAL( $G$ ):

```
1  #INPUT: Graph G
2  #OUTPUT: [v1,...,vi]; a maximal elimination ordering for graph G, possible empty.
3  #RUNNING TIME: O(n^3)
4  MaxSimplicial := function(G);
5      V := {v : v in Vertices(G)};
6      if |V| = 0 then
7          return [];
8      end if;
9      A := [[x,y] : x in V, y in V | x ne y];
10     Diff := [0 : i in [1..|A|]];
11
12     #Calculating diff(x,y) for every vertex x ≠ y:
13     for i:=1 to |A| do
14         if A[i][2] in Neighbors(A[i][1]) then
15             d := {x : x in Neighbors(A[i][1]) |
16                 not x in Neighbors(A[i][2]) and not x eq A[i][2]};
17             Diff[i] := |d|;
18         end if;
19     end for;
20
21     L := [Parent(Random(V)) | ];
22     cont := true;
23
24     while cont do
25         cont := false;
26         for x in V do
27             distance := 0;
28             for y in V do
29                 if y ne x then
30                     if Diff[Index(A,[x,y])] gt 0 then
31                         distance :=1;
32                         break y;
```

```

33         end if;
34     end if;
35 end for;
36
37 #If diff(x,y)=0 for all y, then x is a simplicial vertex for G[V]
38 if distance eq 0 then
39     Include(~L,x);
40     Exclude(~V,x);
41     cont := true;
42
43     #update Diff:
44     for x1, y in V do
45         if x1 ne y and x in Neighbors(x1) and
46             (not x in Neighbors(y)) then
47             Diff[Index(A,[x1,y])] -= 1;
48         end if;
49     end for;
50     break x;
51 end if;
52 end for;
53 end while;
54
55 return L;
56 end function;

```

It is well known that a graph  $G$  is chordal if and only if its maximal elimination ordering is an ordering of *all* vertices of the graph. There exists an  $\mathcal{O}(n+m)$  time algorithm to figure out if a graph is Chordal or not [2], but for convenient reasons, we will use the following algorithm [14].

**Algorithm 11.** ISCHORDAL( $G$ ):

```

1  #INPUT: Graph G
2  #OUTPUT: True if G is Chordal and false otherwise.
3  #RUNNING TIME:  $\mathcal{O}(n^3)$ 
4  IsChordal := function(G);
5      L := MaxSimplicial(G);
6      if |L| eq |Vertices(G)| then
7          return true;
8      else
9          return false;
10     end if;
11 end function;

```

The following algorithm is to extend a coloring  $c$  for a graph  $G$  by coloring the vertices in *order* where *order* is a simplicial elimination ordering of  $G$ . Therefore, if  $c$  is a minimal proper coloring, so is the extension.

**Algorithm 12.** COLORGREEDY( $G, c, o$ ):

```

1  #INPUT: Graph G, Coloring c of V(G)\order and the order of
2  #      maximal simplicial elimination ordering of G.
3  #OUTPUT: Coloring of all vertices such that the coloring on c remains the same.
4  #RUNNING TIME:  $\mathcal{O}(n^2)$ 
5  ColorGreedy := function(G, c, order);
6      if |c| eq 0 then
7          c := [Parent({order[1]}) | {order[|order|]} ];
8      end if;
9      for i:= |order| to 1 by -1 do
10         j := 1;
11         Neigh := Neighbors(order[i]);

```

```

12   cont := true;
13   while cont and j le |c| do
14       cont := false;
15       #Check if there is a neighbor of order[i] already colored with
16       #color c[j]
17       for v in c[j] do
18           if v in Neigh then
19               cont := true;
20               j+=1;
21               break v;
22           end if;
23       end for;
24   end while;
25   if j le |c| then
26       Include(~c[j],order[i]);
27   else
28       Include(~c, {order[i]});
29   end if;
30 end for;
31 return c;
32 end function;

```

If a graph is chordal, we have a simplicial elimination ordering. With this ordering, we can color the graph optimal by greedy coloring the reversed order.

**Algorithm 13.** COLORCHORDAL( $G$ ):

```

1   #INPUT: Chordal graph G
2   #OUTPUT: Optimal coloring of G
3   #RUNNING TIME:  $\mathcal{O}(n^2)$ 
4   ColorChordal := function(G);
5       ord := MaxSimplicial(G);
6       return ColorGreedy(G, [], ord);
7   end function;

```

The following is a simple, but handy algorithm.

**Algorithm 14.** INDEXSET(Set,v)

```

1   #INPUT: A sequence of sets or sequences called 'Set' and a variable v.
2   #OUTPUT: The first index i such that Set[i] contains v and 0 otherwise.
3   IndexSet := function(Set, var);
4       for i:= 1 to |Set| do
5           if {var} subset Set[i] then
6               return i;
7           end if;
8       end for;
9   end function;

```

## Algorithms for hyperholes

**Algorithm 15.** MAKEHYPERHOLE( $X$ ):

```

1   #INPUT: Enumerated list I of integers
2   #OUTPUT: The hyperhole H where the clique-size of  $X_i$  is equal to  $I[i]$ .
3   #RUNNING TIME: ??
4   MakeHyperhole := function(X);
5       n := 0;

```

```

6   for i in X do
7       n += i;
8   end for;
9
10  G := Graph<n|>;
11  V := Vertices(G);
12  koekjes := 1;
13  for i:=1 to |X| do
14      for l:=0 to X[i]-1 do
15          #make complete to X[i+1]
16          for m:=0 to X[i mod |X| +1]-1 do
17              G += {V[koekjes+l], V[(koekjes + X[i] + m -1) mod n +1]};
18          end for;
19
20          #make X[i] a clique
21          for k:=l+1 to X[i]-1 do
22              G += {V[koekjes+l], V[koekjes + k]};
23          end for;
24      end for;
25      koekjes += X[i];
26  end for;
27  return G;
28 end function;

```

## Algorithms for rings

The following algorithm is based on Bubble Sort.

**Algorithm 16.** SORTCLIQUE( $G, X_i$ ):

```

1   #INPUT: Graph G,  $X_i := [v_1, \dots, v_t]$  such that  $G[X_i]$  is a clique
2   #OUTPUT: True together with an ordering if  $X_i$  can be ordered such that
3   #          $N_G[v_{i'}] \subseteq \dots \subseteq N_G[v_{i'']}$ . False otherwise.
4   #RUNNING TIME:  $\mathcal{O}(t \cdot n)$ 
5   SortClique := function(G, Xi);
6       i:=1;
7       while i le |Xi| do
8           j := i;
9           while j ge 2 and Degree(Xi[j-1]) lt Degree(Xi[j]) do;
10              y:= Xi[j-1];
11              Xi[j-1] := Xi[j];
12              Xi[j] := y;
13              j-:=1;
14          end while;
15          i+=1;
16      end while;
17
18      #Check if the ordering is correct
19      for j:= 1 to |Xi|-1 do
20          if not Include(Neighbors(Xi[j+1]), Xi[j+1]) subset
21              Include(Neighbors(Xi[j]), Xi[j]) then
22              return false, [];
23          end if;
24      end for;
25      return true, Xi;
26 end function;

```



The following algorithm determines the clique size of a Ring. It is based on Lemma 2.6 [14] which is based on Lemma 8.25 [2].

**Algorithm 17.** CLIQUESIZERING( $R, X$ ):

```
1  #INPUT: Ring R together with its ring partition X.
2  #OUTPUT: The clique size  $\omega(R)$  of ring R.
3  #RUNNING TIME:  $\mathcal{O}(n^3)$ .
4  CliqueSizeRing := function(R,X);
5      V := Vertices(R);
6      R1 := sub<R | {v : v in V | not v in X[1]}>;
7      R2 := sub<R | {v : v in V | not v in X[3]}>;
8      c1 := |ColorGreedy(R1, [], MaxSimplicial(R1))|;
9      c2 := |ColorGreedy(R2, [], MaxSimplicial(R2))|;
10     return Max({c1,c2});
11 end function;
```