3-coloring Circle Graphs in Theory and Practice

Master Thesis of

Patricia Bachmann

At the Department of Informatics and Mathematics
Chair of Theoretical Computer Science

Reviewers:	Prof. Dr. Ignaz Rutter
	Prof. Dr. Dirk Sudholt
Advisors:	Peter Stumpf

Time Period: 9th June 2022 – 5th December 2022

Statement of Authorship

I hereby declare that this document has been composed by myself and describes my own work, unless otherwise acknowledged in the text.

Passau, December 5, 2022

Abstract

Let $G = (V, E)$ be an undirected graph. We call G a *circle graph* if the vertices of G can be represented as a set of chords on a circle such that two chords intersect if and only if their respective vertices are adjacent. The *coloring problem* is the problem of finding the smallest number of colors k such that the vertices of a graph G can be assigned a color such that any two adjacent vertices of G have distinct colors. Asking if G can be colored this way with at most k colors, is called the *k-coloring problem*. The coloring problem and k-coloring problem for $k \geq 4$ are known to be $\mathcal{NP}$-hard and $\mathcal{NP}$-complete respectively for circle graphs. Walter Unger claimed in his Ph.D. thesis that the 3-coloring problem can be solved in time $O(n \log n)$ for circle graphs [51]. We discuss the ideas and arguments he presents regarding the 3-coloring of circle graphs. We introduce an auxiliary coloring function $c^* : V(G)^2 \to \mathbb{B}$ which is defined on pairs of vertices that share a common neighbor but are themselves not adjacent to each other. We give constraints which c^* has to satisfy on so-called *important subgraphs*. This function c^* is called a *consistent auxiliary coloring function* if there exists a 3-coloring for each important subgraph H such that two vertices a and b of H have the same color if and only if $c^*(a, b) = \texttt{true}$. Such a 3-coloring is said to *realize* c^*. For a connected circle graph G, a 3-coloring that realizes $c*$ is unique up to permutation of the colors. We present a counterexample to the claim that G is 3-colorable if and only if there exists a consistent auxiliary coloring function c^*. The counterexample gives a consistent auxiliary coloring function c^* for which there exists no 3-coloring that realizes it. We also discuss the proposed running time of $O(n \log n)$ for solving 3-coloring of circle graphs. To this end, we highlight a certain step of the algorithm to compute c^* and argue why it is not clear that this running time holds. Lastly, we determine the efficiency and accuracy of his algorithms for computing c^* and the corresponding 3-coloring through experimental evaluation. We conclude that the 3-coloring problem should be considered an open problem for circle graphs.

Deutsche Zusammenfassung

Sei $G = (V, E)$ ein ungerichteter Graph. Wir nennen G einen *Kreissehnengraphen*, wenn $V(G)$ als eine Menge von Sehnen auf einem Kreis dargestellt werden kann, die sich genau dann schneiden, wenn die entsprechenden Knoten benachbart sind. Das *Färbungsproblem* ist das Problem, die kleinste Anzahl von Farben k zu finden mit der die Knoten von G so gefärbt werden können, dass je zwei benachbarte Knoten von G unterschiedliche Farben haben. Die Frage, ob G so mit höchstens k Farben gefärbt werden kann, wird als *k-Färbungsproblem* bezeichnet. Sowohl das Färbungsproblem als auch das k-Färbungsproblem für $k \geq 4$ sind bekanntermaßen $\mathcal{NP}$-schwer, bzw. $\mathcal{NP}$-vollständig für Kreissehnengraphen. Walter Unger behauptete in seiner Doktorarbeit, dass das 3-Färbungsproblem in der Zeit $O(n \log n)$ für Kreisgraphen gelöst werden kann [51]. Wir betrachten die Ideen und Argumente, die er bezüglich der 3-Färbung von Kreisgraphen präsentiert. Wir führen eine Färbungshilfsfunktion $c^* : V(G)^2 \to \mathbb{B}$ ein, die für Paare von Knoten definiert ist, die einen gemeinsamen Nachbarn haben, aber selbst nicht benachbart sind. Wir geben Bedingungen an, welche c^* auf sogenannten *wichtigen Teilgraphen* erfüllen muss. Diese Funktion c^* wird eine *konsistente Färbungshilfsfunktion* genannt, wenn es eine 3-Färbung für jeden wichtigen Teilgraphen H gibt, so dass zwei Knoten a und b von H genau dann gleich gefärbt sind, wenn $c^*(a, b) = \texttt{true}$. Wir sagen, dass so eine Färbung c^* *realisiert*. Eine 3-Färbung die c^* für einen zusammenhängenden Kreissehnengraphen G realisiert ist eindeutig bis auf Permutation der Farben. Wir stellen ein Gegenbeispiel zu der Behauptung vor, dass G genau dann 3-färbbar ist, wenn es eine konsistente Färbungshilfsfunktion c^* gibt. Im Gegenbeispiel konstruieren wir eine konsistente Färbungshilfsfunktion c^* für die keine 3-Färbung existiert welche c^* realisiert. Wir behandeln außerdem die behauptete Laufzeit von $O(n \log n)$ für das Lösen von 3-Färbung von Kreissehnengraphen. Hierfür heben wir einen bestimmten Schritt des Algorithmus' hervor, welcher c^* berechnet, und zeigen, warum es nicht klar ist, dass die Laufzeit gilt. Schließlich bestimmen wir die Effizienz und Genauigkeit seiner Algorithmen zur Berechnung von c^* und der entsprechenden 3-Färbung durch experimentelle Evaluation. Wir kommen zu dem Schluss, dass das 3-Färbungsproblem als ein offenes Problem für Kreissehnengraphen betrachtet werden sollte.

Contents

1. Introduction

Let $G = (V, E)$ be a graph. Finding the smallest number of colors necessary to color G, i.e. assign a color to each vertex, such that any two adjacent vertices have distinct colors is called the *coloring problem*. This smallest number is also called the *chromatic number* and is denoted by $\chi(G)$. Asking whether G can be colored this way using at most k colors is called the *k-coloring problem*. Graph coloring is a very interesting and well-studied topic in graph theory. It dates back as far as the 19th century when Francis Guthrie tried to color the map of English counties and noticed that he only needed for different colors. His observations were later published by his brother Frederick [24] and their conjecture would later become what is known today as the four color theorem. In 1976 Appel and Haken [1] presented the proof for the four color theorem. Their proof caused a sensation at the time, since it was a computer-assisted proof for a major theorem [53]. Their result was confirmed in 1997 by Roberts et al. [43] who presented a simpler proof. Deciding if a graph is 2-colorable, or *bipartite*, can be done via a breadth-first or depth-first search, i.e . in linear time. Solving the k-coloring problem for any $k \geq 3$, however, is $\mathcal{NP}$-hard [30]. The fastest known algorithms for deciding the 3- and 4-coloring problem are respectively $O(1.3289^n)$ [4] and $O(1.7272^n)$ [16] where n is the number of vertices. For *perfect graphs*, the chromatic number can be computed in polynomial time [23]. *Perfect graphs* are graphs where for each induced subgraph the size of the largest clique is equal to the chromatic number. In other words, if a perfect graph G has a largest clique of size k then G can be colored with k colors.

An *intersection graph* is an undirected graph $G = (\mathcal{F}, E)$ that has a family of nonempty sets $\mathcal{F} = \{S_1, \ldots, S_n\}$ as its vertex set. Two vertices S_i and S_j with $i \neq j$ are adjacent if and only if their corresponding sets intersect, that is $S_i \cap S_j \neq \emptyset$. Restricting the family of sets to different geometric objects yields a number of different intersection graph classes. Certain classes of intersection graphs are perfect graphs, for example *interval graphs*. An *interval graph* is an intersection graph whose vertices are sets of intervals on the real line, see Figure 1.1(b) for an example. Interval graphs are a quite prominent graph class and are well-researched. They were first introduced by Hajós in 1957 [27] and have been further characterized by Gilmore and Hoffman [21], Lekkerkerker and Boland [35] and Fulkerson and Gross [17]. An efficient recognition algorithm using the consecutive-1's property of incidence matrices was also presented by the latter [17]. Their result was later improved by Booth and Lueker who were able to prove the first linear time algorithm for interval graph recognition [7]. Further linear time recognition algorithms were presented by Habib et al. [25], who used lexicographic breadth-first search for their fairly simple

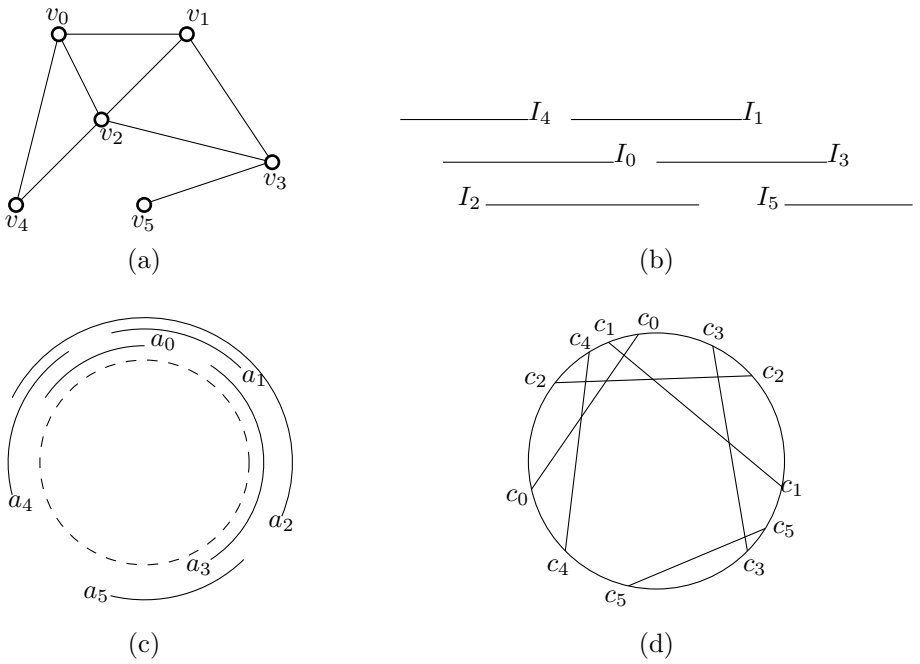


Figure 1.1.: (a) A graph G (b) Graph G represented as a set of intersecting intervals (c) Graph G represented as a set of intersecting arcs on the circle (d) Graph G represented as a set of intersecting chords on the cycle

recognition algorithm, and Corneil et al. [13], who describe a similar approach using 6-sweep LexBFS. Interval graphs are perfect, and solving the coloring problem can be solved even more efficiently than for perfect graphs in general, namely in linear time using a greedy heuristic [39]. The k-coloring of a maximum cardinality subset of intervals can also be solved in time $O(k + n)$, assuming the intervals are already sorted by their endpoints [8]. Other generally hard problems that can be solved efficiently on interval graphs are the *maximum clique problem* and the *maximum independent set problem* [23]. One of the earliest applications of interval graphs is a study on the fine structure of genes by S. Benzer [5]. Further applications for interval graphs include psychophysics [28], psychology [12, 42], archaeology [32, 42] as well as the phasing of traffic lights and the mobile radio frequency assignment problem [41].

A natural generalization of interval graphs are so-called *circular arc graphs*. A *circular arc graph* the intersection graph of ars around a circle, see Figure 1.1(c) for an example. By their definition, every interval graph is a circular arc graph, although, in contrast to interval graphs, ciruclar arc graphs are not perfect. Circular arc graphs are used to illustrate objects or processes of a circular or repetitive nature, which is a interesting property to study. First results for circular arc graphs were published by Hadwiger et al. [26], Klee [33] and Tucker [49]. The first characterizations of circular arc graphs were given by Tucker [47] and Gavril [19]. Tucker also introduced two subclasses of circular arc graphs, namely *unit circular arc graphs* and *proper circular arc graphs*, which are characterized by forbidden induced subgraphs. Tucker also published the first polynomial time recognition algorithm for circular arc graphs [48]. The first linear time recognition algorithm was given by McConnel [36] and another simpler linear time recognition algorithm was published by Kaplan and Nussbaum [29]. While the coloring problem for circular arc graphs is $\mathcal{NP}$-complete, the 3-coloring problem can be solved in polynomial time for circular arc graphs in general [18] and in $O(n^2)$ time for proper circular arc graphs [40]. Their applications include combinatorial auctions with structured item graphs [11] and

routing ring networks [45]. For a more in-depth analysis on structural results for circular arc graphs we refer to a survey by Durán et al. [14]

If we can represent G as chords on a cycle instead of arcs, then G is called a *circle graph*, see Figure 1.1(d). Circle graphs were originally introduced by Even and Itai [15] for solving the problem of realizing a given permutation using a minimum number of parallel queues. Their problem was equivalent to finding the chromatic number of a circle graph, which is, similar to circular arc graphs, $\mathcal{NP}$-complete [18]. Also similar to cicular arc graphs, circle graphs are not perfect. Several different characterizations and recognition algorithms have been given for circle graphs over the last decades [37, 20, 44, 22]. Most notably, Spinrad's recognition algorithm runs in time $O(n^2)$ [44] while the one presented by Gioan et al. runs in time$O((n+m))\alpha(n+m)$ where α is the inverse of the Ackerman function [22]. Several $\mathcal{NP}$-complete problems can be solved in polynomial time on circle graphs, such as determining a circle graph's treewidth [34], finding a maximum clique [46] as well as finding a maximum independent set on an unweighted circle graph [38]. Other problems however remain $\mathcal{NP}$-complete for circle graphs. These include a number of *dominating set* problems [31] as well as the k-coloring problem for $k \geq 4$ [50]. The k-coloring of circle graphs has several interesting theoretical and practical applications [3, 9, 6]. This, in part, stems from the fact that solving the k-coloring problem for circle graphs is equivalent to finding a *k-page book embedding* with a given arrangement of the vertices on the *spine*, see Figure 1.2 for an illustration. A *book embedding* is a planar embedding of a graph

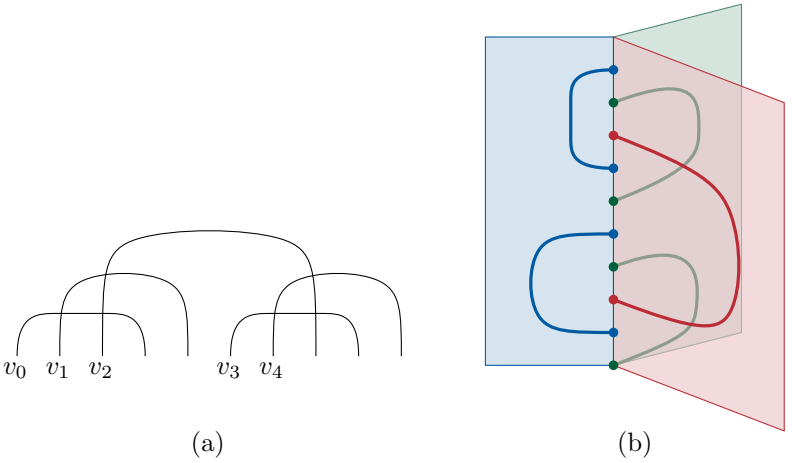


Figure 1.2.: (a) A circle graph G (b) Finding a 3-coloring of G is equivalent to finding a 3-page book embedding for the graph whose vertices are those on the book spine and the edges are the chords connecting them

onto several half-planes, called the *pages*, sharing the same line as one of their boundaries. The vertices of a graph are usually required to lie on that shared boundary, the so-called *spine*. The edges must stay within a single page and edges sharing the same page must not cross each other. The *book thickness* of a graph is the minimum number of pages needed for any book embedding and a k-page book embedding is a book embedding with k pages. Given a graph G together with a fixed vertex spine ordering, we can place the vertices on the boundary of a circle in the given order and add the edges such that they form chords on that circle. From this drawing Γ of G we get the circle graph G' as follows. For each edge of G we have a corresponding vertex in G'. Two vertices of G' are adjacent if and only if the corresponding edges of G cross in Γ. Clearly, G' is a circle graph and a drawing Γ' of G' in which the vertices of G' are represented as chords on a circle is equivalent to Γ. A coloring of G' is a partition of the edges of G into subsets which can be drawn onto a single page without any crossings. Therefore, finding the chromatic number for circle graphs is equivalent to an optimal book embedding. Since the two problems

are so closely related, applications for book embeddings are also of high interest when discussing circle graphs. Book embeddings were studied for applications in "Very Large Scale Integration" (VLSI) design by Chung et al . [9] In fact, this was one of the original motivations for studying book embeddings. VLSI is the process of creating an integrated circuit by combining large numbers of transistors onto a single chip. Here, the vertices of a book embedding represent components of a circuit and the edges represent wires between those components. Chung et al. also proposed sorting permutations using stacks as an application for book embeddings [9]. Scheduling the phases of a traffic signal at a controlled intersection was described by Kainen [6] as a book embedding problem as follows. Incoming and outgoing lanes of roads at an intersection are represented as vertices placed on the spine of a book embedding. Their position on the spine is given by their clockwise order around the intersection. The edges of the graph represent paths through the intersection to get from an incoming lane to an outgoing lane. A subset of edges then represents paths that can all be traversed simultaneously without creating any interference if and only if the subset does not contain a pair of edges that would cross when embedded to the same page in a book embedding. A book embedding therefore describes a partition of intersection traversals into non-interfering subsets. The book thickness is then the minimum number of distinct phases needed for a traffic signal schedule that includes all possible traffic paths that can be taken through the intersection at the same time.

In his Ph.D. thesis, Unger [51] gives $\mathcal{NP}$-completeness proofs for the 4-coloring problem for circle graphs and for the $(2 \cdot k - 1)$-coloring for circle graph with a maximum clique of size k. He also lists several efficient algorithms in his thesis, most notably an algorithm that he claims solves the 3-coloring problem in time $O(n \log n)$ for circle graphs [51]. His algorithm relies on a so-called *auxiliary coloring function* $c^* : V(G)^2 \to \mathbb{B}$, or val_3 as he called it, which is defined for all pairs of vertices that share a common neighbor but are themselves not adjacent to each other. If c^* satisfies a number of constraints for certain *important subgraphs* of a circle graph G then, according to Unger, G is also 3-colorable. This 3-coloring is further claimed to be unique up to permuting the colors. We note that the above claim was published in an extended abstract [52], but, as also pointed out by David Epstein [1], it omits many details. Unfortunately, a journal version including the missing elements was never published. Instead, the paper refers to Unger's Ph.D. thesis for a complete version.

We discuss Unger's ideas, concepts and proofs for solving the 3-coloring of circle graphs. We point to problems in his statements and argue why finding an efficient algorithm that solves 3-coloring on circle graphs should be considered an open problem. We start off with definitions for circle graphs in Chapter 2. In Chapter 3, we introduce the *auxiliary coloring function* c^*. We give a number of constraints for c^* on certain subgraphs which we call *important subgraphs*. In Chapter 3 we also present a counterexample to one of the more crucial lemmas needed to prove that a circle graph is 3-colorable if and only if there exists an auxiliary coloring function c^* that satisfies certain constraints on important subgraph. In Chapter 4 we show how this constrained auxiliary coloring function c^* can be computed. The heart of this chapter is the backtracking algorithm used to compute values for c^* for certain chordless cycles of length at least 5. We discuss Unger's claim that the number of leaves for the backtracking tree is in $O(n \log n)$ and show why it is not clear that this holds. Lastly we present an experimental evaluation of the algorithms described in Unger's Ph.D. thesis and discuss its efficiency in practice in Chapter 5. We also compare the backtracking approach to one using a SAT solver. Finally, we present our conclusions and some open problems in Chapter 6.

[1] https://11011110.github.io/blog/2014/08/09/three-colorable-circle-graphs.html

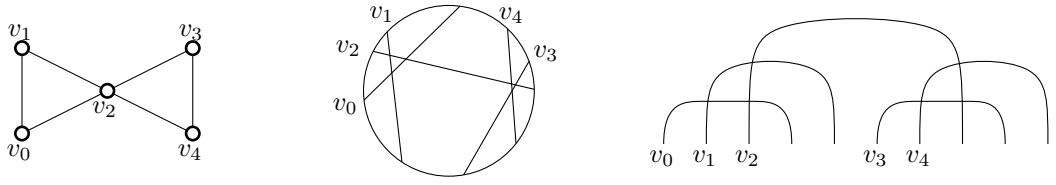
2. Preliminaries

Let $G = (V, E)$ be an undirected graph. A function $\text{col} : V \to \{1, \ldots, k\}$ is called a *k-coloring* of G if every pair of adjacent vertices $a, b \in V$ have different colors assigned to them, i.e. $\text{col}(a) \neq \text{col}(b)$. We define $\text{col}(W)$ for a set of vertices $W \subseteq V$ as the set $\{\text{col}(v) : v \in W\}$. We say that a 3-coloring is *unique* if G has only one possible 3-coloring up to permutation of the colors.

A *circle graph* is an undirected graph G that can be represented using intersecting chords in a circle. We represent each vertex $v \in V(G)$ as such a chord where two chords $v, u \in V(G)$ intersect if and only if $\{v, u\} \in E(G)$. We can assume that no two chords share a common endpoint on the circle. An illustration of a circle graph is given in Figure 2.1(b).

Throughout this thesis we assume G is connected and does not contain cliques of size 4 or greater. Our results and definitions can easily be extended to disconnected graphs by applying them to each connected component. Since we are interested in 3-colorings, circle graphs with cliques of size 4 or larger can be efficiently deemed not 3-colorable [46]. For our purposes, we use a representation of circle graphs in which the circle is cut open and rolled out such that the chords form arcs on a straight line, see Figure 2.1(c). Each chord $v \in V(G)$ has a left and right endpoint on the straight line. Since no two chords share a common endpoint, we can number the endpoints from left to right and uniquely define v by its two endpoints $v^\ell, v^r \in \mathbb{N}$ with $v^\ell < v^r$. From this representation of circle graphs we can infer the following definitions.

Let G be a circle graph. We say that a chord $v \in V(G)$ *crosses the left endpoint* of another chord $u \in V(G)$ with $v \neq u$ if and only if $v^\ell < u^\ell < v^r < u^r$. Analogously, v *crosses the right endpoint* of u if and only if $u^\ell < v^\ell < u^r < v^r$.



(a) (b) (c)

Figure 2.1.: (a) An undirected graph G (b) Representation of G of chords on a circle (c) Alternative circle graph representation for G

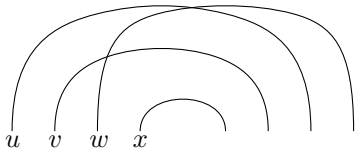


Figure 2.2.: Chord u encases both v and x and directly encases v, v and w directly encase x

We say v *crosses* u if v crosses either the left or the right endpoint of u.

We say that $c \in V(G)$ is the *minimum chord* of a set of chords $W \subseteq V(G)$ or $\min(W)$ if and only if for all chords $v \in W$, $c^\ell \leq v^\ell$. Analogously, c is the *maximum chord* in W or $\max(W)$ if and only if for all chords $v \in W$, $c^r \geq v^r$.

We define the *left neighborhood* of a chord v in G as

$$\mathcal{N}_G^\ell(v) \coloneqq \{u \in V(G) \colon u \text{ crosses the left endpoint of } v\}.$$

Analogously, the *right neighborhood* of v is defined as

$$\mathcal{N}_G^r(v) \coloneqq \{u \in V(G) \colon u \text{ crosses the right endpoint of } v\}.$$

The *neighborhood* of v is then defined as

$$\mathcal{N}_G(v) \coloneqq \{u \in V(G) \colon u \text{ crosses } v\}.$$

The neighborhood of a set of chords $W \subseteq V(G)$ is defined as

$$\mathcal{N}_G(W) \coloneqq \bigcup_{w \in W} \mathcal{N}_G(w).$$

The definitions for left and right neighborhoods of sets of chords are defined analogously as $\mathcal{N}_G^\ell(W) \coloneqq \bigcup_{w \in W} \mathcal{N}_G^\ell(w)$ and $\mathcal{N}_G^r(W) \coloneqq \bigcup_{w \in W} \mathcal{N}_G^r(w)$.

Since the chords are represented by arcs on a straight line, we can define the notion of one chord *encasing* another chord which gives us a level-like structure as follows.

We say that a chord $v \in V(G)$ *encases* another chord $u \in V(G)$ if and only if $v^\ell < u^\ell < u^r < v^r$. We say that v *directly encases* u if there is no other chord w such that v encases w and w encases u. See Figure 2.2 for an example.

For two sets $U, W \subseteq V(G)$ we say that U (directly) encases W if and only if for all $w \in W$ there exists some chord $u \in U$ such that u (directly) encases w.

We recursively define the *levels* of a circle graph G as follows:

$\text{level}_G(1) \coloneqq \{v \in V(G) \colon \nexists u \in V(G) \colon u \text{ encases } v\}$
$\text{level}_G(l) \coloneqq \{v \in V(G) \colon \exists u \in \text{level}_G(l-1) \colon u \text{ directly encases } v\}$

The *position* of a chord v is defined as $\text{pos}(v) = i$ if and only if $v \in \text{level}_G(i)$. The *level number* of G is defined as $L(G) \coloneqq \max\{i \colon \text{level}_G(i) \neq \emptyset\}$.

3. 3-Coloring Circle Graphs using c^*

Let G be a 3-colorable circle graph where col is a 3-coloring for G. If we have such a coloring, we can define a function $f : V(G)^2 \to \mathbb{B}$ that tells us if a pair of vertices has the same color or not, i.e. $f(a, b) = \texttt{true}$ if $\text{col}(a) = \text{col}(b)$ and $f(a, b) = \texttt{false}$ otherwise. What is more interesting is to see if constructing a 3-coloring using a function similar to f is also possible. We know that adjacent vertices can never have the same color, so defining values of f for them is unnecessary. Instead, it suffices to define f for all pairs of vertices that share a common neighbour and are not adjacent to each other. We give constraints for f on certain subgraphs such that values of f are equivalent to a 3-coloring on these subgraphs. We then try and color the graph level by level using these subgraphs as connections between levels, starting with $\text{level}_G(1)$. We show that while this coloring strategy is promising at first, there is a counterexample illustrating how this way of constructing a 3-coloring can result in an incorrect coloring.

3.1. Auxiliary Coloring Function c^*

We define an *auxiliary coloring function* c^* for vertices that share a common neighbor and are not adjacent themselves as follows.

Definition 3.1 (Auxiliary Coloring Function)**.** *Let G be a circle graph. We define a* auxiliary coloring function $c^* : \mathcal{D} \to \mathbb{B}$ *with* $\mathcal{D} \coloneqq \{(a, b) \in V(G)^2 : b \in \mathcal{N}_G(\mathcal{N}_G(a)) \setminus \mathcal{N}_G(a)\}$ *as a function that assigns a boolean value to all pairs of vertices* $(a, b) \in \mathcal{D}$*.*

Next, we establish the connection between c^* and a 3-coloring for G. We want that the values of c^* for all pairs of vertices $(a, b) \in \mathcal{D}$ correspond to the fact that these vertices have the same color in a 3-coloring.

Definition 3.2 (Realizable Auxiliary Coloring Function)**.** *Let G be a circle graph and c^* an auxiliary coloring function. We say that c^* is* realizable *if and only if there exists a* 3*-coloring* col *such that* $c^*(a, b) = \texttt{true}$ *if and only if* $\text{col}(a) = \text{col}(b)$*, and* $c^*(a, b) = \texttt{false}$ *otherwise. We say that* col realizes c^* *and* c^* satisfies col*.*

It arguably would not make sense to use a realizable auxiliary coloring function on a whole circle graph G as it already assumes the existence of a 3-coloring col for G. Instead, we introduce a number of *important subgraphs*, for which we prove the existence of such a

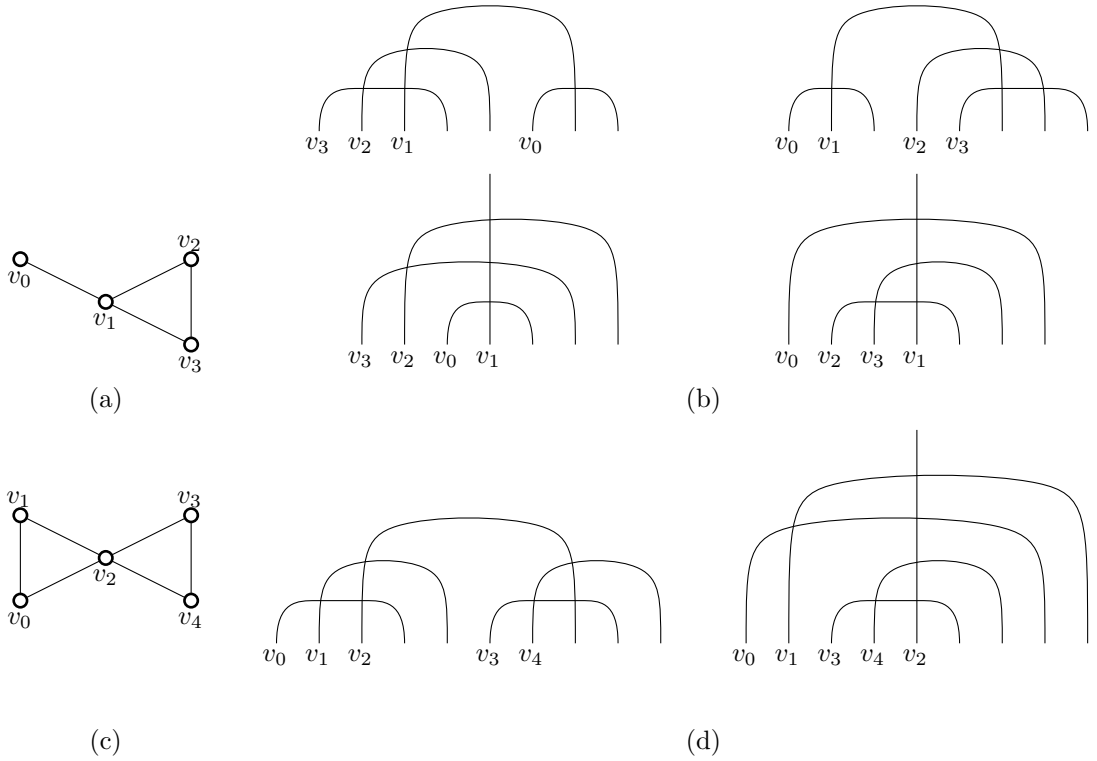


Figure 3.1.: Important subgraph $G_{\triangleleft}$. (a) and (c) represent the possible (underlying) graphs, (b) and (d) show the possible circle graph representations

realizable auxiliary coloring function c^*. We note that the definitions given in the following are derived from both the definitions of the equivalent important subgraphs from Unger's thesis [51, p. 78] and the respective recognition algorithms [51, p. 146] he presents.

Definition 3.3 (Important Subgraph $G_{\triangleleft}$)**.** *Let G be a circle graph and let G' be a subgraph of G. Then G' is the* important subgraph $G_{\triangleleft}$ *if it is isomorphic to the graph illustrated in Figure 3.1(a) or (c) and there exists a chord $c \in V(G')$ such that either*

- $V(G') = (\mathcal{N}_G(c) \cup \{c\}) \cap \text{level}_G(\text{pos}(c))$*, i.e. G' lies within one level, and*
- *chord a does not cross chord b for all $a \in \mathcal{N}^{\ell}_{\text{level}_G(\text{pos}(c))}(c)$ and $b \in \mathcal{N}^{r}_{\text{level}_G(\text{pos}(c))}(c)$*

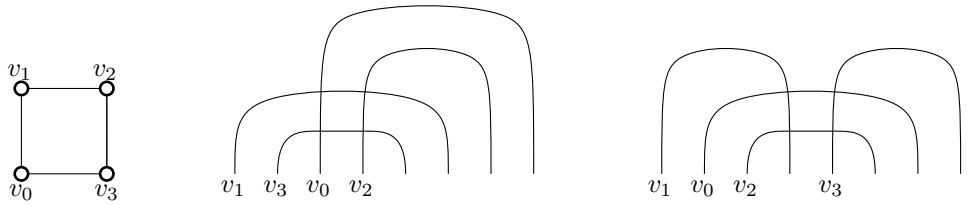
or, for $l < L(G)$ and $H \subseteq \text{level}_G(l)$, $H' \subseteq \text{level}_G(l+1)$

- $V(G') = (\mathcal{N}_G(c) \cap (\text{level}_G(l) \cup \text{level}_G(l+1))) \cup \{c\}$*, i.e. the neighbors of c lie on exactly two adjacent levels, and*
- *H directly encases H' and*
- $c = \min\left(\bigcap_{v \in H \cup H'} \mathcal{N}_G(v)\right)$

then G' is the important subgraph $G_{\triangleleft}$.

The chord c can be considered the "center" of $G_{\triangleleft}$. In Figure 3.1(a) $c = v_1$ and in Figure 3.1(c) $c = v_2$. The two possible underlying structures of $G_{\triangleleft}$ illustrated in Figure 3.1(a) and (c) depend on the size of c's neighborhoods.

Definition 3.4 (Important Subgraph $G_{\square}$)**.** *Let G be a circle graph and let G' be a subgraph of G with $V(G') = \{v_0, \ldots, v_3\}$ where v_i crosses $v_{(i+1) \bmod 4}$ for all $i \in \{0, \ldots, 3\}$, v_1 does not cross v_3 and v_0 directly encases v_2. If G' is isomorphic to the graph illustrated in Figure 3.2(a) and if either v_1 directly encases v_3 or*



(a) (b)

Figure 3.2.: Important subgraph $G_{\square}$. (a) represents the underlying graph, (b) shows the possible circle graph representations

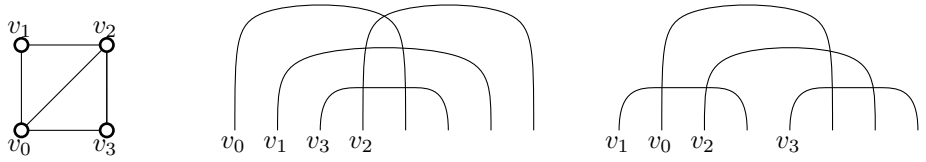
- $v_1^r < v_3^\ell$ *and*
- $v_1 = \max(\mathcal{N}_G^\ell(v_1) \cup \mathcal{N}_G^\ell(v_2))$ *and*
- $v_3 = \min(\mathcal{N}_G^r(v_1) \cup \mathcal{N}_G^r(v_2))$

then G' *is the* important subgraph $G_{\square}$.

Definition 3.5 (Important Subgraph $G_{\boxslash}$)**.** *Let G be a circle graph and let G' be a subgraph of G with $V(G') = \{v_0, \ldots, v_3\}$ where v_1 does not cross v_3 and v_0, v_1, v_2 and v_0, v_2, v_3 form a 3-clique each. If G' is isomorphic to the graph illustrated in Figure 3.3(a) and if either v_1 directly encases v_3 or if*

- $v_1^r < v_3^\ell$ *and*
- $v_1 = \max(\mathcal{N}_G^\ell(v_0) \cap \mathcal{N}_G^\ell(v_2))$ *and*
- $v_3 = \min(\mathcal{N}_G^r(v_0) \cap \mathcal{N}_G^r(v_2))$

then G' *is the* important subgraph $G_{\boxslash}$.



(a) (b)

Figure 3.3.: Important subgraph $G_{\boxslash}$. (a) represents the underlying graph, (b) shows the possible circle graph representations

Definition 3.6 (Important Subgraph $G_{\pentagon}$)**.** *Let G be a circle graph and let G' be a subgraph of G with $V(G') = \{v_0, \ldots, v_4\}$ where v_i crosses $v_{(i+1) \bmod 5}$ and does not cross $v_{i+2 \bmod 5}$ for all $i \in \{0, \ldots, 4\}$. Then G' is the important subgraph $G_{\pentagon}$ if it is isomorphic to the graph illustrated in Figure 3.4(a) and if either*

- v_0 *directly encases* v_2 *and* v_3 *and*
- $v_1 = \max((\mathcal{N}_G^\ell(v_2) \setminus \mathcal{N}_G^\ell(v_3)) \cap \mathcal{N}^\ell(v_0))$ *and*
- $v_4 = \min((\mathcal{N}_G^r(v_3) \setminus \mathcal{N}_G^r(v_2)) \cap \mathcal{N}_G^r(v_0))$

or if

- $\{v_0, v_1\}$ *directly encases* $\{v_2, v_3, v_4\}$ *and*
- $v_1 = \max((\mathcal{N}_G^r(v_2) \setminus \mathcal{N}_G^r(v_3)) \cap \mathcal{N}^r(v_0))$ *and*
- $v_4 = \min((\mathcal{N}_G^r(v_3) \setminus \mathcal{N}_G^r(v_2)) \cap \mathcal{N}^r(v_0))$ *and*

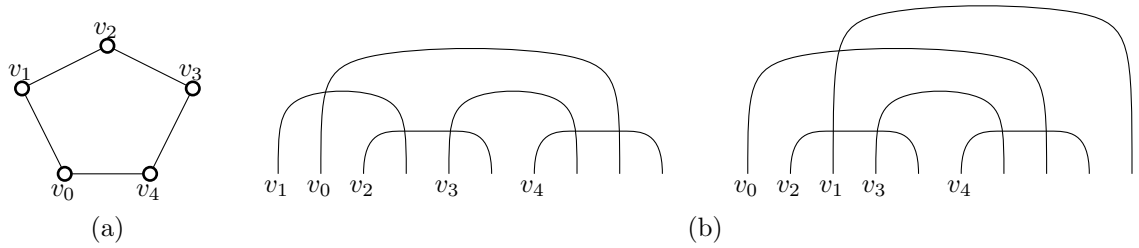


Figure 3.4.: Important subgraph $G_{\pentagon}$. (a) represents the underlying graph, (b) shows the possible circle graph representations

Definition 3.7 (Family of Important Subgraphs $\mathbb{G}_{\bigcirc}$)**.** *Let G be a circle graph and let G' with $V(G') = \{v_0, \ldots, v_n\}$ be a subgraph of G with $n \geq 5$ where v_i crosses $v_{(i+1) \bmod n+1}$ and does not cross $v_{i+2 \bmod n+1}$ for all $i \in \{0, \ldots, n\}$. Then G' belongs to the family of important subgraphs $\mathbb{G}_{\bigcirc}$ if it is isomorphic to a chordless cycle of length n and if either*

- v_0 *directly encases* v_i *with* $2 \leq i \leq n-1$ *and*
- $\mathcal{N}_G(v_i) \cup \mathcal{N}_G(v_0) = \emptyset$ *for* $3 \leq i \leq n-2$ *and*
- $v_i = \min(\mathcal{N}_G(v_{i-1}) \cap \mathcal{N}_G(v_{i+1}))$ *for* $2 \leq i \leq n-1$ *and*
- $v_1 = \max((\mathcal{N}^{\ell}_G(v_2) \setminus \mathcal{N}^{\ell}_G(v_1)) \cap \mathcal{N}^{\ell}_G(v_0))$ *and*
- $v_n = \min((\mathcal{N}^{r}_G(v_{n-1}) \setminus \mathcal{N}^{r}_G(v_{n-2})) \cap \mathcal{N}^{r}_G(v_0))$

or

- v_0 *directly encases* v_i *with* $2 \leq i \leq n-1$ *and*
- v_1 *directly encases* v_i *with* $3 \leq i \leq n$ *and*
- $v_i = \min(\mathcal{N}_G(v_{i-1}) \cap \mathcal{N}_G(v_{i+1}))$ *for* $2 \leq i \leq n$ *and*
- $v_1 = \max((\mathcal{N}^{r}_G(v_2) \setminus \mathcal{N}^{r}_G(v_3)) \cap \mathcal{N}^{r}_G(v_0))$ *and*
- $v_n = \min((\mathcal{N}^{r}_G(v_{n-1}) \setminus \mathcal{N}^{r}_G(v_{n-2})) \cap \mathcal{N}^{r}_G(v_0))$ *and*
- $\mathcal{N}_G(v_i) \cup \mathcal{N}_G(v_0) = \emptyset$ *for* $4 \leq i \leq n-2$ *and*
- $\mathcal{N}_G(v_i) \cup \mathcal{N}_G(v_1) = \emptyset$ *for* $4 \leq i \leq n-1$

We first show that if any only if c^* has certain properties, it is realizable for important subgraphs.

Lemma 3.8 (Properties of c^* for $G_{\triangleleft}$, $G_{\square}$, $G_{\boxslash}$ and $G_{\pentagon}$)**.** *Let G be a circle graph and c^* an auxiliary coloring function for G.*

a) Let I be an important subgraph $G_{\triangleleft}$ of G. Then c^* is realizable for I if and only if c^* has the following properties:

(G1.1) $c^*(v_0, v_2) =$ `true` *or* $c^*(v_0, v_3) =$ `true`

(G1.2) $c^*(v_0, v_2) =$ `false` *or* $c^*(v_0, v_3) =$ `false`

b) Let I be an important subgraph $G_{\square}$ of G. Then c^* is realizable for I if and only if c^* has the following property:

(G2) $c^*(v_0, v_2) =$ `true` *or* $c^*(v_1, v_3) =$ `true`

c) Let I be an important subgraph $G_{\boxslash}$ of G. Then c^* is realizable for I if and only if c^* has the following property:

(G3) $c^*(v_0, v_2) =$ `true`

d) Let I be an important subgraph $G_{\pentagon}$ of G with $V(I) = \{v_0, \ldots v_4\}$ Then c^* is realizable for I if and only if c^* has the following properties:

(G4.1) For two pairs of vertices v_i, v_j and $v_i, v_{j'}$ with $0 \leq i \leq 4$ and $j = i+2 \bmod 5$ and $j' = i+3 \bmod 5$, i.e. v_j and $v_{j'}$ are adjacent, $c^*(v_i, v_j) =$ `false` or $c^*(v_i, v_{j'}) =$ `false` must hold

(G4.2) For exactly two distinct pairs of chords v_a, v_b and v_c, v_d, that is $|\{v_a, v_b, v_c, v_d\}| = 4$ and $a, b, c, d \in \{0, \ldots, 4\}$, it must hold that $c^*(c_a, c_b) = c^*(c_c, c_d) =$ `true`

Proof. We prove the statements as follows, using the vertex labeling from Figures 3.1 to 3.4.

a): Assume c^* is realizable and let col be a 3-coloring that realizes c^*. Since v_1, v_2, v_3 form a 3-clique, they must have a different color each in any 3-coloring. It must also hold that $\text{col}(v_0) \neq \text{col}(v_1)$, since v_0 and v_1 are adjacent. Therefore v_1 must have one of the remaining two colors $\{0, 1, 2\} \setminus \text{col}(\{v_1\})$, so either $\text{col}(v_0) = \text{col}(v_2)$ and $\text{col}(v_0) \neq \text{col}(v_3)$, or $\text{col}(v_0) = \text{col}(v_3)$ and $\text{col}(v_0) \neq \text{col}(v_2)$. Hence, c^* must have properties G1.1 and G1.2. Conversely, if c^* has properties G1.1 and G1.2. then any 3-coloring col constructed as described above realizes c^*, hence c^* is realizable. Therefore, c^* is realizable for $G_{\triangleleft}$ if and only if it has properties G1.1 and G1.2.

b): Assume c^* is realizable and let col be a 3-coloring col that realizes c^*. Clearly, for any 3-coloring, $\text{col}(v_0) \neq \text{col}(v_1)$. Then either $\text{col}(v_2) = \text{col}(v_0)$ or $\text{col}(v_2) = \{0, 1, 2\} \setminus \text{col}(\{v_0, v_1\})$ in which case $\text{col}(v_3) = \text{col}(v_1)$. From this we see that c^* must have property G2. Conversely, if c^* has property G2, then any 3-coloring col constructed as described above realizes c^*, hence c^* is realizable. Therefore, c^* is realizable for $G_{\square}$ if and only if it has property G2.

c): Assume c^* is realizable and let col be a 3-coloring col that realizes c^*. The vertices $\{v_0, v_1, v_2\}$ form a 3-clique, so for any 3-coloring it holds w.l.o.g. that $\text{col}(v_0) = 0, \text{col}(v_1) = 1$ and $\text{col}(v_2) = 3$. Then the only color v_3 can have is $\text{col}(v_1)$, since v_3 is adjacent to both v_0 and v_2. Hence, c^* must have property G3. Conversely, if c^* has property G3, then the 3-coloring from before realizes c^*, meaning that c^* is realizable. Therefore, c^* is realizable for $G_{\boxslash}$ if and only if it has property G3.

d): Assume c^* is realizable and let col be a 3-coloring for G with $\text{col}(v_i) = \text{col}(v_j)$ and $\text{col}(v_i) = \text{col}(v_{j'})$ for $0 \leq i \leq 4$ and $j = i + 2 \bmod 5$ and $j' = i + 3 \bmod 5$. This is not a 3-coloring since $\{v_j, v_{j'}\} \in E(G)$. Hence, either $\text{col}(v_i) = \text{col}(v_j)$ and $\text{col}(v_i) \neq \text{col}(v_{j'})$, $\text{col}(v_i) = \text{col}(v_{j'})$ and $\text{col}(v_i) \neq \text{col}(v_j)$ or $\text{col}(v_j) \neq \text{col}(v_i) \neq \text{col}(v_{j'})$. It follows that c^* must have property G4.1. Since G is a cycle of length 5 it must be colored with at least 3 colors and therefore w.l.o.g. the chord $\text{col}(v_0) = 0$ and the remaining chords are colored as follows: $\text{col}(v_1) = 1, \text{col}(v_2) = 2, \text{col}(v_3) = 1$ and $\text{col}(v_4) = 2$. This means that exactly $c^*(v_1, v_3) = c^*(v_2, v_4) =$ `true` must hold and c^* must have property G4.2. Conversely, if c^* has properties G4.1 and G4.2, w.l.o.g. c^* is defined as follows: $c^*(v_0, v_2) =$ `true`, $c^*(v_1, v3) =$ `true`, $c^*(v_2, v_4) =$ `false`, $c^*(v_3, v_0) =$ `false` and $c^*(v_4, v_1) =$ `false`. We see that w.l.o.g. col can be derived as follows: $\text{col}(v_0) = \text{col}(v_2) = 0$, $\text{col}(v_1) = \text{col}(v_3) = 2$ and $\text{col}(v_4) = 2$. Therefore, if c^* has properties G4.1 and G4.2 there exists a 3-coloring realizing it. In conclusion, c^* is realizable for $G_{\pentagon}$ if and only if it has properties G4.1 and G4.2. □

Next, we prove that if and only if certain properties hold for c^*, it is realizable for important subgraphs of $\mathbb{G}_{\bigcirc}$. Recall that an important subgraph $G_{\bigcirc}$ of $\mathbb{G}_{\bigcirc}$ is isomorphic to a chordless cycle of length $l \geq 6$. Since c^* is defined only on vertices that share a common neighbor, we introduce a recursive function $h : V(G)^2 \to \mathbb{B}$ to describe the relationship between the first two vertices of the cycle v_0 and v_1 and every other vertex v_i of $G_{\bigcirc}$ with $2 \leq i \leq l$ with regard to having the same color in a 3-coloring. Namely, if c^* is realizable and $c^*(v_{i-2}, v_i) =$ `true`, then clearly v_0 and v_i have the same color if and only if v_0 and v_{i-2} have the same color. Hence, we set $h(v_0, v_i) \coloneqq h(v_0, v_{i-2})$. On the other hand, if $c^*(v_{i-2}, v_i) =$ `false`, then

v_0 and v_i have the same color if and only if that color is not used by either v_{i-1} or v_{i-2}. Hence, $h(v_0, v_i) \coloneqq \neg h(v_0, v_{i-2}) \wedge \neg h(v_0, v_{i-1})$. By the same arguments this also holds for v_1 and each v_i. Finally, after applying h to all vertices of the cycle, we want the first and last vertex of the cycle to have different colors, i.e. $h(v_0, v_l) = \texttt{false}$ and that the values of $h(v_0, v_{l-1})$ and $h(v_1, v_l)$ are equal to $c^*(v_0, v_{l-1})$ and $c^*(v_1, v_l)$ respectively. This implies that we were able to color the cycle while traversing it such that the coloring realizes c^*. With the help of this recursive function h we prove the following lemma.

Lemma 3.9. *Let G be a circle graph and I an important subgraph $G_\bigcirc \in \mathbb{G}_\bigcirc$ of G with $V(I) = \{v_0, \dots, v_l\}$ and $l \leq 5$ and let c^* be an auxiliary coloring function for G. We define a recursive function $h : V(I)^2 \to \mathbb{B}$ with $x \in \{0, 1\}$ and $x + 3 \leq i \leq l$ as follows:*

$$
\begin{aligned}
&h(v_x, v_{x+1}) \coloneqq \texttt{false} && && \text{and}\\
&h(v_x, v_{x+2}) \coloneqq c^*(v_x, v_{x+2}) && && \text{and}\\
&h(v_x, v_i) = \begin{cases} h(v_x, v_{i-2}) & \text{if } c^*(v_{i-2}, v_i) = \texttt{true} \\ \neg h(v_x, v_{i-2}) \wedge \neg h(v_x, v_{i-1}) & \text{if } c^*(v_{i-2}, v_i) = \texttt{false} \end{cases}
\end{aligned}
$$

Then c^* is realizable for I if and only if the following statements hold:

- $h(v_0, v_l) = \texttt{false}$ *and*
- $h(v_0, v_{l-1}) = c^*(v_0, v_{l-1})$ *and*
- $h(v_1, v_l) = c^*(v_1, v_l)$

Proof. Assume c^* is realizable on I and let col be a 3-coloring that realizes c^*. As an intermediate step, we show by induction on i that if c^* is realizable on the first i vertices of I, then $h(v_0, v_i) = \texttt{true}$ if and only if $\text{col}(v_0) = \text{col}(v_i)$ where $1 \leq i \leq l$. For $i \leq 2$ it holds by construction of h that $h(v_0, v_1) = \texttt{false}$ and $h(v_0, v_2) = c^*(v_0, v_2)$, so the intermediate statement follows directly. Now assume that $h(v_0, v_i) = \texttt{true}$ if and only if $\text{col}(v_0) = \text{col}(v_i)$ for $2 \leq i < l$. We show that the intermediate statement holds for $i + 1$. Assume c^* is realizable on $\{v_0, \dots, v_{i+1}\}$ and let col be a 3-coloring that realizes c^*. If $c^*(v_{i-1}, v_{i+1}) = \texttt{true}$, then $\text{col}(v_{i-1}) = \text{col}(v_{i+1})$. This means that $\text{col}(v_0) = \text{col}(v_{i+1})$ if and only if $\text{col}(v_0) = \text{col}(v_{i-1})$. Then $h(v_0, v_{i+1}) = \texttt{true}$ if and only if $h(v_0, v_{i-1}) = \texttt{true}$ which proves the intermediate statement for this case. If $c^*(v_{i-1}, v_{i+1}) = \texttt{false}$, then $\text{col}(v_{i+1})$ must be $\{0, 1, 2\} \setminus \text{col}(v_{i-1}, v_i)$. Therefore, $\text{col}(v_0) = \text{col}(v_{i+1})$ holds if and only if $\text{col}(v_0) \neq \text{col}(v_{i-1})$ and $\text{col}(v_0) \neq \text{col}(v_i)$. Hence, $h(v_0, v_{i+1}) = \texttt{true}$ if and only if $h(v_0, v_{i-1}) = \texttt{false}$ and $h(v_0, v_i) = \texttt{false}$. This concludes the proof for the intermediate statement. The same statement can be proven analogously for $h(v_1, v_i)$ with $3 \leq i \leq l$. Since v_0 and v_l are adjacent, $\text{col}(v_0) \neq \text{col}(v_l)$, hence from the intermediate statement it follows that $h(v_0, v_l) = \texttt{false}$. With this we get that if c^* is realizable, then $h(v_0, v_l) = \texttt{false}$, $h(v_0, v_{l-1}) = c^*(v_0, v_{l-1})$ and $h(v_1, v_l) = c^*(v_1, v_l)$.

Now assume $h(v_0, v_l) = \texttt{false}$, $h(v_0, v_{l-1}) = c^*(v_0, v_{l-1})$ and $h(v_1, v_l) = c^*(v_1, v_l)$. Using induction over i with $1 \leq i \leq l$ we show that if h is defined as above then there exists a 3-coloring col realizing c^* for vertices $\{v_0, \dots, v_i\}$ of I such that $h(v_0, v_i) = \texttt{true}$ if and only if $\text{col}(v_0) = \text{col}(v_i)$. For $i \leq 2$ we have that $h(v_0, v_1) = \texttt{false}$ and $h(v_0, v_2) = c^*(v_0, v_2)$. Clearly, in any 3-coloring, v_0 and v_1 must have different colors, so $\text{col}(v_0) \neq \text{col}(v_1)$. W.l.o.g. we pick the colors 0 and 1 for v_0 and v_1 respectively. For v_2 we then either pick 0 or 2, depending on c^*. This proves the base case. Now assume that there exists a 3-coloring that realizes c^* for $\{v_0, \dots, v_i\}$ and $h(v_0, v_i) = \texttt{true}$ if and only if $\text{col}(v_0) = \text{col}(v_i)$. We show that c^* is also realizable for $i + 1$ and that $h(v_0, v_{i+1}) = \texttt{true}$ if and only if $\text{col}(v_0) = \text{col}(v_{i+1})$. For this we consider the two cases for $c^*(v_{i-1}, v_{i+1})$. First assume $c^*(v_{i-1}, v_{i+1}) = \texttt{true}$. By definition of h we get $h(v_0, v_{i+1}) = h(v_0, v_{i-1})$. By induction hypothesis there exists a 3-coloring col that realizes c^* for $\{v_0, \dots, v_i\}$. with $h(v_0, v_{i-1}) = \texttt{true}$ if and only if

$\text{col}(v_0) = \text{col}(v_{i-1})$. Clearly, $\text{col}(v_{i-1}) \neq \text{col}(v_i)$ must hold, therefore $\text{col}(v_{i+1}) := \text{col}(v_{i-1})$ is a valid extension for col, realizing c^*. This also means that $\text{col}(v_0) = \text{col}(v_{i+1})$ if and only if $\text{col}(v_0) = \text{col}(v_{i-1})$. Hence, $h(v_0, v_{i+1}) = h(v_0, v_{i-1}) = \texttt{true}$ if and only if $\text{col}(v_0) = \text{col}(v_{i+1})$. Now assume $c^*(v_{i-1}, v_{i+1}) = \texttt{false}$. Then $h(v_0, v_{i+1})$ is $\texttt{true}$ if and only if $h(v_0, v_{i-1}) = h(v_0, v_i) = \texttt{false}$. By induction hypothesis, there exists a 3-coloring col that realizes c^* for $\{v_0, \ldots, v_i\}$ and $h(v_0, v_{i-1}) = \texttt{true}$ if and only if $\text{col}(v_0) = \text{col}(v_{i-1})$. We want to extend col to v_{i+1} such that col realizes c^*, i.e. v_{i-1} and v_{i+1} have different colors. The obvious choice for v_{i+1} is the color $\{0, 1, 2\} \setminus \text{col}(\{v_{i-1}, v_i\})$. This color is the same as $\text{col}(v_0)$ if and only if $\text{col}(v_0) \neq \text{col}(v_{i-1})$ and $\text{col}(v_0) \neq \text{col}(v_i)$. Hence, $h(v_0, v_{i+1}) = \neg h(v_0, v_{i-1}) \wedge \neg h(v_0, v_i) = \texttt{true}$ if and only if $\text{col}(v_0) = \text{col}(v_{i+1})$. This concludes the induction proof for the second intermediate statement, which can be proven analogously for $h(v_1, v_i)$ with $2 \leq i \leq l$. We lastly need to argue that once we're done iteratively coloring the circle such that the coloring realizes c^*, we do not end up with conflicting colors for v_0 and v_l, i.e. that $\text{col}(v_0) \neq \text{col}(v_l)$. Since by assumption it holds that $h(v_0, v_l) = \texttt{false}$ and that by induction there exists a 3-coloring col realizing c^* with $h(v_0, v_i) = \texttt{true}$ if and only if $\text{col}(v_0) = \text{col}(v_i)$, we have that $\text{col}(v_0) \neq \text{col}(v_l)$. This proves the lemma. □

We have shown that for important subgraphs $G_{\triangleleft}$, $G_{\square}$, $G_{\boxslash}$ and $G_{\pentagon}$ as well as for important subgraphs of $\mathbb{G}_{\bigcirc}$, c^* is realizable if and only if its values for the vertices in $\mathcal{D}$ correspond to 3-colorings on these subgraphs. From this we infer the following definition of a *consistent auxiliary coloring function*.

Definition 3.10 (Consistent Auxiliary Coloring Function)**.** *Let G be a circle graph and c^* an auxiliary coloring function. We say that c^* is* consistent *if and only if it is realizable for each important subgraph $G_{\triangleleft}$, $G_{\square}$, $G_{\boxslash}$ and $G_{\pentagon}$ of G as well as the important subgraphs $G_{\bigcirc} \in \mathbb{G}_{\bigcirc}$ of G.*

We note that these definitions and lemmas are equivalent to the ones presented by Unger in his thesis [51, p. 83ff.].

3.2. 3-Coloring Graphs with c^*

In this section we discuss if a graph G is 3-colorable if and only if there exists a consistent auxiliary coloring function c^*. As argued before, deriving an auxiliary coloring function from an existing 3-coloring can be done fairly easily. What is less obvious is how we can prove the existence of a 3-coloring using such a function. Our first step was to shown that if and only if c^* has certain properties, there exists a 3-coloring for important subgraphs realizing c^*. From this we derived the definition of a consistent auxiliary coloring function c^*. Now we want to construct a 3-coloring for a circle graph G using a consistent auxiliary coloring function c^*. We do this level by level starting with the outermost one, that is $\text{level}_G(1)$, which we color from left to right such that the resulting 3-coloring realizes c^*. The remaining graph is colored by extending the 3-coloring level by level, i.e. from level i to level $i + 1$. During each of these steps, the 3-coloring expands through the circle graph using the important subgraphs as bridges across and within levels.

An additional result is that any 3-coloring col of a circle graph G is unique if it is constructed using a consistent auxiliary coloring function c^* for G as follows. We start by picking some first edge $\{v_0, v_1\} \in E(G)$ and color its endpoints with two different colors. This base case coloring is clearly unique. We then pick some uncolored vertex $v \in V(G)$ that has at least one colored neighbor and assign a color to it as follows. We assume that vertices $\{v_0, \ldots, v_i\}$ have been colored uniquely. Let v_{i+1} be an uncolored vertex that has at least

one colored neighbor v_i. By the induction hypothesis and due to the connectivity of G, v_i has another colored neighbor v_j with $j < i$ and $\text{col}(v_i) \neq \text{col}(v_j)$. Note that v_{i+1} can be adjacent to v_i only or to v_j as well. If v_{i+1} is adjacent to both v_i and v_j, v_{i+1} must be colored with the one remaining color $\{0,1,2\} \setminus \text{col}(\{v_i, v_j\})$, which yields a unique coloring. If v_{i+1} is adjacent only to v_i, then $c^*(v_j, v_{i+1})$ decides whether v_j and v_{i+1} have the same color. In this case we either get $\text{col}(v_j) = \text{col}(v_{i+1})$ or, as before, the remaining color $\{0,1,2\} \setminus \text{col}(v_i, v_j)$. In every case, the 3-coloring including $\text{col}(v_{i+1})$ is unique and we get the following lemma.

Lemma 3.11. *Let G be a 3-colorable circle graph and let c^* be a consistent auxiliary function for G. Any 3-coloring that realizes c^* and is constructed as described above is unique.*

Next, we show that if two important subgraphs share exactly two vertices in a circle graph, then the 3-coloring of the union of these two subgraphs realizing c^* is also unique. This lemma is relevant for when we use the important subgraphs as connective elements between and within levels of a circle graph.

Lemma 3.12. *Let G_i and G_j be two important subgraphs in a circle graph G and let c^* be a consistent auxiliary coloring function for G. If there are exactly two adjacent vertices in $V(G_i) \cap V(G_j)$ then there exists a unique 3-coloring for these subgraphs that realizes c^*.*

Proof. By Lemma 3.11, the colorings we can infer from c^* for G_i and G_j by Lemma 3.8 and 3.9 are in fact unique. W.l.o.g. we color G_i first. Then the colors of two adjacent vertices of G_j are already given by the coloring for G_i. Since c^* is consistent on G we know that there also exists a 3-coloring for G_j. Using the two colored vertices in G_j we can then by Lemma 3.11, 3.8 and 3.9 construct the unique coloring for G_j. Hence the coloring for both subgraphs is also a unique 3-coloring realizing c^*. □

We move on to the first step of constructing a 3-coloring for a circle graph G. That is, we color the chords of the first layer $\text{level}_G(1)$ from left to right and show that for each new chord we encounter we can extend our current 3-coloring using c^*.

Lemma 3.13. *Let G be a circle graph for which there exists a consistent auxiliary coloring function c^*. Then there exists a unique 3-coloring for $\text{level}_G(1)$ that realizes c^*.*

Proof. Let G' be the subgraph induced by $\text{level}_G(1) = \{v_1, v_2, \ldots, v_n\}$ with the chords ordered by their left endpoints. Since by assumption G is connected, so is G', therefore $v_i \in \mathcal{N}^\ell_{G'}(v_{i+1})$ for $1 \leq i < n$. We show that if there exists a 3-coloring col for chords v_1 up to v_i that realizes c^*, then col can be extended to v_{i+1} such that it realizes c^*.

We define the set of chords $\mathcal{L}_i$ as follows:

$$\mathcal{L}_i \coloneqq \mathcal{N}^\ell_{G'}(v_i) \cup \mathcal{N}^\ell_{G'}(\mathcal{N}^\ell_{G'}(v_i)) \text{ for all } i \text{ with } 1 \leq i \leq n.$$

Since G' is connected and consists of only one level, i.e. no chord can encase another chord, $\mathcal{N}^\ell_{G'}(v_i) \cup \{v_i\}$ forms a clique. Also, by assumption G' does not contain any 4-cliques and hence, $|\mathcal{N}^\ell_{G'}(v_i)| \leq 2$. We show in the following that $|\mathcal{L}_i| \leq 4$ by considering every possible size of $\mathcal{N}^\ell_{G'}(v_i)$.

If $|\mathcal{N}^\ell_{G'}(v_i)| = 0$ then $i = 0$ and clearly $|\mathcal{L}_i| = 0 \leq 4$.

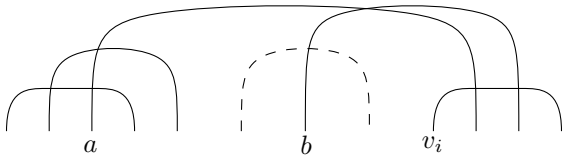


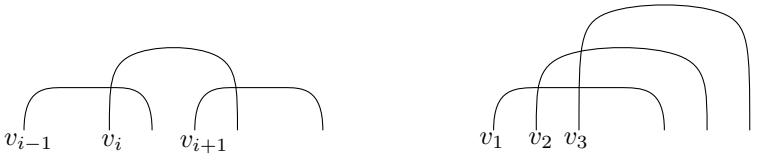
Figure 3.5.: Chords that cross only the left endpoint of b would be encased by a and therefore on a different level

Now assume that $|\mathcal{N}^\ell_{G'}(v_i)| = 1$ with $\mathcal{N}^\ell_{G'}(v_i) = \{a\}$. Similar to before, since G' is connected and only consists of one level, a can have at most 2 other chords crossing its left endpoint, forming a clique.
Hence, $|\mathcal{N}^\ell_{G'}(a)| \leq 2$, and therefore $|\mathcal{L}_i| = |\mathcal{N}^\ell_{G'}(v_i)| + |\mathcal{N}^\ell_{G'}(\mathcal{N}^\ell_{G'}(v_i))| \leq 1 + 2 \leq 4$.

Now assume that $|\mathcal{N}^\ell_{G'}(v_i)| = 2$ with $\mathcal{N}^\ell_{G'}(v_i) = \{a, b\}$ and assume w.l.o..g. that $a^l < b^l$. Again, because G' is connected and consists of only one level, a and b can each have at most 2 neighbors, forming two 3-cliques respectively. We show that all chords in $\mathcal{N}^\ell_{G'}(\mathcal{N}^\ell_{G'}(v_i))$ either cross both a and b simultaneously, or they only cross a. Assume there exists some chord in $\mathcal{N}^\ell_{G'}(\mathcal{N}^\ell_{G'}(v_i))$ crossing only the left endpoint of b and not a. Then a encases any such chord, see Figure 3.5. This is a contradiction, since all chords of G' belong to the same level. Hence, all chords in $\mathcal{N}^\ell_{G'}(\mathcal{N}^\ell_{G'}(v_i))$ either cross both a and b or only a, so $\mathcal{N}^\ell_{G'}(\mathcal{N}^\ell_{G'}(v_i)) = \mathcal{N}^\ell_{G'}(a)$. Since $\mathcal{N}^\ell_{G'}(a) \leq 2$, it follows that $|\mathcal{L}_i| = |\mathcal{N}^\ell_{G'}(v_i)| + |\mathcal{N}^\ell_{G'}(\mathcal{N}^\ell_{G'}(v_i))| \leq 2 + 2 \leq 4$.

We now construct a 3-coloring of G' that realizes c^* For this we consider each possible size of $\mathcal{L}_{i+1}$. We start with the two trivial cases of $|\mathcal{L}_{i+1}| = 0$ and $|\mathcal{L}_{i+1}| = 1$. If $|\mathcal{L}_{i+1}| = 0$, then, since G' is connected, $i = 0$ and $\mathcal{L}_1 = \{v_1\}$, so we set $\text{col}(v_1) \coloneqq 1$. If $|\mathcal{L}_{i+1}| = 1$, then $i = 1$ and $\mathcal{L}_2 = \{v_1, v_2\}$ where v_1 has color 1. Hence, we set $\text{col}(c_2) \coloneqq 2$. The coloring so far is not relevant to c^*, since c^* is only defined on chords that are not adjacent. Therefore, this coloring trivially realizes c^*. From now on we assume that there exists a 3-coloring col for $\{v_1, \ldots, v_i\}$ that realizes c^* and show that col can be extended to v_{i+1} such that it still realizes c^*.

Assume $|\mathcal{L}_{i+1}| = 2$. We can distinguish two subcases, namely $|\mathcal{N}^\ell_{G'}(v_{i+1})| = 1$ and $|\mathcal{N}^\ell_{G'}(v_{i+1})| = 2$. If $|\mathcal{N}^\ell_{G'}(v_{i+1})| = 1$, then $|\mathcal{N}^\ell_{G'}(\mathcal{N}^\ell_{G'}(v_{i+1}))| = 1$, see Figure 3.6(a). We set $\text{col}(v_{i+1})$ according to $c^*(v_{i-1}, v_{i+1})$, which means v_{i+1} either has the same color as v_{i-1} or the color $\{0, 1, 2\} \setminus \text{col}(\{v_{i-1}, v_i\})$. In both cases, the resulting 3-coloring is correct and realizes c^*.



(a) (b)

Figure 3.6.: Illustration of Case $|\mathcal{L}_{i+1}| = 2$ in Lemma 3.13

If $|\mathcal{N}^\ell_{G'}(v_{i+1})| = 2$, then $i = 2$ and $\mathcal{N}^\ell_{G'}(v_{i+1}) = \mathcal{N}^\ell_{G'}(v_3) = \{v_1, v_2\}$, see Figure 3.6(b). By Definition 3.1, c^* is not defined for this 3-clique and therefore v_3 gets the remaining color $\{0, 1, 2\} \setminus \text{col}(\{v_1, v_2\}) = 0$. This coloring trivially realizes c^*.

Now consider the case $|\mathcal{L}_{i+1}| = 3$. For this, we again consider the following subcases: If $|\mathcal{N}^\ell_{G'}(v_{i+1})| = |\{v_i\}| = 1$, then $|\mathcal{N}^\ell_{G'}(\mathcal{N}^\ell_{G'}(v_{i+1}))| = 2$ and $\mathcal{L}_{i+1}$ is the important subgraph $G_{\dashv}$, see Figure 3.7(a) and 3.1(a) and (b). $|\mathcal{N}^\ell_{G'}(v_{i+1})| = 1$ or $|\mathcal{N}^\ell_{G'}(v_{i+1})| = 2$. From Lemma 3.8(a) we get a 3-coloring that realizes c^* by setting either $\text{col}(v_{i+1}) \coloneqq \text{col}(v_{i-2})$ or $\text{col}(v_{i+1}) \coloneqq$

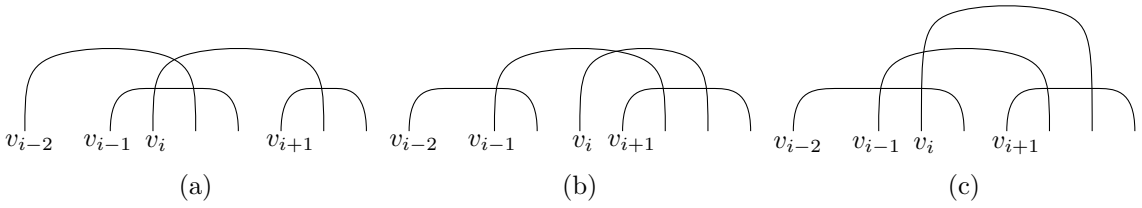


Figure 3.7.: Illustration of Case $|\mathcal{L}_{i+1}| = 0$ in Lemma 3.13

$\text{col}(v_{i-1})$, depending on c^*. Hence, $\text{col}(v_{i+1})$ is given by c^* and the resulting 3-coloring realizes c^*.

For $|\mathcal{N}^{\ell}_{G'}(v_{i+1})| = |\{v_{i-1}, v_i\}| = 2$ we consider the two possible positions of v_{i-2}. Either v_{i-2} crosses only v_{i-1}, see Figure 3.7(b) or it crosses both v_{i-1} and v_i, see Figure 3.7(c). In the first case, $\mathcal{L}_{i+1}$ is the important subgraph $G_{\dashv}$, see Figure 3.1(a) and (b). From Lemma 3.8 we get that either $\text{col}(v_{i+1}) \coloneqq \text{col}(v_{i-2})$ if $c^*(v_{i-2}, v_{i+1}) = \texttt{true}$ or $\{0,1,2\} \setminus \text{col}(\{v_{i-1}, v_i)\}$ otherwise, which are both correct color assignments for v_{i+1} that realize c^*.

In the second case, where v_{i-2} crosses both v_{i-1} and v_i, see Figure 3.7(c), $\mathcal{L}_{i+1}$ is the important subgraph $G_{\boxslash}$, see Figure 3.3. By Lemma 3.8(c), we have that $c^*(v_{i-2}, v_{i+1}) = \texttt{true}$, hence $\text{col}(v_{i+1}) \coloneqq \text{col}(v_{i-2})$, which gives us a 3-coloring that realizes c^*.

Lastly, we consider the case $|\mathcal{L}_{i+1}| = 4$. Since G' does not contain any cliques larger than 3, so the chords in $\mathcal{L}_{i+1}$ must be distributed as follows: $\mathcal{N}^{\ell}_{G'}(v_{i+1}) = \{v_{i-1}, v_i\}$ and $\mathcal{N}^{\ell}_{G'}(\mathcal{N}^{\ell}_{G'}(v_{i+1})) = \{v_{i-2}, v_{i-3}\}$. We consider the possible positions for v_{i-2} and v_{i-3}. Because no chord can be encased by another chord, v_{i-2} and v_{i-3} must cross each other. They also must cross v_{i-1}. If v_{i-2} or v_{i-3} only cross v_i, then they would be encased by v_i - a contradiction. The only way a chord in $\mathcal{N}^{\ell}_{G'}(\mathcal{N}^{\ell}_{G'}(v_{i+1}))$ can cross both v_{i-1} and v_i is if it does not encase the other chord in $\mathcal{N}^{\ell}_{G'}(\mathcal{N}^{\ell}_{G'}(v_{i+1}))$. This yields the following two possible cases. In the first case, illustrated in Figure 3.8(a), $\mathcal{L}_{i+1}$ forms four important subgraphs $G_{\dashv}$, also illustrated in Figure 3.1(c). From Lemma 3.8(a) we get that the color for v_{i+1}

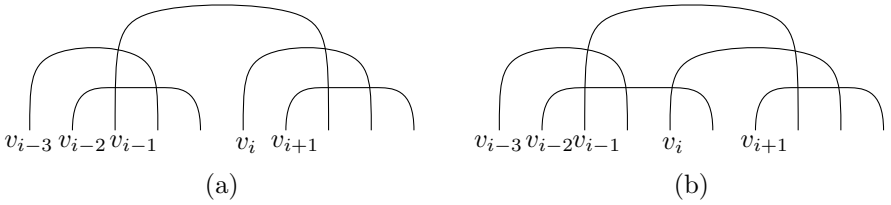


Figure 3.8.: Illustration of Case $|\mathcal{L}_{i+1}| = 4$ in Lemma 3.13

is determined by $c^*(v_{i-3}, v_{i+1})$ and $c^*(v_{i-2}, v_{i+1})$. Assume $c^*(v_{i-3}, v_{i+1}) = \texttt{true}$. Then $c^*(v_{i-2}, v_{i+1}) = \texttt{false}$ and $c^*(v_{i-2}, v_i) = \texttt{true}$. To realize c^*, we set $\text{col}(v_{i+1}) \coloneqq \text{col}(v_{i-3})$. Since $c^*(v_{i-2}, v_{i+1}) = \texttt{false}$ and $c^*(v_{i-2}, v_i) = \texttt{true}$ and by induction hypothesis we have that $\text{col}(v_{i+1}) \neq \text{col}(v_i)$. The case $c^*(v_{i-2}, v_{i+1}) = \texttt{true}$ works analogously. Both cases give a correct coloring of v_{i+1} that realizes c^*. In the second case, illustrated in Figure 3.8(b), $\{v_{i-2}\}$ is the important subgraph $G_{\boxslash}$, see Figure 3.3. Lemma 3.8(c) then gives us that $\text{col}(v_{i+1}) \coloneqq \text{col}(v_{i-2})$ yields a correct 3-coloring, which also realizes c^*.

From this proof together with Lemma 3.11 we get that if there exists a consistent auxiliary coloring function c^* for G then there exists a unique 3-coloring for $\text{level}_G(1)$ that realizes c^*. □

Next, we show that a 3-coloring of the first l layers of a circle graph G that realizes c^* can be extended to layer $l+1$. We separate the proof for this statement into two lemmas.

First, we show that we can extend the 3-coloring to the chords in layer $l+1$ that have a neighbor in layer l. Then, we prove that the 3-coloring can be extended to the remaining chords in layer $l+1$.

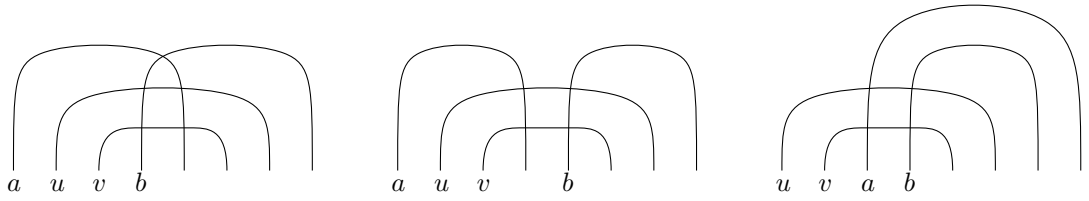
Lemma 3.14. *Let G be a circle graph for which there exists a consistent auxiliary coloring function c^*. Further, let there be a unique 3-coloring for $\bigcup_{i=1}^{l} \text{level}_G(i)$ that realizes c^* with $l < L(G)$. Then this 3-coloring can be extended to chords in $\text{level}_G(l+1)$ that have a neighbor in $\bigcup_{i=1}^{l} \text{level}_G(i)$. This 3-coloring is unique.*

Proof. Let v be a chord in $\text{level}_G(l+1)$ with a neighbor in $\bigcup_{i=1}^{l} \text{level}_G(i)$. Then there exists a chord $u \in \text{level}_G(l)$ such that u directly encases v and $v \in \mathcal{N}_G(\mathcal{N}_G(u))$. In particular, there exists some chord w in the set of chords $W_v \coloneqq \bigcup_{i=1}^{l} \text{level}_G(i) \cap \mathcal{N}_G(v) \cap \mathcal{N}_G(u)$ that connects v to the previous layer. We show that for each such v, u and W_v the following coloring method is correct:

$$\text{col}(v) \coloneqq \begin{cases} \text{col}(u), & \text{if } c^*(v,u) = \texttt{true} \\ \{0,1,2\} \setminus (\text{col}(W_v) \cup \text{col}(\{u\})), & \text{if } c^*(v,u) = \texttt{false} \end{cases}$$

Assume $c^*(v,u) = \texttt{true}$ and we set $\text{col}(v) \coloneqq \text{col}(u)$. Since $W_v \subseteq \bigcup_{i=1}^{l} \text{level}_G(i) \cap \mathcal{N}_G(u)$ and by assumption the 3-coloring for all chords in $\bigcup_{i=1}^{l} \text{level}_G(i)$ is correct, $\text{col}(u) \neq \text{col}(w)$ for all $w \in W_v$. Therefore, coloring v with the same color as u yields a correct 3-coloring.

Now assume $c^*(v,u) = \texttt{false}$. We show that $\text{col}(v) \coloneqq \{0,1,2\} \setminus (\text{col}(W_v) \cup \text{col}(\{u\}))$ is also a correct 3-coloring. For this, we first prove that $|\text{col}(W_v) \cup \{\text{col}(u)\}| = 2$ by contradiction. We assume that u directly encases v, $c^*(u,v) = \texttt{false}$ and $|\text{col}(W_v) \cup \{\text{col}(u)\}| \neq 2$. Because $W_v \subseteq \bigcup_{i=1}^{l} \text{level}_G(i) \cap \mathcal{N}_G(u)$ and we assume a 3-coloring for $\bigcup_{i=1}^{l} \text{level}_G(i)$, all chords in W_v must be colored differently from u and therefore only $|\text{col}(W_v) \cup \{\text{col}(u)\}| = 3$ can hold. If so, there exist two chords $a, b \in W_v$ with $\text{col}(a) \neq \text{col}(b)$. If a and b cross, see Figure 3.9(a), it would form the important subgraph $G_{\boxslash}$, see Figure 3.3 and by Lemma 3.8(c) $c^*(v,u) = \texttt{true}$ must hold, a contradiction. If a and b don't cross, then their two possible positions are



(a) (b) (c)

Figure 3.9.: Possible positions of chords $a, b \in W_v$

illustrated in Figure 3.9(b) and (c), that is, w.l.o.g. either $a^r < b^\ell$ or a encases b. In both cases $\{a, b, v, u\}$ form the important subgraph $G_{\square}$ see Figure 3.2. By Lemma 3.8b) and our assumption that the c^* is realizable for $\bigcup_{i=1}^{l} \text{level}_G(i)$, either $c^*(v,u) = \texttt{true}$, which is a contradiction, or $c^*(a,b) = \texttt{true}$, which would mean $|\text{col}(W_v) \cup \{\text{col}(u)\}| = 2$, also a contradiction. Since for $|\text{col}(W_v) \cup \{\text{col}(u)\}| \neq 2$ all cases led to contradictions, we can conclude that $|\text{col}(W_v)| = 1$ and therefore $|\text{col}(W_v) \cup \{\text{col}(u)\}| = 2$.

We now show that we don't create a conflicting coloring for two crossing chords v, v' with $v \in \text{level}_G(l+1)$ and $v' \in \bigcup_{i=1}^{l+1} \text{level}_G(i)$, i.e. that all such crossing chords v, v' have different colors, after applying our coloring strategy to all chords in $\text{level}_G(l+1)$ that have a neighbor in $\bigcup_{i=1}^{l} \text{level}_G(i)$. For this, we distinguish two cases for v', that is $v' \in \bigcup_{i=1}^{l} \text{level}_G(i)$ and $v' \in \text{level}_G(l+1)$. Assume $v' \in \bigcup_{i=1}^{l} \text{level}_G(i)$ and $\text{col}(v) = \text{col}(v')$.

Since v' crosses v but $\text{pos}(v') < \text{pos}(v)$, v' must cross any chord that directly encases v. Otherwise, v' would also be encased by that chord and $\text{pos}(v') = \text{pos}(v)$. In particular, v' crosses u and therefore $v' \in W_v$. By construction, $\text{col}(v) \coloneqq \text{col}(u)$ if $c^*(v,u) = \texttt{true}$ and $\text{col}(v) \coloneqq \{0,1,2\} \setminus (\text{col}(W_v) \cup \{\text{col}(u)\})$ otherwise. If $c^*(v,u) = \texttt{true}$, this would imply $\text{col}(u) = \text{col}(v')$, which is a contradiction to the existence of a 3-coloring for $\bigcup_{i=1}^{l} \text{level}_G(i)$. If $c^*(v,u) = \texttt{false}$, v would have been assigned a color different from $\text{col}(v')$, since $v' \in W_v$.

Now, let $v' \in \text{level}_G(l+1)$ and assume $\text{col}(v) = \text{col}(v')$. Since v and v' belong to the same level, u either directly encases both v and v' or there exist two crossing chords $u, u' \in \text{level}_G(l)$ directly encasing v and v' respectively. We consider these two cases in the following. Assume u directly encases both v and v'. Then $u \in \mathcal{N}_G(\mathcal{N}_G(v)) \cap \mathcal{N}_G(\mathcal{N}_G(v'))$. Therefore, there either exists a chord a that crosses both v and v' as well as u, see Figure 3.10(a), or there exist two chords a, b crossing v and v' respectively, as illustrated in Figure 3.10(b).

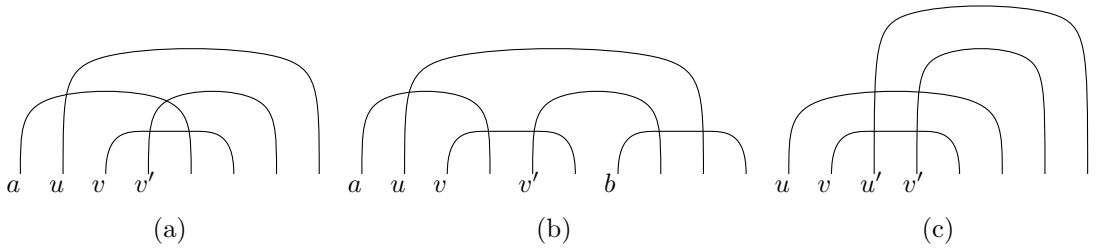


Figure 3.10.: Possible positions for chords in W_v and $W_{v'}$ relative to x and x'

First, we consider the case illustrated in Figure 3.10(a), where a single chord a crosses both x and x'. In this case, $\{v, v', u, a\}$ form the important subgraph $G_{\triangleleft}$ and by Lemma 3.8(a) we have that $c^*(v,u) = \texttt{true}$ if and only if $c^*(v',u) = \texttt{false}$ and analogously, $c^*(v,u) = \texttt{false}$ if and only if $c^*(v',u) = \texttt{true}$. We see that if $\text{col}(v) \coloneqq \text{col}(u)$, we set $\text{col}(v') \coloneqq \{0,1,2\} \setminus (\text{col}(W_{v'}) \cup \{\text{col}(u)\})$ and vice versa. Since $u \in W_v$, resp. $W_{v'}$, this means that v and v' are assigned different colors.

Now assume there exist two chords a, b as in Figure 3.10(a). Then $\{a, b, u, v, v'\}$ form the important subgraph $G_{\pentagon}$, see Figure 3.4. Since $\text{col}(v) = \text{col}(v')$, this means that either $c^*(v,u) = c^*(v',u) = \texttt{true}$ or that $\text{col}(W_v) = \text{col}(W_{v'})$. From Lemma 3.8(d), it follows that $c^*(v,u) = c^*(v',u) = \texttt{true}$ is a contradiction to c^* being consistent. Further, if $\text{col}(W_v) = \text{col}(W_{v'})$, then $\text{col}(a) = \text{col}(b)$. Then the values for c^* are as follows: $c^*(a,b) = \texttt{true}$ and $c^*(v,u) = c^*(v',u) = c^*(a,v') = c^*(b,v) = \texttt{false}$. This is also a contradiction to Lemma 3.8(d), since G4.2 states that two values of c^* that are defined for $G_{\pentagon}$ must be $\texttt{true}$. Hence, if $\text{col}(v) = \text{col}(v')$, then this is a contradiction to our assumption that c^* is consistent. Now assume there exist two chords u, u' such that u directly encases v and u' directly encases v' as illustrated in Figure 3.10(c). Similarly to before, $u \in \mathcal{N}_G(\mathcal{N}_G(v))$ and $u' \in \mathcal{N}_G(\mathcal{N}_G(v'))$. Since v crosses v' and u does not encase v', u crosses v'. By the same argument u' crosses v, see Figure 3.10(c). The chords $\{u, v, u', v'\}$ then form the important subgraph $G_{\square}$, see Figure 3.2. By Lemma 3.8(b), we know that $c^*(v,u) = \texttt{true}$ or $c^*(v',u') = \texttt{true}$. Also, since we assume that there exists a 3-coloring for $\bigcup_{i=1}^{l} \text{level}_G(i)$, we have $\text{col}(u) \neq \text{col}(u')$. If $c^*(v,u) = c^*(v',u') = \texttt{true}$, then $\text{col}(v) \coloneqq \text{col}(u)$ and $\text{col}(v') \coloneqq \text{col}(u')$ and therefore $\text{col}(v) \neq \text{col}(u')$. If $c^*(v,u) = \texttt{true}$ and $c^*(v',u') = \texttt{false}$, then $\text{col}(v) \coloneqq \text{col}(u)$ and $\text{col}(v') \coloneqq \{0,1,2\} \setminus (\text{col}(W_{v'}) \cup \{\text{col}(u')\})$. Since $u \in \mathcal{N}_G(u')$, $\text{col}(u) \in \text{col}(W_{v'})$ and therefore $\text{col}(v) \neq \text{col}(v')$. The case $c^*(v,u) = \texttt{true}$ and $c^*(v',u') = \texttt{false}$ works analogously. Therefore, v and v' are assigned different colors by our coloring method, a contradiction.

In conclusion, the proposed method to extend the 3-coloring to $\text{level}_G(l+1)$ is correct and by Lemma 3.11 also unique. □

3.3. The Limits of c^* - a Counterexample

So far we were able to confirm Unger's results regarding the construction of a 3-coloring using a consistent auxiliary coloring function c^*. The next step of constructing the 3-coloring would be to show the following claim.

Disputable Claim 3.15. *[51, p. 97] Let G be a circle graph for which there exists a consistent auxiliary coloring function c^*. Further, let there be a 3-coloring that realizes c^* for $\bigcup_{i=1}^{l} \text{level}_G(i)$ as well as for the chords in $\text{level}_G(l+1)$ that have a neighbor in $\bigcup_{i=1}^{l} \text{level}_G(i)$ with $l < L(G)$. Then this 3-coloring can be extended to all uncolored chords in $\text{level}_G(l+1)$. The 3-coloring is unique.*

In other words, we want to show that the coloring of some level $l+1 \leq L(G)$, which we get from Lemma 3.14, can be extended to the remaining chords in level $l+1$, i.e. those that do not have a neighbor in level l, given a consistent auxiliary coloring function c^*. Note that in Unger's thesis this claim is part of a proof for a more general lemma [51, p. 97]. In the following we give a high level description of the part of the proof that is relevant for Claim 3.15.

Proof Sketch [51]. From Lemma 3.14 we get a 3-coloring that realizes c^* for the chords $v \in \text{level}_G(l+1)$ for which there exists a chord $y \in \text{level}_G(l)$ such that u directly encases v and $v \in \mathcal{N}_G(\mathcal{N}_G(u))$. Hence, the chords that still have to be colored are those, that lie in $\text{level}_G(l+1)$, but don't have a neighbor in $\bigcup_{i=1}^{l} \text{level}_G(i)$. We first consider all chords that are part of cycles that also contain a chord of level l. For chords that are part of a $G_{\pentagon}$ or $G_{\bigcirc}$ we get a unique and correct 3-coloring by Lemma 3.8(d) and 3.9. The remaining chords, i.e. chords that cross those that are part of the aforementioned important subgraphs, can then supposedly be colored trivially or using Lemma 3.12. Namely, they either form a 3-clique or a $G_{\boxslash}$, see Figure 3.11. The chords in level $l+1$, that are not part of any cycle

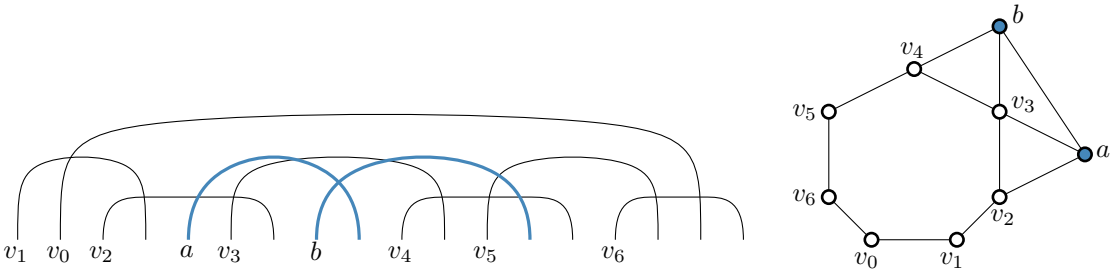


Figure 3.11.: Chords a and b form several $G_{\boxslash}$ with the chords of $G_{\bigcirc}$

are then colored using the procedure in Lemma 3.13. Since they are not part of any cycle, nor do they cross chords of one, there exists some chord x that doesn't have a left, resp. a right neighbor, see Figure 3.12 for some examples. Hence, we can choose x as the starting point for the coloring method described in the proof for Lemma 3.13. Note that we might need to permute the colors of these chords once we reach the first chord that is already colored. □

In the following we construct a counterexample in which following the steps described in Lemma 3.13 and 3.14 and Claim 3.15 using a consistent auxiliary coloring function c^* does not result in a 3-coloring. Moreover, it disproves the claim that there always exists a 3-coloring for G if c^* is consistent for G. This counterexample illustrates how coloring chords that are part of a cycle but not of an important subgraph $G_{\pentagon}$ or $G_{\bigcirc}$ such that the coloring realizes c^* might not be possible even if c^* is consistent. Consider the graph G

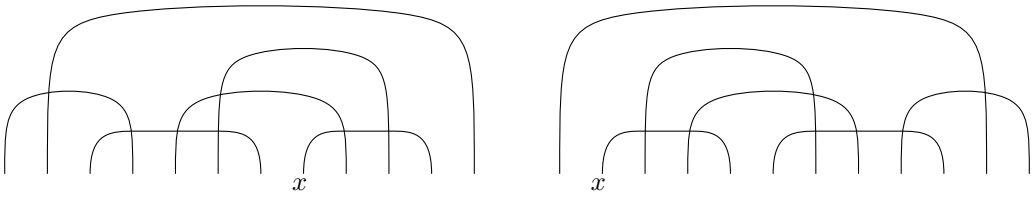


Figure 3.12.: Two examples where the chords in $\text{level}_G(l{+}1)$ are not part of any cycle. Chord x is the starting point for applying the coloring method from Lemma 3.13.

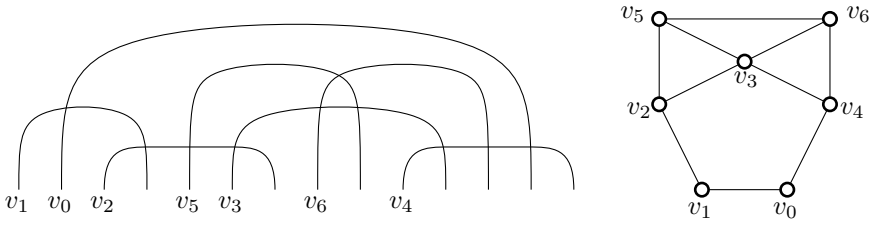


Figure 3.13.: The Graph used in our counterexample

illustrated in Figure 3.13. We first list all important subgraphs of G. The graph contains one $G_{\lhd}$ formed by $\{v_0, v_3, v_4, v_6\}$. Note that by Definition 3.3 the chords $\{v_1, v_2, v_3, v_5\}$ do not form a $G_{\lhd}$ since $\text{pos}(v_1) \neq \text{pos}(v_2)$ and v_1 also does not directly encase the other chords. The same argument holds for $\{v_3, v_4, v_5, v_6\}$, namely $\text{pos}(v_4) \neq \text{pos}(v_3)$ and v_4 does not directly encase v_3, v_5 and v_6. Definition 3.3 does also not apply to $\{v_2, v_3, v_5, v_6\}$ since both v_2 and v_6 cross v_5. We also note that our definition for $G_{\lhd}$ is less strict that Unger's. By the definition given in his thesis, this graph would not contain any $G_{\lhd}$. But, since both in this and Unger's thesis c^* is defined for $c^*(v_0, v_3)$ and $c^*(v_0, v_6)$, the values for c^* we give later on do not contradict any of Unger's original definitions or restrictions.

The graph further contains two $G_{\boxslash}$ formed by $\{v_2, v_3, v_5, v_6\}$ and $\{v_3, v_4, v_5, v_6\}$.

Lastly, G contains the $G_{\pentagon}$ formed by $\{v_0, v_1, v_2, v_3, v_4\}$. Note that G does not contain any $G_{\bigcirc} \in \mathbb{G}_{\bigcirc}$ since $v_3 = \min(\mathcal{N}_G(v_4) \cap \mathcal{N}_G(v_6))$ and $v_2 = \min(\mathcal{N}_G(v_1) \cap \mathcal{N}_G(v_3))$ and therefore $\{v_0, v_1, v_2, v_5, v_6, v_4\}$ do not form a $G_{\bigcirc}$.

Next, we give an auxiliary coloring function c^* and show that it is in fact consistent, i.e. it is realizable for all important subgraphs of G.

From Definition 3.1 we get the following domain for c^*:

$$\mathcal{D} \coloneqq \{(v_0, v_2), (v_0, v_3), (v_0, v_6), (v_1, v_3), (v_1, v_4), (v_1, v_5), (v_2, v_4), (v_2, v_6), (v_4, v_5)\}$$

We further define c^* as follows. To satisfy the constraints given for $G_{\lhd}$ in Lemma 3.8(a) we set $c^*(v_0, v_3) = \texttt{false}$ and $c^*(v_0, v_6) = \texttt{true}$. Further, we set $c^*(v_4, v_5) = \texttt{true}$ and $c^*(v_2, v_6) = \texttt{true}$ to satisfy the constraints for $G_{\boxslash}$ by Lemma 3.8(c). Lastly, we need to set the remaining values for c^* such that it is realizable for $G_{\pentagon}$. Lemma 3.8(d) gives us that $c^*(v_i, v_{(i+2) \bmod 5}) = \texttt{false}$ or $c^*(v_i, v_{(i+3) \bmod 5}) = \texttt{false}$ for $i \in \{0, \ldots, 4\}$ and that there can be only two distinct pairs of vertices for which c^* is $\texttt{true}$. We set the remaining values for c^* as follows: $c^*(v_1, v_3) = \texttt{true}$, $c^*(v_2, v_4) = \texttt{true}$, $c^*(v_0, v_2) = \texttt{false}$ and $c^*(v_1, v_4) = \texttt{false}$. Since for each of the five constraints we always have at least one value of c^* set to $\texttt{false}$ and exactly two values set to $\texttt{true}$, c^* is realizable on $G_{\pentagon}$ and therefore on all important subgraphs of G. Hence, by Definition 3.10, c^* is a consistent auxiliary coloring function.

We now apply the steps described in Lemma 3.13 and 3.14 as well as the proof sketch for Claim 3.15. We start with coloring $\text{level}_G(1)$. Using Lemma 3.13, we get a 3-coloring col

for $\text{level}_G(1)$ where $\text{col}(v_1) = 1$, $\text{col}(v_0) = 2$ and $\text{col}(v_4) = 0$, since $c^*(v_1, v_4) = \texttt{false}$. Next, we color the chords in $\text{level}_G(2)$ that have a neighbor in $\text{level}_G(1)$, namely v_2, v_3 and v_6. From Lemma 3.14 and the respective values of c^*, we get $\text{col}(v_2) = \{0,1,2\} \setminus \text{col}(\{0,1\}) = 2$, since $c^*(v_0, v_2) = \texttt{false}$, $\text{col}(v_3) = \{0,1,2\} \setminus \text{col}(\{1,2\}) = 0$, since $c^*(v_0, v_3) = \texttt{false}$ and $\text{col}(v_6) = \text{col}(v_0) = 1$. The coloring up to this point is illustrated in Figure 3.14. We now

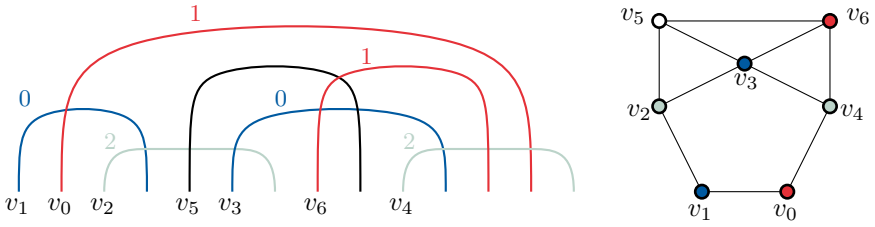


Figure 3.14.: An example of how the graph in the counterexample can be colored

apply the steps for coloring the remaining chords of $\text{level}_G(2)$, in this case v_5. From $c^*(v_4, v_5)$ we get that $\text{col}(v_5) = \text{col}(v_4) = 0$. But then col is no longer a 3-coloring since v_5 crosses v_2 and $\text{col}(v_5) = \text{col}(v_2)$. In fact, $\text{col}(\mathcal{N}_G(v_5)) = \{0,1,2\}$, which means there is no color left for v_5. This counterexample therefore disproves the claim that if given a consistent auxiliary coloring function c^* for a circle graph G, we can construct a 3-coloring col of G that realizes c^* for G by following the steps of Lemma 3.13, 3.14 and Claim 3.15. It also disproves the statement that there always exists a 3-coloring realizing a consistent auxiliary coloring function c^*. Note that we set $c^*(v_0, v_6) = c^*(v_2, v_6) = c^*(v_2, v_4) = \texttt{true}$. No 3-coloring realizes this consistent auxiliary coloring function since it would require v_0, v_4 and v_6 to have the same color. We can further observe, that there exists a contradiction within the values of c^*, this contradiction being that $c^*(v_0, v_6) = \texttt{true}$ and $c^*(v_2, v_6) = \texttt{true}$, but also $c^*(v_0, v_2) = \texttt{false}$. But, as we have seen, neither transitively assigning the same color to crossing chords nor this contradiction does not result in c^* not being consistent. The constraints we defined for c^* only ensured that c^* is realizable on all important subgraphs individually, not on all of them at the same time. Thus, these kinds of problematic values for c^* are possible for pairs of chords that do not appear in the same constraint. Fixing this inconsistency, that is requiring $c^*(v_0, v_2) = \texttt{true}$, would in fact result in a 3-coloring for this example. If we have $c^*(v_0, v_2) = \texttt{true}$, then $\text{col}(v_2)$ is set to $\text{col}(v_0) = 2$. Further, to satisfy the constraints for $G_{\pentagon}$, $c^*(v_0, v_3)$ and $c^*(v_2, v_4)$ must be set to $\texttt{false}$. This means that $\text{col}(v_5) = \text{col}(4) = 0 \neq 2 = \text{col}(v_0) = \text{col}(v_2) = \text{col}(v_6)$. This 3-coloring is illustrated in Figure 3.15. This also shows that the characterization of a consistent auxiliary coloring

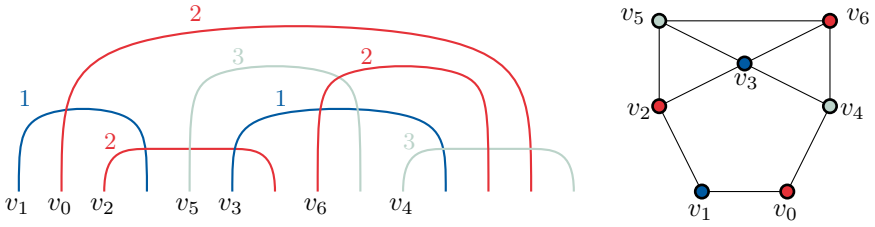


Figure 3.15.: A coloring for the graph from our counterexample. This shows that the characterization of c^* was not disproven.

function c^* is not inherently disproven, that is we have not shown that a circle graph G for which there exists a consistent auxiliary coloring function c^* is not 3-colorable. But the properties of a consistent auxiliary coloring function c^* clearly do not suffice to ensure that it is always realizable for a circle graph G. We leave it as an open problem to specify a strategy to ensure that the properties of c^* do not allow contradicting or un-realizable values.

4. Constructing c^*

In the previous chapter we discussed whether a circle graph G is 3-colorable if and only if there exists a consistent auxiliary coloring function c^*. We have shown that in general, for a circle graph G for which there exists a consistent auxiliary coloring function c^* there does not always exist a 3-coloring of G realizing c^*, even if G is 3-colorable. But although the properties of c^* were not enough to always find a correct 3-coloring we are still interested in finding a consistent auxiliary coloring function c^* and seeing how the coloring algorithms hold up in practice. To this end, we show how to construct a consistent auxiliary coloring function c^* using the properties given in Lemma 3.8 and 3.9. More precisely, we transform these constraints into boolean formulas F_1 and F_2. In F_1, we have the clauses we get from the important subgraphs $G_{\dashv\triangleleft}$, $G_{\square}$, $G_{\boxslash}$ and $G_{\pentagon}$. We show that F_1 is an instance of 2-SAT of polynomial size and solving it therefore takes polynomial time. If there exists no solution for F_1, we know that our graph is not 3-colorable and we stop. If we find a solution for F_1, this solution gives us values for c^* for all important subgraphs except for those that belong to $\mathbb{G}_{\bigcirc}$. Note that we also have to make sure that property G4.2 applies to c^* after finding a solution for F_1, that is, for every $G_{\pentagon}$ exactly two values of c^* for vertices of $G_{\pentagon}$ are `true`. To accomplish these two tasks we take the following two approaches. The first is to search for a valid value assignment for c^* using a backtracking algorithm. The second approach is constructing a second boolean formula F_2 containing more boolean clauses for $G_{\pentagon}$ as well as for all $G_{\bigcirc} \in \mathbb{G}_{\bigcirc}$. This formula F_2 is not an instance of 2-SAT, hence it cannot be solved in polynomial time generally speaking. Similar to before, solving this boolean formula either gives us all values for c^* or tells us that G is not 3-colorable.

4.1. Upper Bounds for the Number of Important Subgraphs

To show that we only need polynomial time to solve F_1 we prove that the number of clauses for a circle graph G is polynomial. To this end, we give upper bounds for the number of each important subgraph in a circle graph G.

Lemma 4.1. *Let G be a circle graph with $|V(G)| = n$. Then G contains $O(n^2)$ important subgraphs $G_{\dashv\triangleleft}$.*

Proof. By Definition 3.3, $G_{\dashv\triangleleft}$ ranges over either one or two levels. Let c be a chord in G such that $(\mathcal{N}_G(c) \cup \{c\}) \cap \text{level}_G(\text{pos}(c))$ forms an important subgraph $G_{\dashv\triangleleft}$. By Definition 3.3 and the fact that G does not contain any cliques larger than 3, chord c can only be part of

one such graph $G_{\prec\triangleleft}$ per level, hence G contains $O(n)$ of these subgraphs. Now assume that for some level $l < L(G)$, $(\mathcal{N}_G(c) \cap (\text{level}_G(l) \cup \text{level}_G(l+1))) \cup \{c\}$ forms the important subgraph $G_{\prec\triangleleft}$. By Definition 3.3 chord $c = \min\left(\bigcap_{v \in H \cup H'} \mathcal{N}_G(v)\right)$, with $H \subseteq \text{level}_G(l)$ and $H' \subseteq \text{level}_G(l+1)$ and H directly encases H', i.e. c is uniquely determined for each pair of subsets H, H'. Therefore c can only be part of one $G_{\prec\triangleleft}$ for any two adjacent levels per endpoint, which are at most $2 \cdot (L(G) - 1)$ per chord, so $O(n^2)$ overall. Hence, G contains $O(n^2)$ important subgraphs $G_{\prec\triangleleft}$. □

To determine the number of important subgraphs $G_{\square}$ and $G_{\boxslash}$ in a circle graph G we introduce sets of chords $U(G)$ and $P(G)$. The set $U(G)$ contains all pairs of chords that directly encase each other, while $P(G)$ contains all pais of chords that are the maximum chord of the left neighborhood and the minimum chord of the right neighborhood of some chord $v \in V(G)$.

Definition 4.2. *Let G be a circle graph. Then we define $U(G)$ and $P(G)$ as follows:*

$$U(G) \coloneqq \{\{a, b\} \colon a, b \in V(G) \text{ and } a \text{ directly encases } b\}$$
$$P(G) \coloneqq \{\{a, b\} \colon \exists c \in V(G) : a = \max(\mathcal{N}_G^{\ell}(c)) \text{ and } b = \min(\mathcal{N}_G^{r}(c))\}$$

We can show that the sizes of these sets are in $O(n)$. This helps us to prove the upper bound on the number of important subgraphs $G_{\square}$ and $G_{\boxslash}$ in a circle graph G.

Lemma 4.3. *Let G be a circle graph with $|V(G)| = n$. Then $|U(G)| \in O(n)$ and $|P(G)| \in O(n)$.*

Proof. Let c be some chord in G. We show that the number of chords that can directly encase c is at most 3. Let E_c be the set of chords that directly encase c. By definition, no chord e in E_c can encase another, since otherwise e would no longer directly encase c. Hence, they form a clique. Since we assumed that G does not contain any cliques of size 4 or larger, $|U_c| \leq 3$. Hence, for any chord in G there can be at most 3 chords directly encasing it and therefore the number of elements in $U(G)$ is at most $3 \cdot n \in O(n)$.

Since for any chord $c \in V(G)$ the chords $a = \max(\mathcal{N}_G^{\ell}(c))$ and $b = \min(\mathcal{N}_G^{r}(c))$ are unique, the number of elements in $P(G)$ is at most $2 \cdot n$ and therefore $|P(G)| \in O(n)$. □

We now prove the upper bound for the number of important subgraphs $G_{\square}$ and $G_{\boxslash}$ in a circle graph G.

Lemma 4.4. *Let G be a circle graph with $|V(G)| = n$. Then G contains $O(n^2)$ important subgraphs $G_{\square}$*

Proof. By Definition 3.4, $\{v_0, v_2\} \in U(G)$. We consider the two cases for the positions of v_1 and v_3. First, assume v_1 directly encases v_3. Then $\{v_1, v_3\} \in U(G)$. With Lemma 4.3 it follows that each of the $O(n)$ possible pairs $\{v_0, v_2\}$ can be combined with $O(n)$ possible pairs (v_1, v_3) to form $G_{\square}$ and therefore there are at most $O(n^2)$ important subgraphs $G_{\square}$ in G. Next, assume that $v_1^r < v_3^l$. By Definition 3.4 $\{v_1, v_3\}$ is uniquely determined for all $O(n)$ possible choices of $\{v_0, v_2\}$ and G contains at most $O(n)$ such important subgraphs $G_{\square}$. □

Lemma 4.5. *Let G be a circle graph with $|V(G)| = n$. Then G contains $O(n^3)$ important subgraphs $G_{\boxslash}$*

Proof. By Definition 3.5 we have to consider two cases for $G_{\boxslash}$. First, we assume that v_1 directly encases v_3. In this case $\{v_1, v_3\} \in U(G)$ and we have at most $O(n)$ choices for v_0 and v_2 resp. This gives us $O(n^3)$ important subgraphs $G_{\boxslash}$. Next, assume $v_1^r < v_3^l$. By Definition 3.5, v_1 and v_3 are uniquely determined by v_0 and v_2. Hence, G contains at most $O(n)$ such important subgraphs $G_{\boxslash}$. □

Lemma 4.6. *Let G be a circle graph with $|V(G)| = n$. Then G contains $O(n)$ important subgraphs $G_{\pentagon}$.*

Proof. We show that for a chord $v \in V(G)$ there are at most six important subgraphs $G_{\pentagon}$ such that v directly encases two of the five chords. Assume a chord v_ℓ crosses the left endpoint of v such that there are three chords $\{v_2, v_3, v_4\}$ directly encased by $\{v, v_\ell\}$ and $\{v, v_\ell, v_2, v_3, v_4\}$ induce an important subgraph $G_{\pentagon}$. We show that there exist two sets of chords for which this holds. Assume $\{v, v_\ell\}$ directly encase another set of chords $\{v_2', v_3', v_4'\}$ such that $\{v, v_\ell, v_2', v_3', v_4'\}$ also induce a $G_{\pentagon}$. Since $\{v_0, v_\ell\}$ must directly encase chords in order for them to be a $G_{\pentagon}$, the chords $\{v_2, v_3, v_4\}$ must not encase any chord in $\{v_2', v_3', v_4'\}$ and vice versa. They therefore cross each other as illustrated in Figure 4.1(a), i.e. in such a way that none of them encase each other. Now assume there is a third set of

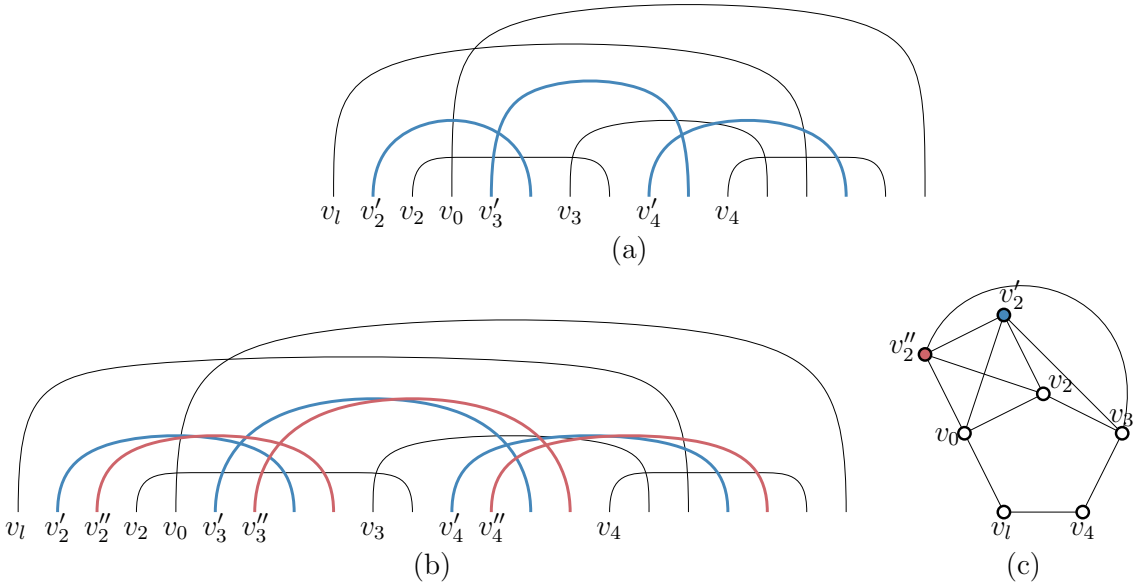


Figure 4.1.: (a) In order for $\{v, v_\ell, v_2', v_3', v_4'\}$ to induce a $G_{\pentagon}$, the vertices in $\{v_2, v_3, v_4\}$ and $\{v_2', v_3', v_4'\}$ must not encase each other, therefore they cross (b) If there is a third set of chords directly encased by $\{v, v_\ell\}$ then the chords induce 4-cliques (c) An example of 4-cliques induced by $\{v_0, v_2, v_2', v_2''\}$ and $\{v_2, v_2', v_2'', v_3\}$

chords $\{v_2'', v_3'', v_4''\}$ directly encased by $\{v, v_\ell\}$ such that $\{v, v_\ell, v_2'', v_3'', v_4''\}$ also induce a $G_{\pentagon}$. Similar to before, these chords must also cross the chords in $\{v_2, v_3, v_4\}$ and $\{v_2', v_3', v_4'\}$ such that they don't encase each other, see Figure 4.1(b). Then G contains cliques of size 4, see Figure 4.1(c), a contradiction. Hence, a set of chords $\{v_0, v_\ell\}$ can directly encase at most two sets of chords $\{v_2, v_3, v_4\}$ and $\{v_2', v_3', v_4'\}$ such that $\{v, v_\ell, v_2, v_3, v_4\}$ and $\{v, v_\ell, v_2', v_3', v_4'\}$ each induce a $G_{\pentagon}$. The case for a chord v_r crossing the right endpoint of v works analogously.

By the same arguments v can directly encase at most two sets of chords $\{v_2, v_3\}$ and $\{v_2', v_3'\}$ such that $\{v, v_1, v_2, v_3, v_4\}$ and $\{v, v_1, v_2', v_3', v_4\}$ each induce an important subgraph $G_{\pentagon}$ with $v_1 = \max((\mathcal{N}_G^r(v_2) \setminus \mathcal{N}_G^r(v_3)) \cap \mathcal{N}^r(v))$ and $v_4 = \min((\mathcal{N}_G^r(v_3) \setminus \mathcal{N}_G^r(v_2)) \cap \mathcal{N}^r(v))$, see Figure 4.2 for an illustration. Note that any chord that crosses the left endpoint of v_2'

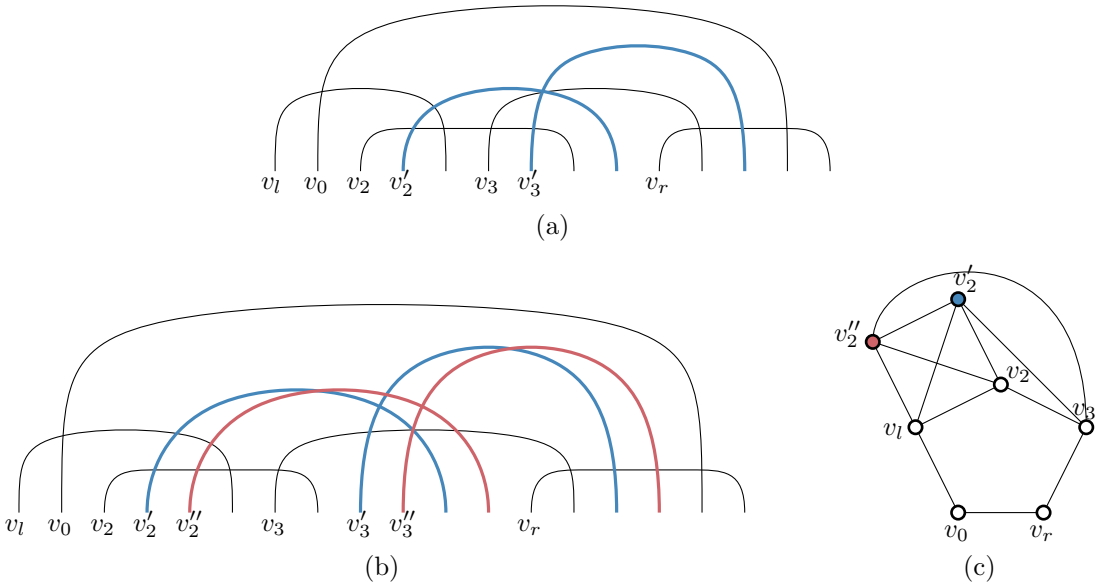


Figure 4.2.: (a) In order for v_2' and v_3' to form $G'^c_{\pentagon}$ without encasing or being encased by the chords of $G^c_{\pentagon}$ they have to cross each other (b) If there is a third set of chords directly encased by v_0, then the chords for 4-cliques (c) An example of 4-cliques formed by $\{v_l, v_2, v_2', v_2''\}$ and $\{v_2, v_2', v_2'', v_3\}$

must also cross the left endpoint of v_2 and hence $v_1 = \max((\mathcal{N}^r_G(v_2) \setminus \mathcal{N}^r_G(v_3)) \cap \mathcal{N}^r(v)) = \max((\mathcal{N}^r_G(v_2') \setminus \mathcal{N}^r_G(v_3')) \cap \mathcal{N}^r(v))$. The same applies to v_4.

In conclusion, we have that a circle graph G contains at most $n \cdot 6 = O(n)$ important subgraphs $G_{\pentagon}$. □

We now prove an upper bound for the size of $\mathbb{G}_{\bigcirc}$.

Lemma 4.7. *Let G be a circle graph with $|V(G)| = n$. Then G contains $O(n)$ important subgraphs $G_{\bigcirc} \in \mathbb{G}_{\bigcirc}$.*

Proof. We show that for a chord $v \in V(G)$ there are at most three $G_{\bigcirc} \in \mathbb{G}_{\bigcirc}$ such that v directly encases $l-2$ chords of $G_{\bigcirc}$ with $l = |V(G_{\bigcirc})|$. Let v_ℓ be a chord crossing the left endpoint of v such that $\{v, v_\ell\}$ encase a set of chords $\{v_2, \dots, v_l\}$ and $\{v, v_\ell, v_2, \dots, v_l\}$ induce an important subgraph $G_{\bigcirc}$. Assume there exists another set of chords $\{v_j' \in V(G) \colon 2 \leq j \leq l\}$ that is also directly encased by $\{v, v_\ell\}$ and w.l.o.g. induces a connected graph, i.e. a path. If a chord v_j' encases a chord v_i with $2 \leq i \leq l$, then v_i is no longer directly encased by v or v_ℓ and hence v_i is not part of the induced important subgraph $G_{\bigcirc}$. The statement is analogous for v_i encasing a chord v_j'. Hence, no chord of $\{v_2, \dots, v_l\}$ encases any chords of $\{v_j' \in V(G) \colon 2 \leq j \leq l\}$ and vice versa. Therefore, a chord v_j' crosses two crossing chords v_i and $v_{(i+1) \bmod l}$ and $v_i^\ell < v_j'^\ell < v_i^r < v_j'^r$, in which case $j = i$, see Figure 4.3(a). By Definition 3.7, $v_i = \min\left(\mathcal{N}_G(v_{i-1}) \cap \mathcal{N}_G(v_{(i+1) \bmod l})\right)$. Let $v_k' \coloneqq \max\{v_j' : 2 \leq j \leq m\}$ with v_k crossing v_i and $v_{(i+1) \bmod l}$. Then

$$v_k' = \min\left(\mathcal{N}_G(v_{k-1}') \cap \mathcal{N}_G(v_{(i+1) \bmod l})\right) = \min(\{v_{k-2}', v_k', v_{i-1}, v_i\} \cap \{v_{i+2 \bmod l}, v_i, v_k'\})$$
$$= \min(v_i, v_k') = v_i$$

since $v_i^\ell < v_k'^\ell < v_i^r < v_k'^r$. Therefore, the chord v_k' is not part of any induced important subgraph $G_{\bigcirc}$. Since this holds for all v_j' we have that $\{v, v_\ell\} \cup \{v_j' \colon 2 \leq j \leq l\}$ does not

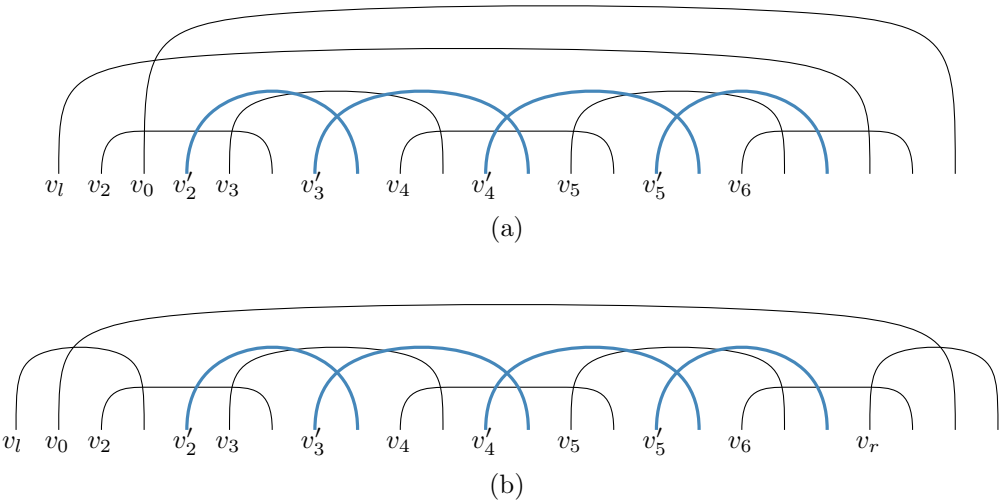


Figure 4.3.: In order for $\{v, v_\ell, v'_2, \ldots, v'_5\}$ to induce a $G_{\pentagon}$ without encasing or being encased by the chords of the $G_{\pentagon}$ induced by $\{v, v_\ell, v'_2, \ldots, v'_5\}$ they have to cross each other

induce an important subgraph $G_{\bigcirc}$ and therefore the important subgraph $G_{\bigcirc}$ induced by $\{v, v_\ell, v_2, \ldots v_l\}$ is unique. The case for a chord v_r crossing the right endpoint of v works analogously.

We apply the same arguments and show that v can induce only one important subgraph $G_{\bigcirc}$ with a set of chords $\{v_1, v_2, \ldots, v_l\}$ where $\{v_2, \ldots, v_{l-1}\}$ are directly encased by v and $v_1 = \max((\mathcal{N}^r_G(v_2) \setminus \mathcal{N}^r_G(v_3)) \cap \mathcal{N}^r(v))$ and $v_l = \min((\mathcal{N}^r_G(v_{l-1}) \setminus \mathcal{N}^r_G(v_{l-2})) \cap \mathcal{N}^r(v))$, see Figure 4.2 for an illustration. Assume there exists another set of chords $\{v'_j : 2 \leq j < m\}$ that are directly encased by v such that they also induce an important subgraph $G_{\bigcirc}$. Again, these chords must not encase any of the chords in $\{v_2, \ldots, v_{l-1}\}$ and vice versa. Since no chord v'_j encases some chord v_i and vice versa, we have that v'_j crosses two crossing chords v_i and v_{i+1} in which case $j = i$ and $v^\ell_i < v'^\ell_j < v^r_i < v'^r_j$. Let $v'_k := \max\{v'_j : 2 \leq j \leq m\}$ with v_k crossing v_i and v_{i+1} and $v_{i+1} \in V(G'^\ell_{\bigcirc})$. Then by Definition 3.7 and with $v^\ell_i < v'^\ell_k < v^r_i < v'^r_k$ we have

$$\begin{aligned} v'_k = \min\left(\mathcal{N}_G(v'_{k-1}) \cap \mathcal{N}_G(v_{i+1})\right) &= \min\left(\{v'_{k-2}, v'_k, v_{i-1}, v_{i-1}\} \cap \{v_{i+2}, v_i, v'_k\}\right) \\ &= \min\left(\{v_i, v'_k\}\right) = v_i \end{aligned}$$

Therefore v'_k is not a chord of the important subgraph $G_{\bigcirc}$ induced by v, v_1, v_l and the chords directly encased by v. Since this holds for all v'_j we have that the important subgraph $G_{\bigcirc}$ induced by $v, v_1, v_2, \ldots, v_l$ is unique. □

4.2. Constructing Clauses of Size 2

To construct F_1, we consider the important subgraphs $G_{\triangleleft}$, $G_{\square}$, $G_{\boxslash}$ and $G_{\pentagon}$. From the coloring constraints we defined in Lemma 3.8(a) to (d) we infer boolean clauses of length 2 as follows.

$G_{\triangleleft}$: $(c^*(v_0, v_2) \vee c^*(v_0, v_3)) \wedge (c^*(v_0, v_2) \vee c^*(v_0, v_3))$

$G_{\square}$: $c^*(v_0, v_2) \vee c^*(v_1, v_3)$

$G_{\boxslash}$: $c^*(v_0, v_2)$

$G_{\pentagon}$: $(\neg c^*(v_0, v_2) \vee \neg c^*(v_0, v_3)) \wedge (\neg c^*(v_3, v_0) \vee \neg c^*(v_3, v_1)) \wedge (\neg c^*(v_1, v_3) \vee \neg c^*(v_1, v_4)) \wedge (\neg c^*(v_4, v_1) \vee \neg c^*(v_4, v_2)) \wedge (\neg c^*(v_2, v_4) \vee \neg c^*(v_2, v_0))$

The conjunction of all these clauses for each important subgraph gives us a 2-SAT instance F_1. From Lemma 4.1 to 4.6 we know that the number of important subgraph is polynomially bounded, hence F_1 has polynomial size and therefore a satisfying assignment for F_1 can be found in polynomial time [2].

4.3. Constructing Clauses for $G_{\pentagon}$ and $\mathbb{G}_{\bigcirc}$

To ensure that c^* satisfy the constrained given by property G4.2 in Lemma 3.8(d), i.e. for exactly two distinct pairs c^* is `true`, we add the following clauses to our formula F_2 for every $G_{\pentagon}$ with $V(G_{\pentagon}) = \{v_0, \dots, v_4\}$ as in Figure 3.4. For each possible permutation of $c^*(v_i, v_{(i+2) \bmod 5})$ and $c^*(v_j, v_{(j+2) \bmod 5})$ where $i, j \in \{0, 1, 2, 3, 4\}$ and $i \neq j$ we construct a conjuction in which every value of c^* that is defined for $V(G_{\pentagon})$ is negated except for $c^*(v_i, v_{(i+2) \bmod 5})$ and $c^*(v_j, v_{(j+2) \bmod 5})$. We then construct the disjunction of all these conjunctions. This disjunction is equivalent to the constraint G4.2 in Lemma 3.8(d). Since this formula is a disjunction of conjunctions, i.e. it is in DNF, and we get such a formula for every $G_{\pentagon}$, which are $O(n)$ many by Lemma 4.6, we cannot solve it in polynomial time.

We now construct a boolean formula for $G_{\bigcirc} \in \mathbb{G}_{\bigcirc}$ using Lemma 3.9. Let $G_{\bigcirc} \in \mathbb{G}_{\bigcirc}$ with $V(G_{\bigcirc}) = \{v_0, \dots, v_l\}$. Recall that we used a recursive function $h : V(G^2) \to \mathbb{B}$ to determine if a vertex v_i has the same color as v_x with $x \in \{0, 1\}$ and $x + 3 \leq i \leq l$ in some 3-coloring col that realizes c^*. From Lemma 3.9 it follows that c^* is realizable on G if and only if $h(v_0, v_1) = h(v_1, v_2) = h(v_0, v_l) = \texttt{false}$, hence we require our boolean formula to include

$$\neg h(v_0, v_1) \wedge \neg h(v_1, v_2) \wedge \neg h(v_0, v_l)$$

Next, we add to our formula

$$h(v_x, v_{x+2}) \Leftrightarrow c^*(v_x, v_{x+2})$$

as well as

$$(h(v_0, v_{l-1}) \Leftrightarrow c^*(v_0, v_{l-1})) \wedge (h(v_1, v_l) \Leftrightarrow c^*(v_1, v_l))$$

These clauses also follow directly from Lemma 3.9.

Lastly, we need to add clauses representing the recursion of h as follows. We can formulate the statement $h(v_x, v_i) = h(v_x, v_{i-2})$ if $c^*(v_{i-2}, v_i) = \texttt{true}$ as

$$c^*(v_{i-2}, v_i) \Rightarrow (h(v_x, v_i) \Leftrightarrow h(v_x, v_{i-2}))$$

Similarly, from $h(v_x, v_i) = \neg h(v_x, v_{i-2}) \wedge \neg h(v_x, v_{i-1})$ if $c^*(v_{i-2}, v_i) = \texttt{false}$ we get

$$\neg c^*(v_{i-2}, v_i) \Rightarrow (h(v_x, v_i) \Leftrightarrow (\neg h(v_x, v_{i-2}) \wedge \neg h(v_x, v_{i-1})))$$

Putting all of these clauses together, we get the following boolean formula for $G_{\bigcirc}$:

$$\begin{gathered}\neg h(v_0, v_1) \wedge \neg h(v_1, v_2) \wedge \neg h(v_0, v_l) \wedge (h(v_x, v_{x+2}) \Leftrightarrow c^*(v_x, v_{x+2})) \wedge \\ (h(v_0, v_{l-1}) \Leftrightarrow c^*(v_0, v_{l-1})) \wedge (h(v_1, v_l) \Leftrightarrow c^*(v_1, v_l)) \wedge \\ (c^*(v_{i-2}, v_i) \Rightarrow (h(v_x, v_i) \Leftrightarrow h(v_x, v_{i-2}))) \wedge \\ (\neg c^*(v_{i-2}, v_i) \Rightarrow (h(v_x, v_i) \Leftrightarrow (\neg h(v_x, v_{i-2}) \wedge \neg h(v_x, v_{i-1}))))\end{gathered}$$

This formula clearly is also not a 2-SAT instance and therefore not solvable in polynomial time. The conjunction of the DNF formulas for all $G_{\pentagon}$ and the formulas for all $G_{\bigcirc} \in \mathbb{G}_{\bigcirc}$ gives us our boolean formula F_2.

4.4. Finding c^* values for $G_{\pentagon}$ and $\mathbb{G}_{\bigcirc}$ using Backtracking

In the following, we say that the value of $c^*(a, b)$ is *relevant* for an important subgraph I, if there exists a pair of vertices $\{a, b\} \in V(I)$ for which $c*$ is defined. As we have seen in Section 4.3, to find values such that c^* has the properties we required in Lemma 3.8 and 3.9 we would need to solve a boolean formula that is not a 2-SAT instance. This is due to the more complicated formulas we get for the important subgraphs $G_{\pentagon}$ and $G_{\bigcirc} \in \mathbb{G}_{\bigcirc}$. To find a value assignment for all the c^* values that are relevant for $G_{\pentagon}$ and $\mathbb{G}_{\bigcirc}$ we present the following backtracking algorithm. This algorithm is derived from Unger's description [51, p. 108f.].

We first consider the important subgraphs $G_{\pentagon}$. From solving the 2-SAT formula we get values that satisfy property G4.1. We have to verify that these values also hold up to property G4.2. For this we simply verify if the current values for c^* can yield a 3-coloring by applying the recursive auxiliary function h from Lemma 3.9. If $h(v_0, v_4) =$ `false`, $h(v_0, v_3) = c^*(v_0, v_3)$ and $h(v_1, v_4) = c^*(v_1, v_4)$ then by Lemma 3.9 c^* is realizable and the values for c^* are kept, and we move on to the next $G_{\pentagon}$. Note that we don't require a specific order in which we handle each $G_{\pentagon}$. If one of these statements does not hold for h, or the values result in our 2-SAT formula being unsatisfiable, we choose new values for c^*. Since we have five relevant c^* values for $G_{\pentagon}$ we can try up to 2^5 assignments for c^* for one $G_{\pentagon}$. When choosing new values for c^* we pick one of the 2^5 possibilities which we have not tried yet. If for a $G_{\pentagon}$ none of the possible assignments pass the checks for c^* to be realizable, we backtrack to the previous $G_{\pentagon}$ if possible and try the next assignment there. If we are not able to backtrack any further, i.e. we have tried all possible assignments for all $G_{\pentagon}$, we stop and know that $G_{\pentagon}$ is not 3-colorable. If we find an assignment for c^* such that for all $G_{\pentagon}$ the c^* values satisfy the 2-SAT instance and also property G4.2, i.e. c^* can be realized for each $G_{\pentagon}$ by a 3-coloring by Lemma 3.9, then we move on to $\mathbb{G}_{\bigcirc}$.

The backtracking algorithm works similar for $\mathbb{G}_{\bigcirc}$. We choose $G_{\bigcirc} \in \mathbb{G}_{\bigcirc}$ as the one with the least undefined relevant c^* values up to this point, i.e. with the least relevant values of c^* for $G_{\bigcirc}$ that have not been assigned a value by solving the 2-SAT instance. We assign the free c^* variables of $G_{\bigcirc}$ a possible combination of values we haven't tried yet, similar to what we did for $G_{\pentagon}$, and apply our function h to all c^* values of $G_{\bigcirc}$. Similar to before, we pick a new assignment for the undefined c^* variables should $h(v_0, v_l) =$ `false`, $h(v_0, v_{l-1}) = c^*(v_0, v_{l-1})$ or $h(v_1, v_l) = c^*(v_1, v_l)$ not hold. An illustration of the corresponding backtracking tree is given in Figure 4.4.

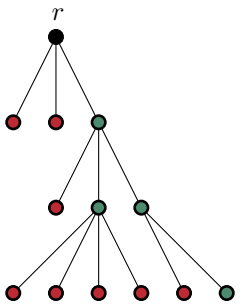


Figure 4.4.: Example backtracking tree. The root r represents the value assignments for c^* given by the 2-SAT instance. Every level in the backtracking tree represents c^* value assignments for one $G_{\pentagon}$ resp. $G_{\bigcirc} \in \mathbb{G}_{\bigcirc}$. Red vertices represent invalid c^* value assignments, green vertices represent valid c^* value assignments.

4.4.1. Discussing the Running Time of the Backtracking Algorithm

At every node in our backtracking tree we verify the assignment chosen for c^*. The procedure is given in Lemma 3.9. Using dynamic programming we can easily compute h in linear time and verify in constant time if our assignment for c^* is valid using h. The running time of our backtracking algorithm is therefore as follows. Since a valid c^* assignment for $\mathbb{G}_{\bigcirc}$ is equivalent to a path in our backtracking tree from the root to the deepest leaf with a positive result, the running time for finding and verifying this assignment is $O(n^2)$. We also have to consider all the other leaves that returned a negative result. For every such leaf x we get a running time of $\text{depth}(x) \cdot O(n) = O(n^2)$. Our total running time is then $O(n^2) + \text{depth}(x) \cdot O(n^2) = (\text{depth}(x) + 1)O(n^2)$.

We see that the running time depends on the number of leaves in the backtracking tree. Therefore, we need to show that the number of leaves is polynomially bounded in order to maintain an efficient algorithm for computing c^*. Unger claims that the backtracking tree constructed by the backtracking algorithm has $O(\log n)$ leaves. This means that according to him we only have to try out $O(\log n)$ assignments until we find an assigment for c^* that satisfies all constraints for all important subgraphs. We first present the definitions needed for the arguments Unger gives to support his claim that the backtracking tree has only $O(\log n)$ leaves.

Definition 4.8. *Let G be a circle graph with its consistent auxiliary coloring function c^* and G' be the important subgraph $G_{\lhd}$, $G_{\square}$ or $G_{\pentagon}$ of G. Let $c^*(x,y) \vee c^*(x',y')$ be one of the clauses we get from G'. Then $\neg c^*(x,y) \nRightarrow c^*(x',y')$ is a* clear implication*. For a clause $\neg c^*(x,y) \vee \neg c^*(x',y')$ we have that $c^*(x,y) \nRightarrow \neg c^*(x,z)$ is a* clear implication*.*

The transitive closure for clear implications is then defined as follows.

Definition 4.9. *Let $\mathcal{D} \coloneqq \{(a,b) \in V(G)^2 : b \in \mathcal{N}_G(\mathcal{N}_G(a)) \setminus \mathcal{N}_G(a)\}$ as in Definition 3.1 and $(x,y),(x',y') \in \mathcal{D}$. We define*

- *$c^*(x,y) \Rrightarrow c^*(x',y')$, if there exists a sequence of chords a_i, b_i with $(a_i,b_i) \in \mathcal{D}$ for $1 \leq i \leq k$ with*

$$(a_0,b_0) = (x,y)$$
$$(a_k,b_k) = (x',y')$$
$$c^*(a_i,b_i) \nRightarrow \neg(a_{i+1},b_{i+1}) \text{ for } 1 \leq i \leq k \text{ and } i \text{ even}$$
$$\neg c^*(a_i,b_i) \nRightarrow c^*(a_{i+1},b_{i+1}) \text{ for } 1 \leq i \leq k \text{ and } i \text{ odd}$$

- *$c^*(x,y) \Rrightarrow \neg c^*(x',y')$, if $c^*(x,y) \nRightarrow \neg c^*(x',y')$ or if there exist two chords a, b with $(a,b) \in \mathcal{D}$ and*

$$c^*(x,y) \Rrightarrow c^*(a,b) \text{ and}$$
$$c^*(a,b) \nRightarrow \neg c^*(x',y')$$

- *$\neg c^*(x,y) \Rrightarrow c^*(x',y')$, if $\neg c^*(x,y) \nRightarrow c^*(x',y')$ or if there exist two chords a, b with $(a,b) \in \mathcal{D}$ and*

$$\neg c^*(x,y) \nRightarrow c^*(a,b) \text{ and}$$
$$c^*(a,b) \Rrightarrow c^*(x',y')$$

- *$\neg c^*(x,y) \Rrightarrow \neg c^*(x',y')$, if there exist two chords a, b with $(a,b) \in \mathcal{D}$ and*

$$\neg c^*(x,y) \nRightarrow c^*(a,b) \text{ and}$$
$$c^*(a,b) \Rrightarrow \neg c^*(x',y')$$

Definition 4.10. *Let $(a,b) \twoheadrightarrow (c,d)$ and $(e,f) \Rightarrow (g,h)$ be two implications. They are* independent *if the following does not hold:*

- $(a,b) \Rightarrow (g,h)$ *or*
- $(e,f) \Rightarrow (c,d)$

With these definitions we are able to present the lemma that is crucial for the argument, that the backtracking tree has $O(\log n)$ leaves.

Unclear Lemma 4.11. *[51, p. 107] Let G be a circle graph with*

- $\{a_i, b_i\}$ *encases* $\{a_{i+1}, b_{i+1}\}$ *for* $1 \leq i < k$
- $\{c_i, d_i\}$ *encases* $\{c_{i+1}, d_{i+1}\}$ *for* $1 \leq i < k$
- $(a_i, b_i) \Rightarrow (c_{p(i)}, d_{p(i)})$ *for* $1 \leq i < k$ *and some permutation* p

Further, for $1 \leq i < j \leq k$ let $(c_i, b_i) \Rightarrow (c_{p(i)}, d_{p(i)})$ and $(c_j, b_j) \Rightarrow (c_{p(j)}, d_{p(j)})$ be independent implications.

Then G has at least 2^k chords.

In the original proof, Unger lists four examples of circle graphs and claims, that for all examples, if another independent implication is added then this would double the number of chords of the circle graph. Unfortunately, there is no explanation given as to why the presented examples cover all possible cases. It is also not clear to us how one would construct another or any independent implication. No arguments for this are given by Unger, either. Since we were not able to reconstruct the proof for this claim, we must regard the statement as non-proven for this thesis. Instead, we give two explanations to further show why we doubt it is that this statement holds.

4.4.2. Naive Worst-Case for Exponential Number of Leaves

We first argue how in a worst-case scenario our backtracking tree might grow to have $2^{O(n^2)}$ leaves. For every $G_{\bigcirc} \in \mathbb{G}_{\bigcirc}$ we have $|V(G_{\bigcirc})| = l$ values of c^* for vertices of $G_{\bigcirc}$. This means that, assuming no values of c^* are otherwise given, we have to try at most 2^l values for c^* for $G_{\bigcirc}$. From Lemma 4.7 we know that we have $O(n)$ subgraphs $G_{\bigcirc}$. Consider the following scenario. Let $\mathbb{G}_{\bigcirc} = \{G_{\bigcirc 0}, \ldots, G_{\bigcirc k}\}$ and let $0, \ldots, k$ be the order in which we search values for c^* using the backtracking algorithm. Assume that for $G_{\bigcirc 0}$ up to $G_{\bigcirc k-1}$ the first c^* assignment we try yields a positive result. But for the last remaining $G_{\bigcirc k}$ we try all $2^{|V(G_{\bigcirc k})|}$ assignments for c^* and all yield a negative result. We then try the next assignment for $G_{\bigcirc k-1}$ and again get a positive result. But for $G_{\bigcirc k}$ we again get $2^{|V(G_{\bigcirc k})}$ negative results. This continues until we have tried evey possible assignment for all $G_{\bigcirc i}$ with $i \in \{0, \ldots, k\}$. Since every inner node in our backtracking tree can have up to $2^{|V(G_{\bigcirc i})|}$ children and clearly $|V(G_{\bigcirc i})| \in O(n)$, we get $(2^{O(n)})^{O(n)} = 2^{O(n^2)}$ leaves in our backtracking tree. This is an upper bound on a possible worst-case size of the backtracking tree. We note that we were not able to construct an explicit (counter-)example for this scenario. On one hand, it is not clear to us that there exists a circle graph such that computing values for c^* results in a backtracking tree with exponentially many leaves. On the other hand, the converse also does not seem obvious, namely why this scenario should never occur. One argument might be as follows. We either don't have many c^* values given for a $G_{\bigcirc}$ which means that in theory we must try many c^* assignments, but in practice we find a valid assignment for c^* fairly fast since there are not many other c^* values correlating with our choice of c^*. If we do have values for c^* given for $G_{\bigcirc}$ by other important subgraphs then the more such values there are the fewer values we are left with to try out. Concretizing these ideas is left as an open problem.

4.4.3. Backtracking parameterized in the largest $G_{\bigcirc} \in \mathbb{G}_{\bigcirc}$

We want to briefly discuss the idea of parameterizing the number of leaves in our backtracking tree by the size of the largest $G_{\bigcirc}$. Let $k \coloneqq \max\{l \colon l = |V(G_{\bigcirc})| \text{ with } G_{\bigcirc} \in \mathbb{G}_{\bigcirc}\}$ be the size of the largest $G_{\bigcirc}$. Similar to Section 4.4.2, we can argue that every inner node in out backtracking tree has at most 2^k children and that the size of $\mathbb{G}_{\bigcirc}$ is $O(n)$. It is easy to see that the number of leaves is still at most $2^{k \cdot O(n)}$. In order do state that the backtracking algorithm is not FPT we have to show that the size of $\mathbb{G}_{\bigcirc}$ is in fact $\Omega(n)$ and that such a worst-case backtracking tree can occur. At this point we are not able to give a proof for this. But, similar to the arguments given in Section 4.4.2, it seems unlikely that our backtracking algorithm is FPT in the size of the largest $G_{\bigcirc} \in \mathbb{G}_{\bigcirc}$.

5. Experimental Evaluation

In this chapter we present the results of our experimental evaluation of the algorithms described in Chapters 3 and 4. We evaluate the running time of our algorithms as well as the backtracking algorithm for $\mathbb{G}_{\bigcirc}$. We also evaluate the backtracking trees for $\mathbb{G}_{\bigcirc}$. We consider the number of backtracking leaves, the amount of backtracking leaves which are actual valid assignments for $c*$ and the amount of backtracking leaves visited until a first solution is found in relation to the size of the worst-case number of leaves in relation to both the number of vertices in the circle graph and the number of important subgraphs in $\mathbb{G}_{\bigcirc}$. Note that here we consider a worst-case backtracking tree to be one where we prune the tree at inner nodes with invalid value assignments for c^* and try all possible assignments for c^* when we reach the last remaining $G_{\bigcirc} \in \mathbb{G}_{\bigcirc}$. By evaluating the number of leaves in a backtracking tree that represent a valid value assignment for c^* we hope to get some insight on how high our chances are of having many good solutions in our tree. The relation of the number of worst-case leaves and the number of leaves we actually visit for our test graphs is meant to show how good our heuristic is at finding the first valid leaf in our tree. We also present the absolute running time for both the backtracking for $\mathbb{G}_{\bigcirc}$ and computing c^* using a SAT solver. In the following we give a high level description of the code before presenting the results of the experimental evaluation.

5.1. Implementation

A circle graph G is read as input. The graph contains the left and right endpoints of each chord as well as the level assignment for each chord. After reading the graph, the first step is to find all important subgraphs. For this, we use relatively straightforward recognition algorithms that we derived from the recognition algorithms presented by Unger [51] with slight alterations to realize them in Python. These have not been fully optimized, since our main focus is the running time of the backtracking algorithm. After finding all important subgraphs we compute values for c^*. To this end, we first build all 2-SAT clauses for the important subgraphs $G_{\dashv\triangleleft}$, $G_{\square}$, $G_{\boxslash}$ and $G_{\pentagon}$ as described in Section 4.2. Then we use a SAT solver to find a satisfying assignment for the values of c^* used in our 2-SAT formula. Then, we use one of the following two approaches to finalize the values of c^* needed for $G_{\pentagon}$ and $\mathbb{G}_{\bigcirc}$. Our first approach is the backtracking algorithm described in Chapter 4, Section 4.4 to find valid assignments for the c^* values of all $G_{\pentagon}$ and $\mathbb{G}_{\bigcirc}$. Our second approach is to build clauses for these important subgraphs and then use the SAT solver from before to find the values for c^*. We compare the efficiency of these two approaches in Sections 5.3.1 and 5.3.2.

5.2. Sample Data

To test the efficiency of our python application we generated 1822 test graphs of different sizes and using different generation methods. All of these graphs are connected and do not contain cliques of size 4 or greater. The graphs are built incrementally, i.e. we randomly add one chord after another and check, whether the resulting graph still satisfies certain constraints. These constraints depend on the generation method. If the constraints are satisfied we add the chord to the graph and repeat these steps until we reach a graph with the desired number of vertices. The two constraints all graph generators have in common is that the graph must remain connected and free of any k-cliques with $k \geq 4$ The graphs are divided into two categories based on the way they were generated. The first graph category are graphs where the chord added in step i has color i mod 3. We refer to this category as C. We assign colors to chords while generating the graph which ensures that the resulting graph is 3-colorable. The color that is assigned to a new chord c_i is i mod 3. If the color is already used by a neighbor, i.e. the resulting coloring is not a 3-coloring, we don't keep the chord and try inserting it another way, still requiring it to have color i mod 3. We slightly modify this method to get our second generation strategy which we call RC. Instead of assigning the color i mod 3, we randomly choose a color $r_i \in \{0, 1, 2\}$ for the chord added in step i. The remaining generation strategy works as before, namely if adding the new chord with this randomly chosen color r_i breaks our 3-coloring, we try to add the colored chord a different way.

Both categories contain only 3-colorable graphs. This allows us to evaluate how accurate the algorithms is able to decide if a graph is 3-colorable. From the counterexample in Section 3.2 we know that following the coloring algorithms using a consistent auxiliary coloring function c^* might not always result in a 3-coloring. To see how well the algorithm performs on our random 3-colorable graphs we verified the coloring after following the steps given in Section 3.2. If the coloring is not correct, i.e. there are at least two crossing chords with the same color, we marked the result as invalid. Otherwise, the result was considered valid. An evaluation on the percentage of valid results is given in the following section. This way of generating instances further proved to yield more interesting graphs, that is graphs that contained several important subgraphs $G_{\bigcirc} \in \mathbb{G}_{\bigcirc}$. An example of graphs generated and colored by our application is given in Figure A.1 and A.2 in Appendix A.

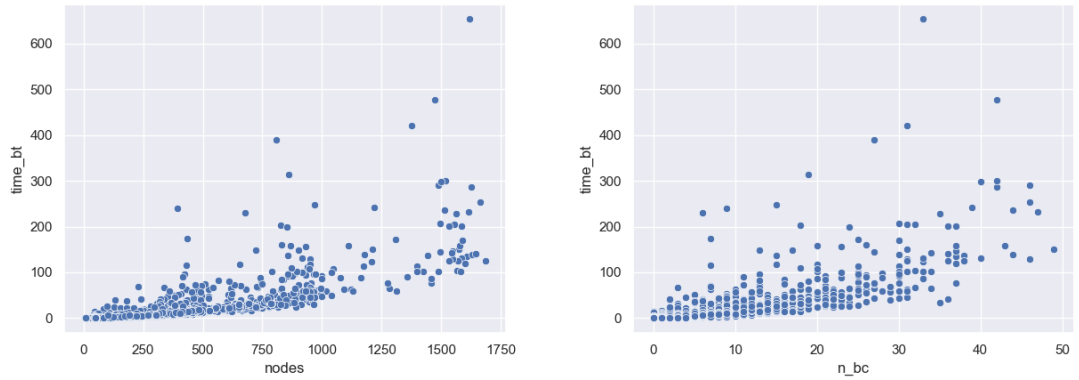
5.3. Evaluation

For the evaluation, we implemented a Python application using Python 3.10 for the backend and the JavaScript framework D3.js [1] in our frontend. We evaluated the efficiency of this application and the used algorithms on cluster nodes each equipped with two Intel Xeon E5 CPUs and 256 GiB RAM. We stored our sample graphs as JSON files, containing the required metadata, i.e. the left and right indices and level of all chords as well as the important subgraphs each chord belongs to. For solving the SAT instances, we used the Python library pySMT [2] with the SMT solver MathSAT 5 [10]. In the following we present the results for both sample data categories combined.

5.3.1. Computing c^* using Backtracking

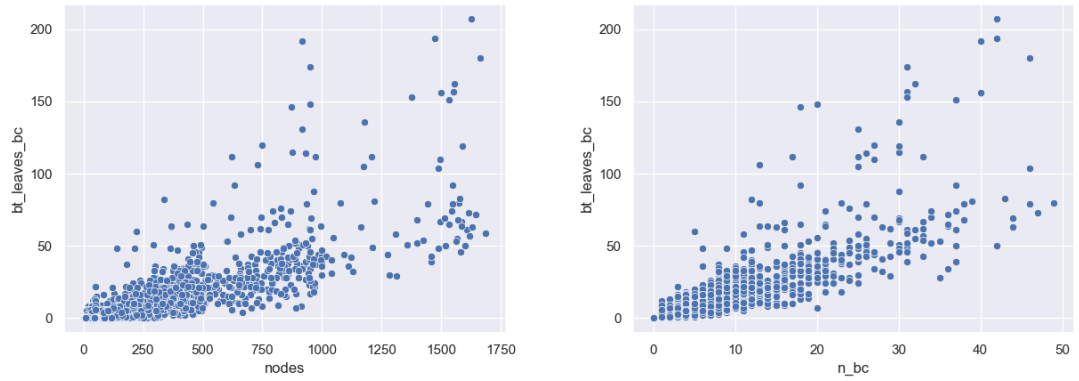
We present the following results for our implementation of the backtracking algorithm. For the absolute running time of the backtracking algorithm, we notice a exponential tendency in our data, especially for graphs with more nodes and $\mathbb{G}_{\bigcirc}$, see Figure 5.1. When looking at the number of backtracking leaves in relation to the number of nodes, resp. the size of

[1] https://d3js.org/
[2] http://www.pysmt.org/



(a) (b)

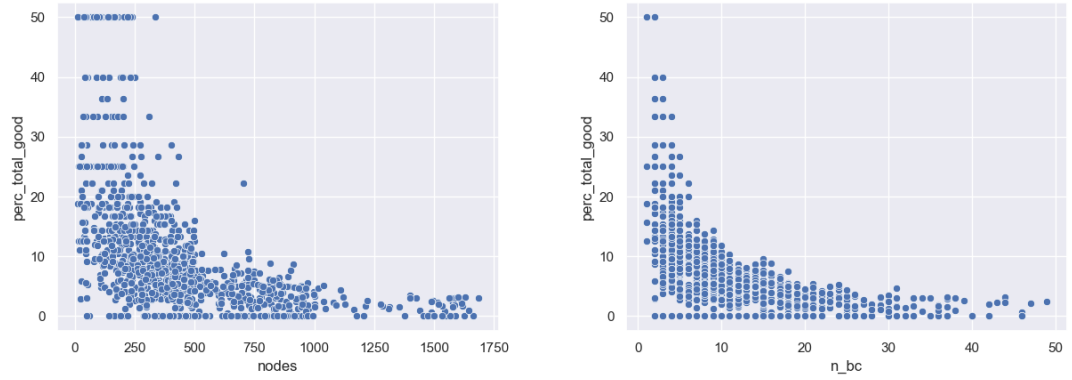
Figure 5.1.: Absolute running time in fractional seconds of backtracking algorithm for $\mathbb{G}_{\bigcirc}$ in relation to (a) the number of nodes in the graph and (b) the number of important subgraphs in $\mathbb{G}_{\bigcirc}$



(a) (b)

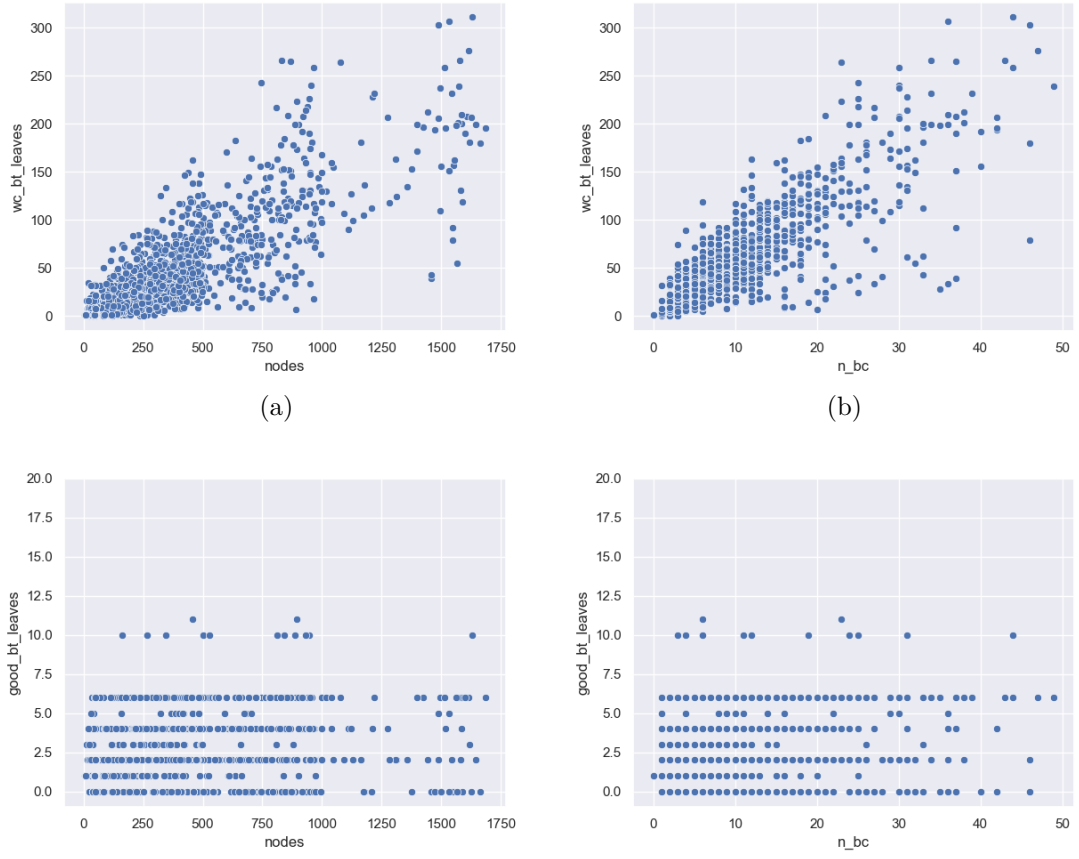
Figure 5.2.: Number of backtracking leaves in relation to (a) the number of nodes in the graph and (b) the number of important subgraphs in $\mathbb{G}_{\bigcirc}$

$\mathbb{G}_{\bigcirc}$, we also see a exponential tendency, again more so for graphs with more nodes and $\mathbb{G}_{\bigcirc}$, see Figure 5.2. But we also note that there is a not insignificant amount of scattering for these values. In order to get more insight into the properties of the backtracking trees we also evaluated the backtracking trees themselves. Namely, we looked at the relation of leaves that represent solutions and leaves that represent invalid c^* value assignments. We also evaluated the number of leaves that we actually visit until we find a solution in relation to the number of leaves in the worst-case backtracking tree. For the percentage of leaves that are solutions in a worst-case backtracking tree we see a rapid decline as the graphs increase in size, both in respect to the number of nodes and the size of $\mathbb{G}_{\bigcirc}$, see Figure 5.3. Figure 5.4 indicates that as we have more nodes in the graph and important subgraphs in $\mathbb{G}_{\bigcirc}$, our backtracking trees grow larger while the number of solutions remains virtually constant. From these numbers it might seem that the chance of quickly finding a first solution in the backtracking tree would decrease as there are fewer solution leaves in comparison to the number of all leaves. Interestingly, the evaluation shows that the number of leaves actually visited until a first solution is reached behaves a bit differently than maybe expected. Figure 5.5 shows that as the number of nodes and the number of important subgraphs $\mathbb{G}_{\bigcirc}$ in the graph grows, we either visit all nodes of the backtracking



(a) (b)

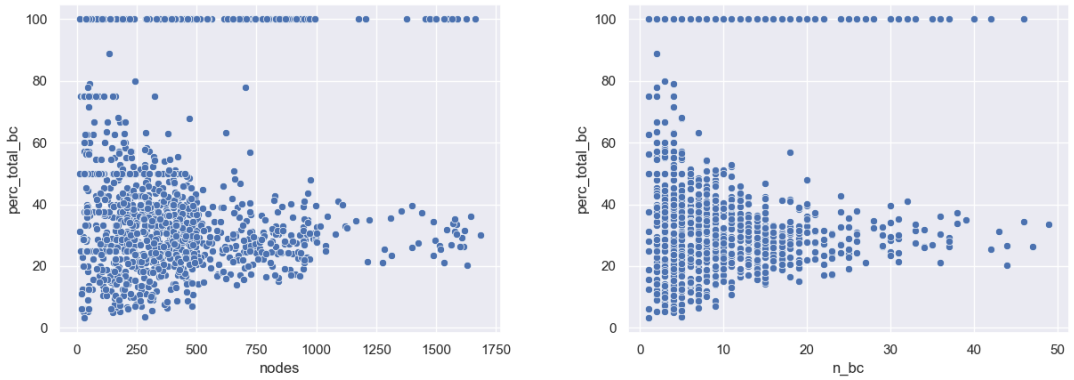
Figure 5.3.: Percentage of leaves in a worst-case backtracking tree that are solutions in relation to (a) the number of nodes in the graph and (b) the number of important subgraphs in $\mathbb{G}_{\bigcirc}$



(a) (b)

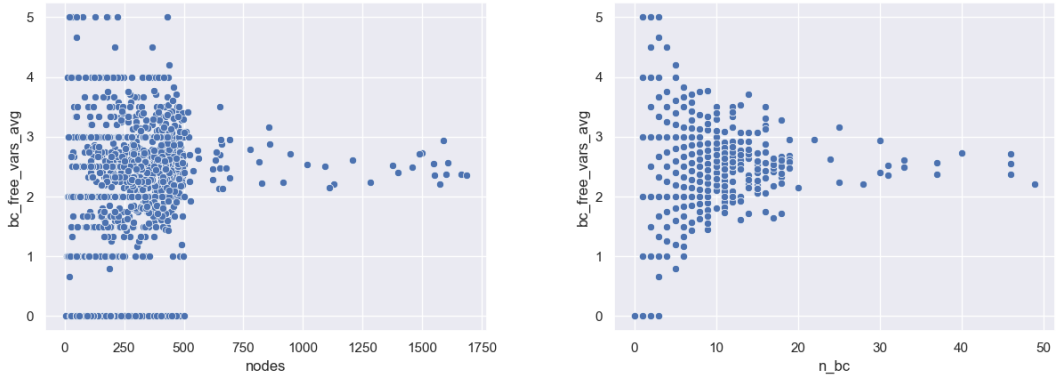
(c) (d)

Figure 5.4.: Absolute number of leaves in a worst-case backtracking tree (top) and absolute number of solutions in a worst-case backtracking tree (bottom) in relation to the number of nodes in the graph (left) and the number of important subgraphs in $\mathbb{G}_{\bigcirc}$ (right)



(a) (b)

Figure 5.5.: Number of backtracking leaves in relation to (a) the number of nodes in the graph and (b) the number of important subgraphs in $\mathbb{G}_{\bigcirc}$



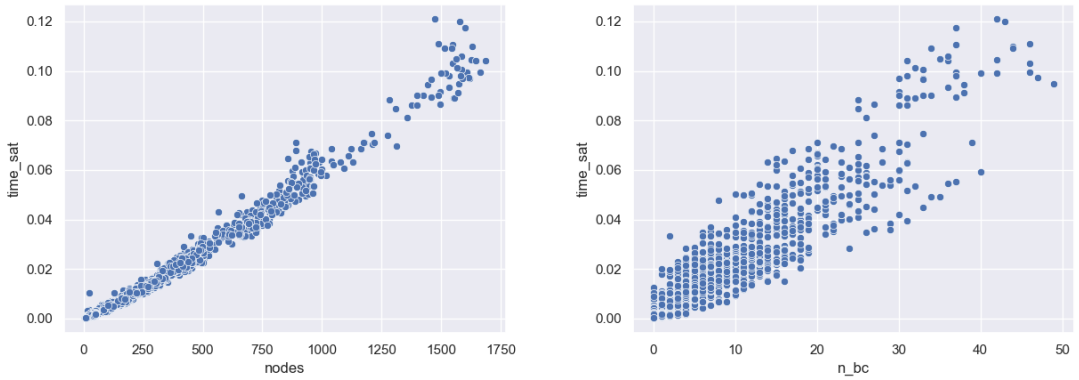
(a) (b)

Figure 5.6.: Number of backtracking leaves in relation to (a) the number of nodes in the graph and (b) the number of important subgraphs in $\mathbb{G}_{\bigcirc}$

tree, i.e. 100%, or about only about 20% to 40%, so significantly less than half. We found that this tendency seems to correlate with the average number of c^* values that are relevant to $\mathbb{G}_{\bigcirc}$ and not shared with other important subgraphs, so the values that we essentially have to "try out" during the backtracking algorithm. Figure 5.6 shows that the number of these c^* values mostly averages at around 3 to 4 per circle graph. Especially Figure 5.6(b) suggests a correlation between these figures. This leads us to believe that there might be a connection between the number of leaves we need to visit until we find a solution and the number of c^* values that are relevant for $\mathbb{G}_{\bigcirc}$ and not shared with other subgraphs. This would then partly coincide with the assumptions made by Unger regarding the number of leaves in the backtracking tree, although a formal proof of this would still be needed.

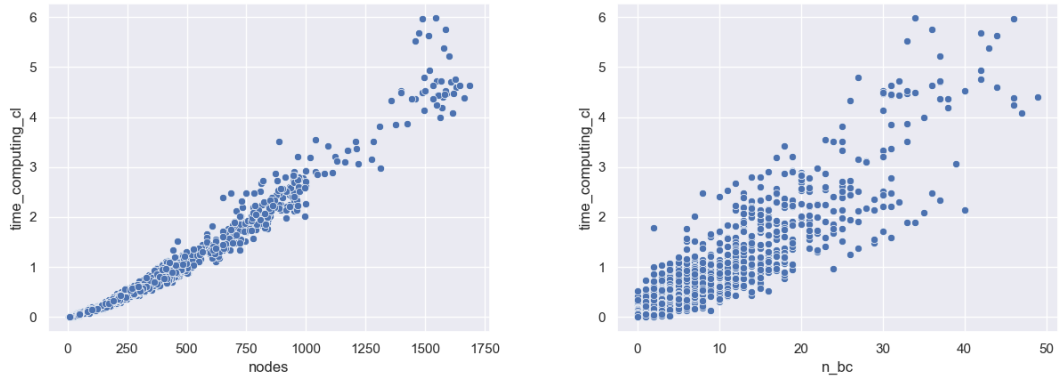
5.3.2. Computing c^* using 3-Sat

For our SAT solver approach we looked at the development of the absolute running time for computing a solution for our formula. Compared to the running time for the backtracking algorithm, we notice a seemingly linear time increase for computing c^* using the SAT solver as opposed to the more exponentially-seeming and scattered time increase for the backtracking algorithm, see Figure 5.7. The SAT solver also performs better in absolute numbers. While the backtracking algorithm needs up to 300 seconds per graph in most



(a) (b)

Figure 5.7.: Absolute running time in fractional seconds of computing a solution for $F_1 \wedge F_2$ in relation to (a) the number of nodes in the graph and (b) the number of important subgraphs in $\mathbb{G}_{\bigcirc}$



(a) (b)

Figure 5.8.: Absolute running time in fractional seconds of computing a solution for $F_1 \wedge F_2$ in relation to (a) the number of nodes in the graph and (b) the number of important subgraphs in $\mathbb{G}_{\bigcirc}$

cases, the SAT solver needs no more than 0.13 seconds, even for larger graphs. If we also factor in the time it takes to compute all clauses, see Figure 5.8, the SAT solver still needs less than 7 seconds for any graph of any size and any number of $\mathbb{G}_{\bigcirc}$. We note that the time it takes to compute the clauses for the 2-SAT instance is not included in the running time of the backtracking algorithm, although it would merely add a few seconds onto an already relatively high amount of time. In terms of running time, the SAT solver clearly performs better than the backtracking algorithm.

5.3.3. Coloring Circle Graphs using c^*

We were interested in seeing how the effects of our counterexample played out in practice, that is, how many consistent auxiliary coloring functions c^* can actually be used to construct a 3-coloring. In Figure 5.9 the percentages of sample circle graphs is given for which the 3-coloring constructed using c^* was correct. Unfortunately, not many of our test graphs were colored correctly. About 8% of graphs were colored correctly with c^* computed by the SAT solver, and nearly 12% with c^* computed using the backtracking algorithm. From

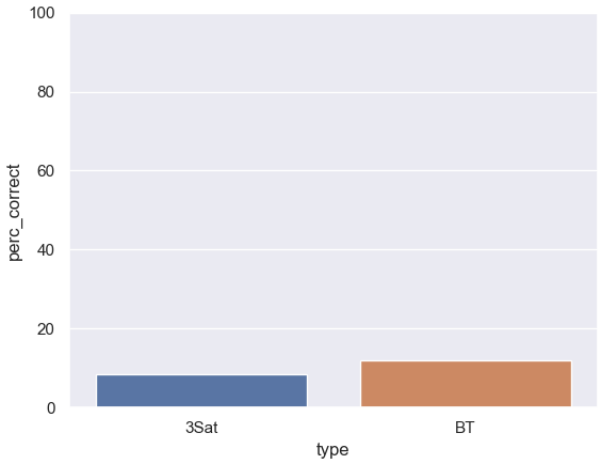


Figure 5.9.: Percentage of graphs for which the consistent auxiliary coloring function c^* was able to construct a 3-coloring. The left bar shows that number for c^* computed with the SAT solver, the right bar shows the number for c^* computed by the backtracking algorithm.

Table 5.1.: Number of vertices of graphs that were correctly colored using c^*

	maximum #vertices	average #vertices	median #vertices
c^* SAT solver	48	10.724	8
c^* backtracking	57	13.376	15

Tables 5.1 and 5.2 we see that the graphs that were colored correctly are relatively small compared to the rest of our samples and often do not contain many $G_{\bigcirc} \in \mathbb{G}_{\bigcirc}$.

Table 5.2.: Number of important subgraphs $G_{\bigcirc} \in \mathbb{G}_{\bigcirc}$ of graphs that were correctly colored using c^*

	maximum $\lvert\mathbb{G}_{\bigcirc}\rvert$	average $\lvert\mathbb{G}_{\bigcirc}\rvert$	median $\lvert\mathbb{G}_{\bigcirc}\rvert$
c^* SAT solver	6	0.699	0
c^* backtracking	8	0.5	0

The difference in performance between the consistent auxiliary coloring functions computed by the SAT solver and the backtracking algorithm suggests that the values for c^* computed by the SAT solver contain more errors in the sense that c^* contains contradicting or unrealizable values, than those computed by the backtracking algorithm.

6. Conclusion

We have shown that the existence of a consistent auxiliary coloring function c^* for a circle graph G is not equivalent to G being 3-colorable. That is, a consistent auxiliary coloring function c^* is not generally realizable for G. With this we have disproven a lemma that is crucial to Unger's result on the 3-coloring of circle graphs. We discussed the backtracking algorithm and the potential number of leaves of the corresponding backtracking tree. While we can neither prove nor disprove Unger's claim that the number of leaves is in $O(\log n)$, we have given our thoughts and arguments on why we doubt that this holds. In our evaluation we investigated the running time for the backtracking algorithm and the SAT solver for computing a consistent auxiliary coloring function c^*. We have seen that in practice, while the SAT solver is significantly faster at computing values for c^*, the number of graphs that were correctly 3-colored using c^* was higher for consistent auxiliary coloring functions computed with backtracking. Our evaluation also showed that the number of leaves we visit until we find a solution can be relatively small, even for circle graphs with many nodes and important subgraphs $G_{\bigcirc} \in \mathbb{G}_{\bigcirc}$. It also suggests a correlation between the number of values for c^* that are defined for the vertices of important subgraphs in $\mathbb{G}_{\bigcirc}$ and not for any other subgraph, i.e. c^* values that are relevant only to $\mathbb{G}_{\bigcirc}$, and the number of backtracking leaves visited until a solution is found. This to some degree coincides with the claims Unger stated in regard to clear implications.

An open problem we gave in Chapter 3 is to see whether there is a way to add more properties or other forms of restrictions to c^* such that our presented counterexample becomes obsolete. One possible approach might be to find a way to prohibit c^* to have contradicting or non-realizable values with respect to the entire circle graph G. Another open problem is to specify and finalize the ideas on why the backtracking tree might have a tighter upper bound on the number of leaves than $2^{O(n^2)}$ or to give a lower bound. From our evaluation it seems likely that there is some correlation between the size of the graph and the number of values of c^* we have to try until we find a solution. Studying and formalizing this potential connection is also left as an open problem. In general, the 3-coloring problem should be considered an open problem for circle graphs still.

Bibliography

[1] K. Appel and W. Haken. The solution of the four-color-map problem. *Scientific American*, 237(4):108–121, 1977.

[2] B. Aspvall, M. F. Plass, and R. E. Tarjan. A linear-time algorithm for testing the truth of certain quantified boolean formulas. *Information Processing Letters*, 8(3):121–123, mar 1979.

[3] M. Avriel, M. Penn, and N. Shpirer. Container ship stowage problem: complexity and connection to the coloring of circle graphs. *Discrete Applied Mathematics*, 103(1-3):271–279, jul 2000.

[4] R. Beigel and D. Eppstein. 3-coloring in time $O(1.3289^n)$. 2000.

[5] S. Benzer. ON THE TOPOLOGY OF THE GENETIC FINE STRUCTURE. *Proceedings of the National Academy of Sciences*, 45(11):1607–1620, nov 1959.

[6] F. Bernhart and P. C. Kainen. The book thickness of a graph. *Journal of Combinatorial Theory, Series B*, 27(3):320–331, dec 1979.

[7] K. S. Booth and G. S. Lueker. Testing for the consecutive ones property, interval graphs, and graph planarity using PQ-tree algorithms. *Journal of Computer and System Sciences*, 13(3):335–379, dec 1976.

[8] M. C. Carlisle and E. L. Lloyd. On the k-coloring of intervals. *Discrete Applied Mathematics*, 59(3):225–235, may 1995.

[9] F. R. K. Chung, F. T. Leighton, and A. L. Rosenberg. Embedding graphs in books: A layout problem with applications to VLSI design. *SIAM Journal on Algebraic Discrete Methods*, 8(1):33–58, jan 1987.

[10] A. Cimatti, A. Griggio, B. Schaafsma, and R. Sebastiani. The MathSAT5 SMT Solver. In N. Piterman and S. Smolka, editors, *Proceedings of TACAS*, volume 7795 of *LNCS*. Springer, 2013.

[11] V. Conitzer, J. Derryberry, and T. Sandholm. Combinatorial auctions with structured item graphs. In *AAAI*, volume 4, pages 212–218, 2004.

[12] C. H. Coombs and J. E. K. Smith. On the detection of structure in attitudes and developmental processes. *Psychological Review*, 80(5):337–351, sep 1973.

[13] D. G. Corneil, S. Olariu, and L. Stewart. The LBFS structure and recognition of interval graphs. *SIAM Journal on Discrete Mathematics*, 23(4):1905–1953, jan 2010.

[14] G. Durán, L. N. Grippo, and M. D. Safe. Structural results on circular-arc graphs and circle graphs: A survey and the main open problems. *Discrete Applied Mathematics*, 164:427–443, feb 2014.

[15] S. Even and A. Itai. QUEUES, STACKS AND GRAPHS. In *Theory of Machines and Computations*, pages 71–86. Elsevier, 1971.

[16] F. V. Fomin, S. Gaspers, and S. Saurabh. Improved exact algorithms for counting 3- and 4-colorings. In *Lecture Notes in Computer Science*, pages 65–74. Springer Berlin Heidelberg.

[17] D. Fulkerson and O. Gross. Incidence matrices and interval graphs. *Pacific Journal of Mathematics*, 15(3):835–855, sep 1965.

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

[19] F. Gavril. Algorithms on circular-arc graphs. *Networks*, 4(4):357–369, 1974.

[20] J. Geelen and S. il Oum. Circle graph obstructions under pivoting. *Journal of Graph Theory*, 61(1):1–11, may 2009.

[21] P. C. Gilmore and A. J. Hoffman. A characterization of comparability graphs and of interval graphs. *Canadian Journal of Mathematics*, 16:539–548, 1964.

[22] E. Gioan, C. Paul, M. Tedder, and D. Corneil. Practical and efficient circle graph recognition. *Algorithmica*, 69(4):759–788, mar 2013.

[23] M. Grötschel, L. Lovász, and A. Schrijver. *Geometric Algorithms and Combinatorial Optimization*. Springer Berlin Heidelberg, 1993.

[24] F. Guthrie. 9. note on the colouring of maps. *Proceedings of the Royal Society of Edinburgh*, 10:727–728, 1880.

[25] M. Habib, R. McConnell, C. Paul, and L. Viennot. Lex-BFS and partition refinement, with applications to transitive orientation, interval graph recognition and consecutive ones testing. *Theoretical Computer Science*, 234(1-2):59–84, mar 2000.

[26] H. Hadwiger. *Combinatorial geometry in the plane*. 2015.

[27] G. HAJOS. Ueber eine art von graphen. *Internationale Mathematische Nachrichten*, page 65, 1957.

[28] F. Harary. A graph theoretic approach to similarity relations. *Psychometrika*, 29(2):143–151, jun 1964.

[29] H. Kaplan and Y. Nussbaum. A simpler linear-time recognition of circular-arc graphs. *Algorithmica*, 61(3):694–737, aug 2010.

[30] R. M. Karp. Reducibility among combinatorial problems. In *Complexity of Computer Computations*, pages 85–103. Springer US, 1972.

[31] J. Keil. The complexity of domination problems in circle graphs. *Discrete Applied Mathematics*, 42(1):51–63, feb 1993.

[32] D. Kendall. Incidence matrices, interval graphs and seriation in archeology. *Pacific Journal of Mathematics*, 28(3):565–570, mar 1969.

[33] V. Klee. What are the intersection graphs of arcs in a circle? *The American Mathematical Monthly*, 76(7):810–813, aug 1969.

[34] T. KLOKS. TREEWIDTH OF CIRCLE GRAPHS. *International Journal of Foundations of Computer Science*, 07(02):111–120, jun 1996.

[35] C. Lekkerkerker and J. Boland. Representation of a finite graph by a set of intervals on the real line. *Fundamenta Mathematicae*, 51(1):45–64, 1962.

[36] R. M. McConnell. Linear-time recognition of circular-arc graphs. *Algorithmica*, 37(2):93–147, jul 2003.

[37] W. Naji. Reconnaissance des graphes de cordes. *Discrete Mathematics*, 54(3):329–337, may 1985.

[38] N. Nash and D. Gregg. An output sensitive algorithm for computing a maximum independent set of a circle graph. *Information Processing Letters*, 110(16):630–634, jul 2010.

[39] S. Olariu. An optimal greedy heuristic to color interval graphs. *Information Processing Letters*, 37(1):21–25, jan 1991.

[40] J. B. Orlin, M. A. Bonuccelli, and D. P. Bovet. An $O(n^2)$ algorithm for coloring proper circular arc graphs. *SIAM Journal on Algebraic Discrete Methods*, 2(2):88–93, jun 1981.

[41] F. S. Roberts. *Graph theory and its applications to problems of society*. SIAM, 1978.

[42] F. S. Roberts. INDIFFERENCE AND SERIATION. *Annals of the New York Academy of Sciences*, 328(1 Topics in Gra):173–182, jun 1979.

[43] N. Robertson, D. Sanders, P. Seymour, and R. Thomas. The four-colour theorem. *Journal of Combinatorial Theory, Series B*, 70(1):2–44, may 1997.

[44] J. Spinrad. Recognition of circle graphs. *Journal of Algorithms*, 16(2):264–282, mar 1994.

[45] S. Stefanakos and T. Erlebach. Routing in all-optical ring networks revisited. In *Proceedings. ISCC 2004. Ninth International Symposium on Computers And Communications (IEEE Cat. No.04TH8769)*. IEEE.

[46] A. Tiskin. Fast distance multiplication of unit-monge matrices. *Algorithmica*, 71(4):859–888, sep 2013.

[47] A. Tucker. Matrix characterizations of circular-arc graphs. *Pacific Journal of Mathematics*, 39(2):535–545, nov 1971.

[48] A. Tucker. An efficient test for circular-arc graphs. *SIAM Journal on Computing*, 9(1):1–24, feb 1980.

[49] A. C. TUCKER. *Two characterizations of proper circular-arc graphs*. Stanford University, 1969.

[50] W. Unger. On the k-colouring of circle-graphs. In *STACS 88*, pages 61–72. Springer-Verlag.

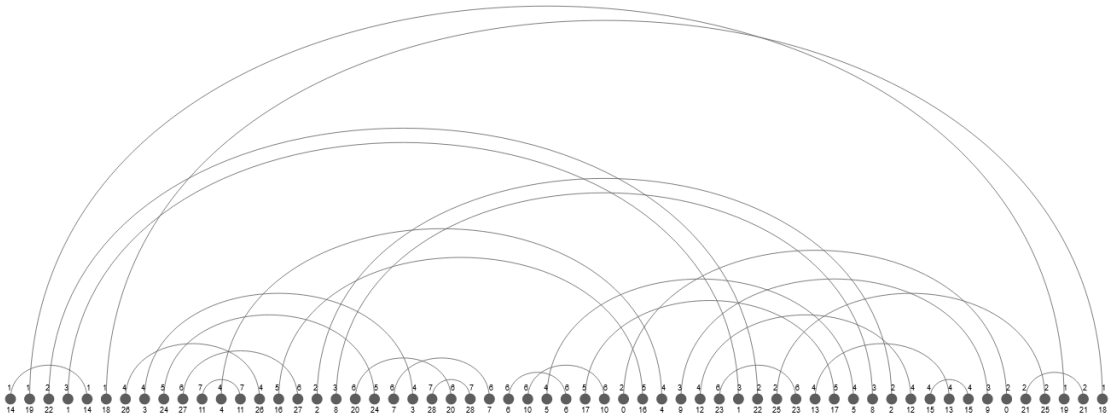
[51] W. Unger. *Färbung von Kreissehnengraphen*. na, 1990.

[52] W. Unger. The complexity of colouring circle graphs. In *STACS 92*, pages 389–400. Springer Berlin Heidelberg, 1992.

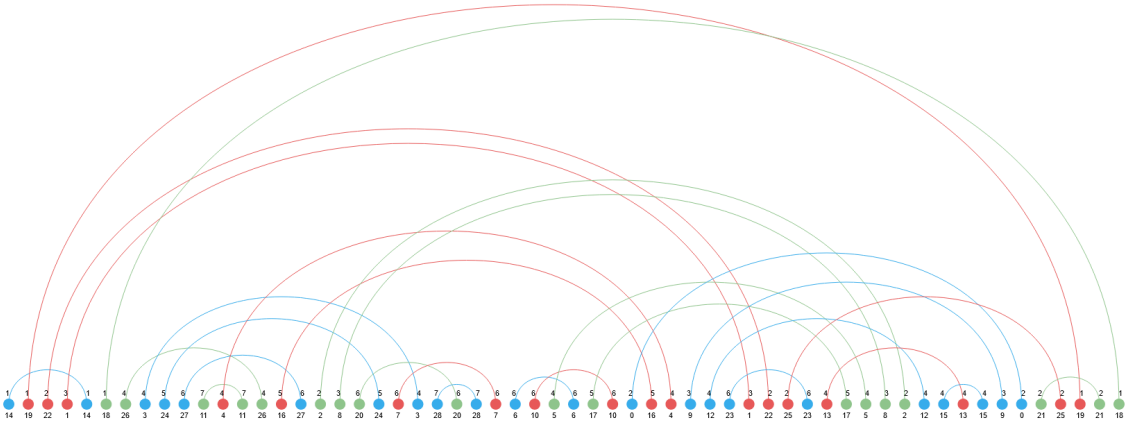
[53] R. Wilson. *Four Colors Suffice: How the Map Problem Was Solved-Revised Color Edition*, volume 30. Princeton university press, 2013.

Appendix

A. Sample Graphs Examples

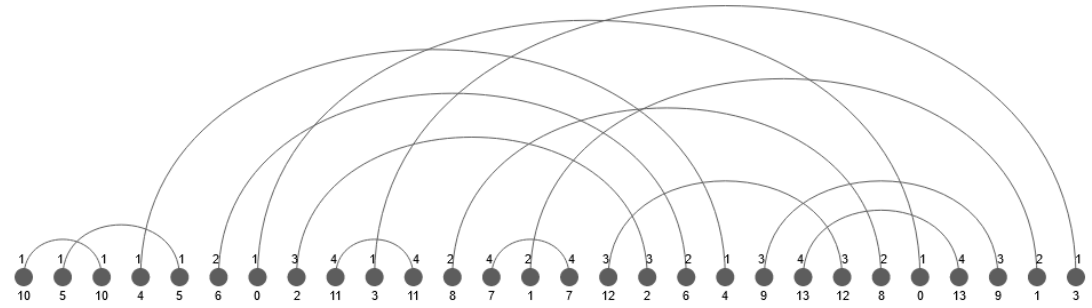


(a)

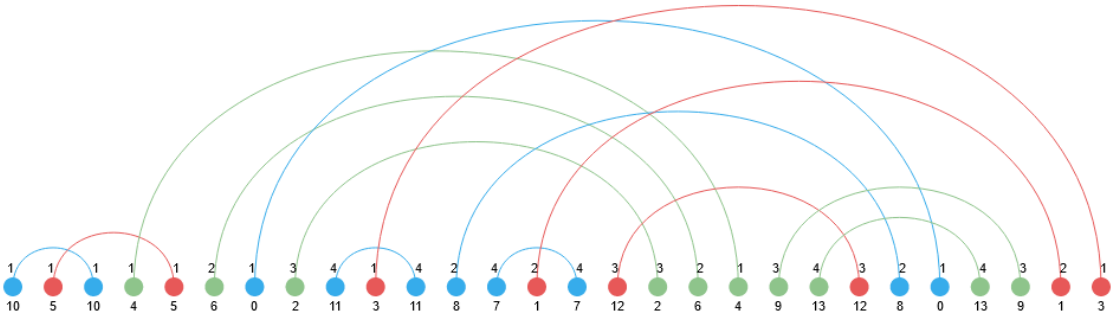


(b)

Figure A.1.: (a) A sample graph of category C (b) Coloring computed by our application



(a)



(b)

Figure A.2.: (a) A sample graph of category RC (b) Coloring computed by our application