Journal of Graph Algorithms and Applications
http://jgaa.info/ vol. 28, no. 1, pp. 389–402 (2024)
DOI: 10.7155/jgaa.v28i1.2991

On 3-Coloring Circle Graphs

Patricia Bachmann[1] *Ignaz Rutter*[1] *Peter Stumpf*[2]

[1]University of Passau, Passau, Germany
[2]Charles University, Prague, Czech Republic

Submitted: November 2023 Accepted: September 2024 Published: October 2024

Article type: Regular Paper Communicated by: A. Symvonis

Abstract. Given a graph G with a fixed vertex order $\prec$, one obtains a circle graph H whose vertices are the edges of G and where two such edges are adjacent if and only if their endpoints are pairwise distinct and alternate in $\prec$. Therefore, the problem of determining whether G has a k-page book embedding with spine order $\prec$ is equivalent to deciding whether H can be colored with k colors. Finding a k-coloring for a circle graph is known to be NP-complete for $k \geq 4$ [9] and trivial for $k \leq 2$. For $k = 3$, Unger (1992) claims an efficient algorithm that finds a 3-coloring in $O(n \log n)$ time, if it exists. Given a circle graph H, Unger's algorithm (1) constructs a 3-SAT formula Φ that is satisfiable if and only if H admits a 3-coloring and (2) solves Φ by a backtracking strategy that relies on the structure imposed by the circle graph. However, the extended abstract misses several details and Unger refers to his PhD thesis (in German) for details.

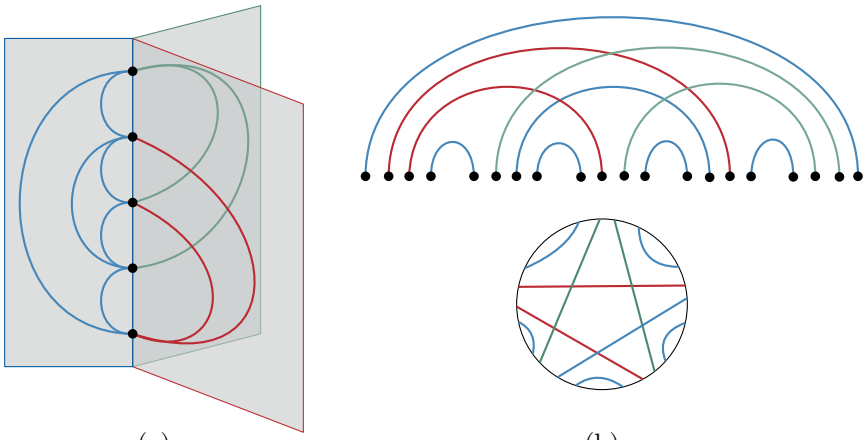
In this paper we argue that Unger's algorithm for 3-coloring circle graphs is not correct and that 3-coloring circle graphs should be considered as an open problem. We show that step (1) of Unger's algorithm is incorrect by exhibiting a circle graph and its representation whose formula Φ is satisfiable but that is not 3-colorable. We further show that Unger's backtracking strategy for solving Φ in step (2) may produce incorrect results and give empirical evidence that it exhibits a runtime behaviour that is not consistent with the claimed running time.

1 Introduction

Let $G = (V, E)$ be a graph. A *k-page book embedding* of G is a total order $\prec$ of V and a partition of E into k sets $E_1, \ldots, E_k$, called *pages* such that no page E_i contains two edges $\{u_1, v_1\}$, $\{u_2, v_2\}$ with $v_1 \prec v_2 \prec u_1 \prec u_2$. The *page number* (also called *stack number*) of a graph is the smallest k such that G admits a k-page booking embedding.

Funded by the Deutsche Forschungsgemeinschaft (German Research Foundation, DFG) under grant RU-1903/3-1.

E-mail addresses: bachmanp@fim.uni-passau.de (Patricia Bachmann) rutter@fim.uni-passau.de (Ignaz Rutter) stumpf@fim.uni-passau.de (Peter Stumpf)



(a) (b)

Figure 1: (a) Book embedding for K_5. (b) Chord diagram of the corresponding circle graph H.

Book embeddings are a central element to graph drawing. They have been studied in the context of VSLI design [5], arc diagrams [12] and circular layouts [2] as well as clustered planarity [8] and simultaneous embedding [1]. There is a plethora of results that aim at bounding the page numbers of various graph classes. For example, the page number of planar graphs is 4 [4, 13, 14] and the page number of 1-planar graphs is at most 39 [3].

Since computing the page number is NP-complete [5], often additional restrictions are imposed. One such restriction is to find a k-page book embedding with a fixed order $\prec$. This problem is closely related to the k-coloring problem on circle graphs. A *k-coloring* of a graph G is a function $\mathrm{col} : V(G) \to \{1, \ldots, k\}$ such that $\mathrm{col}(u) \neq \mathrm{col}(v)$ for every $\{u, v\} \in E(G)$. A *circle graph* is an undirected graph H that has an intersection representation with chords of a circle. More precisely, in a *chord diagram* of H we represent each vertex $v \in V(H)$ by a chord C_v of the unit circle such that two chords C_u and C_v intersect if and only if $\{u, v\} \in E(H)$.

Finding a k-page book embedding of a graph $G = (V, E)$ with a fixed vertex order $\prec$ is equivalent to solving the k-coloring problem for circle graphs. To see this, construct a graph H such that $V(H) = E$ and two edges in $V(H)$ are adjacent in H if and only if their endpoints alternate in $\prec$. Then two edges of G can be in the same page of a book embedding with order $\prec$ if and only if they are not adjacent in H. Therefore we obtain a bijection between the k-page book embeddings with order $\prec$ of G and the k-colorings of H. It is further readily seen that H can be represented by a chord diagram; see Figure 1 for an illustration.

The k-coloring problem for circle graphs is known to be NP-complete for $k \geq 4$ [9] and efficiently solvable for $k \leq 2$. The case $k = 3$ remained as an open problem until Unger [11] claimed it to be solvable in polynomial time. Unfortunately, in the publication many details and proofs are missing and no journal version followed. Instead, Unger refers to his PhD thesis [10] for a full version, which is written in German and not available online[1]. This bad state of affairs has been pointed out by David Eppstein in a blog post, where he writes that the problem should be considered open [7], as well as by Dujmović and Wood [6].

In this paper we present Unger's ideas for an efficient algorithm for the 3-coloring problem

[1]The authors of this paper had access to a copy of the thesis kept at the university library of the Technical University Munich.

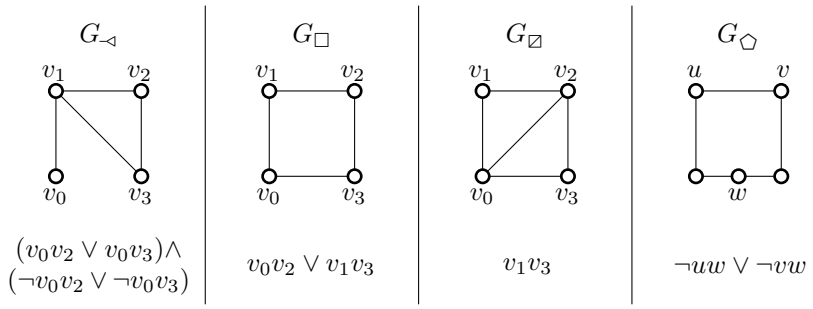


Figure 2: Important subgraphs and the clauses they contribute to Φ.

on circle graphs and show, using both counterexamples and empirical results, why 3-coloring circle graphs and therefore the 3-page book embedding problem should indeed be considered open problems.

Throughout this work let $G = (V, E)$ be a circle graph, for which we want to decide the existence of a 3-coloring. We assume without loss of generality that G is connected and contains no induced K_4.

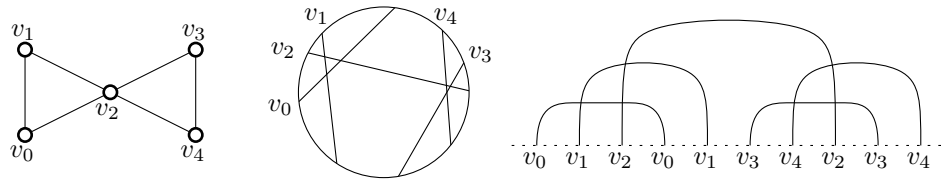
2 Unger's 3-Coloring Algorithm

For a graph H let $\mathcal{P}(H) = \{\{u, v\} \in \binom{V(H)}{2} \setminus E(H) \mid N(u) \cap N(v) \neq \emptyset\}$ be the pairs of non-adjacent vertices that have a common neighbor and let $\mathcal{D} \subseteq \mathcal{P}(H)$. An *auxiliary coloring function for* $\mathcal{D}$ is a function $c_{\text{aux}} \colon \mathcal{D} \to \{\texttt{True}, \texttt{False}\}$. An auxiliary coloring function is *realizable* if there exists a 3-coloring col of H such that for each pair $\{x, y\} \in \mathcal{D}$ we have $\text{col}(x) = \text{col}(y) \iff c_{\text{aux}}(\{x, y\}) = \texttt{True}$.

Let now $G = (V, E)$ be a graph and let $\mathcal{G}$ be a set of induced subgraphs of G and let $\mathcal{P}(\mathcal{G}) = \bigcup_{H \in \mathcal{G}} \mathcal{P}(H)$. An *auxiliary coloring function of* G *with respect to* $\mathcal{G}$ is an auxiliary coloring function for $\mathcal{P}(\mathcal{G})$. Such a function is called *consistent*if its restriction $c_{\text{aux}}|_{\mathcal{P}(H)}$ is realizable for each $H \in \mathcal{G}$. As an example, consider the graph $G_{\dashv}$ in Figure 2. An auxiliary coloring function c_{aux} is defined for the pairs $\{v_0, v_2\}$ and $\{v_0, v_3\}$. Then c_{aux} is realizable if and only if $c_{\text{aux}}(\{v_0, v_2\}) \neq c_{\text{aux}}(\{v_0, v_3\})$. In particular, c_{aux} is not realizable if $c_{\text{aux}}(\{v_0, v_2\}) = c_{\text{aux}}(\{v_0, v_3\})$, since v_2 and v_3 are adjacent and one of them must have the same color as v_0.

In his approach, Unger constructs for a given circle graph G a family $\mathcal{G}$ of induced subgraphs, which he calls *important subgraphs*, such that (i) an auxiliary coloring function c_{aux} with respect to $\mathcal{G}$ is consistent if and only if it is realizable, (ii) the existence of a consistent auxiliary coloring function can be expressed by a 3-SAT formula Φ that (iii) can be solved efficiently with a backtracking algorithm.

Let $\mathcal{G}'$ be the family of induced subgraphs of G that are isomorphic to one of the graphs $G_{\dashv}$, $G_{\square}$, $G_{\pentagon}$, $G_{\boxslash}$ from Figure 2 or to a cycle C_k with $k \geq 6$. For each $H \in \mathcal{G}'$ there is a formula $\Phi(H)$ whose satisfying truth assignments are the realizable auxiliary coloring functions for H. For the three graphs $G_{\dashv}$, $G_{\square}$ and $G_{\boxslash}$, the corresponding formula $\Phi(H)$ is in fact a 2-SAT formula as shown in Figure 2, whereas for C_k with $k \geq 5$, the formula $\Phi(H)$ given by Unger is a 3-SAT formula of size linear in k, which uses some additional variables. For $G_{\pentagon}$, i.e. cycles with $k = 5$, Unger additionally uses the 2-SAT clauses shown in Figure 2. Then the existence of a consistent auxiliary coloring function for $\mathcal{G}'$ can be expressed by the formula $\Phi(\mathcal{G}') = \bigwedge_{H \in \mathcal{G}'} \Phi(H)$.



(a) (b) (c)

Figure 3: (a) An undirected graph G. (b) Representation of G of chords on a circle. (c) Alternative circle graph representation for G.

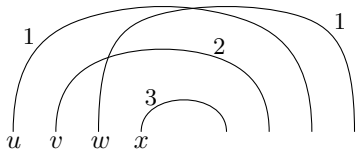


Figure 4: The chords encased by u are v and x; u directly encases only v, while v and w directly encase x.

The family $\mathcal{G}'$ is, however, still too large. For example, it may contain an exponential number of cycles. Therefore, Unger restricts his important subgraphs $\mathcal{G}$ to a subset of $\mathcal{G}'$ that is defined according to a chord diagram of G. To this end, take a chord diagram of G and consider it to be cut open and rolled out such that the chords form arcs over a straight line. This is illustrated in Figure 3. In what follows we identify each vertex v with its chord C_v.

A chord u *encases* a chord v if the endpoints of v lie between the endpoints of u. Further, u *directly encases* v if it encases v and there is no vertex $w \neq u, v$ such that u encases w and w encases v; see Figure 4 for an example.

From these definitions we get the *levels* of a circle graph G. Let E_v be the set of vertices directly encasing some $v \in V(G)$. The *level of* v is recursively defined as follows:

$$l(v) = \begin{cases} 1 & \text{if } E_v = \emptyset \\ \max\{l(u) \mid u \in E_v\} + 1 & \text{otherwise} \end{cases}.$$

The set $\text{level}_G(i)$ then contains all vertices $v \in V(G)$ with $l(v) = i$.

The set of important subgraphs $\mathcal{G}$ consists of those graphs $H \in \mathcal{G}'$ where if H is isomorphic to $G_{\pentagon}$ or to C_k with $k \geq 6$, then all vertices of H belong to two adjacent levels, and otherwise for each pair $\{u, v\} \in \mathcal{P}(H)$ the vertices u and v are either on the same level or one directly encases the other. Unger's algorithm relies on two claims [11, p.394 ff.]:

(1) The graph G is 3-colorable if and only if there exists a consistent auxiliary coloring function with respect to $\mathcal{G}$.

(2) The formula $\Phi(\mathcal{G})$ can be solved efficiently by a backtracking algorithm whose search tree has $O(\log n)$ leaves.

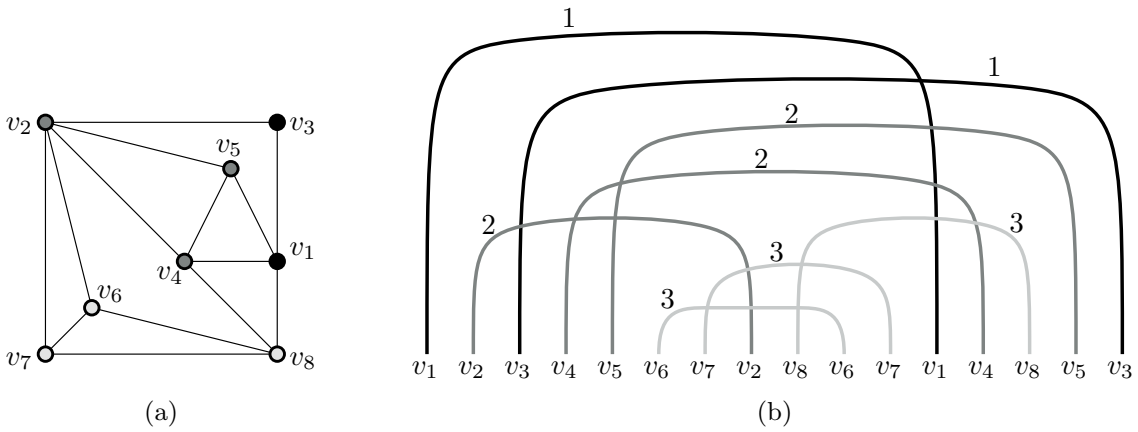


Figure 5: Counterexample to claim (1). Lighter shade indicates a higher level. (a) The graph G. (b) A chord diagram of G.

3 The Counterexample(s)

We show that Unger's claim (1) is false by giving a counterexample. Let G be the graph given in Figure 5a. Observe that G is a circle graph as witnessed by the chord diagram in Figure 5b.

We first show that G is not 3-colorable. Namely, the two subgraphs induced by $\{v_1, v_2, v_4, v_5\}$ and $\{v_2, v_6, v_7, v_8\}$ imply $\mathrm{col}(v_1) = \mathrm{col}(v_2)$ and $\mathrm{col}(v_2) = \mathrm{col}(v_8)$, respectively. However v_1 and v_8 are adjacent.

On the other hand, we show that for the family $\mathcal{G}$ of important subgraphs of G with respect to the chord diagram in Figure 5, the formula $\Phi(\mathcal{G})$ is satisfiable. We first give the important subgraphs of G and then construct the corresponding formula. In Figure 5a the vertices are colored according to their levels, where lighter colors indicate higher levels. It is not hard to see that G contains no induced cycle isomorphic to $G_{\pentagon}$ or to C_k with $k \geq 6$ that is contained in two adjacent levels. It hence suffices to find the important subgraphs isomorphic to $G_{\triangleleft}$,$G_{\square}$, and $G_{\boxslash}$.

Observe that each of the pairs $\{v_2, v_8\}$ and $\{v_3, v_8\}$ neither lies on the same level nor does one of the vertices directly encase the other. Hence, no important subgraph contains both vertices of these pairs. We start with $G_{\square}$. To this end, we list all chordless 4-cycles and then show which of them form a $G_{\square}$. The chordless cycle of length 4 are induced by $\{v_1, v_2, v_3, v_4\}$, $\{v_1, v_2, v_3, v_5\}$, $\{v_2, v_4, v_6, v_8\}$ and $\{v_2, v_4, v_7, v_8\}$. Since v_1 directly encases v_2 and v_3 directly encases $\{v_4, v_5\}$, $\{v_1, v_2, v_3, v_4\}$ and $\{v_1, v_2, v_3, v_5\}$ each form a $G_{\square}$. From the reasoning given above, we know that no important subgraph contains both v_2 and v_8, hence it follows that $\{v_2, v_4, v_6, v_8\}$ and $\{v_2, v_4, v_7, v_8\}$ induce no $G_{\square}$. The list of all $G_{\square}$ is given in Table 1a.

Next, we consider the important subgraph $G_{\boxslash}$. This subgraph consists of two 3-cliques sharing a common edge. The information we want to infer from this important subgraph for c_{aux} concerns the two non-adjacent vertices that are either in the same level or one of them directly encases the other. We consider each level and list all $G_{\boxslash}$ formed with chords in the next level, respectively those that are directly encased, starting with the chords in $\mathrm{level}_G(1)$. We have that v_1 directly encases v_2 and induces a $G_{\boxslash}$ with $\{v_4, v_5\} \subset N(v_1) \cap N(v_2)$. Since $N(v_1) \cap N(v_6) = N(v_1) \cap N(v_7) = \{v_8\}$, there is no induced $G_{\boxslash}$ that contains v_1 and v_6, respectively v_7. Since v_3 is not part of any 3-clique, the is also no induced $G_{\boxslash}$ containing v_3. For $\mathrm{level}_G(2)$ we have that its chords form a clique, hence there is

Table 1: Important subgraphs and the corresponding clauses for each of (a) $G_{\square}$, (b) $G_{\boxslash}$, and (c) $G_{\dashv\lhd}$. For the satisfying variable assignment from Table 2 all true literals are underlined.

Important Subgraph $G_{\square}$	Clauses
$\{v_1, v_2, v_3, v_4\}$	$\underline{v_1 v_2} \vee \underline{v_3 v_4}$
$\{v_1, v_2, v_3, v_5\}$	$\underline{v_1 v_2} \vee v_3 v_5$

(a)

Important Subgraph $G_{\boxslash}$	Clauses
$\{v_1, v_2, v_4, v_5\}$	$\underline{v_1 v_2}$
$\{v_1, v_4, v_5, v_8\}$	$\underline{v_5 v_8}$

(b)

Important Subgraph $G_{\dashv\lhd}$	Clauses
$\{v_1, v_6, v_7, v_8\}$	$(v_1 v_6 \vee \underline{v_1 v_7}) \wedge (\neg \underline{v_1 v_6} \vee \neg\ v_1 v_7)$
$\{v_1, v_3, v_4, v_5\}$	$(\underline{v_3 v_4} \vee v_3 v_5) \wedge (\neg v_3 v_4 \vee \underline{\neg v_3 v_5})$
$\{v_2, v_3, v_4, v_5\}$	$(\underline{v_3 v_4} \vee v_3 v_5) \wedge (\neg v_3 v_4 \vee \underline{\neg v_3 v_5})$
$\{v_2, v_4, v_6, v_7\}$	$(\underline{v_4 v_6} \vee v_4 v_7) \wedge (\neg v_4 v_6 \vee \underline{\neg v_4 v_7})$
$\{v_2, v_5, v_6, v_7\}$	$(v_5 v_6 \vee \underline{v_5 v_7}) \wedge (\underline{\neg v_5 v_6} \vee \neg v_5 v_7)$
$\{v_2, v_4, v_5, v_6\}$	$(\underline{v_4 v_6} \vee v_5 v_6) \wedge (\neg v_4 v_6 \vee \underline{\neg v_5 v_6})$
$\{v_2, v_4, v_5, v_7\}$	$(v_4 v_7 \vee \underline{v_5 v_7}) \wedge (\underline{\neg v_4 v_7} \vee \neg v_5 v_7)$
$\{v_4, v_6, v_7, v_8\}$	$(\underline{v_4 v_6} \vee v_4 v_7) \wedge (\neg v_4 v_6 \vee \underline{\neg v_4 v_7})$

(c)

no induced $G_{\boxslash}$ with non-crossing chords in this level. Since v_2 does not encase any chords, it is not contained in any induced $G_{\boxslash}$. For v_4 we see that $N(v_4) \cap N(v_6) = N(v_4) \cap N(v_7) = \{v_2, v_8\} \notin E(G)$ and therefore there is no induced $G_{\boxslash}$ containing v_4 and v_6, respectively v_7. A similar argument holds for v_5 and v_6, respectively v_7, namely $N(v_5) \cap N(v_6) = N(v_5) \cap N(v_7) = \{v_2\}$, therefore v_5 also does not induce a $G_{\boxslash}$ with v_6, respectively v_7. For v_8, however, we have $N(v_5) \cap N(v_8) = \{v_1, v_4\} \in E(G)$, therefore $\{v_1, v_4, v_5, v_8\}$ induces a $G_{\boxslash}$. The chords in $\text{level}_G(3)$ do not induce any $G_{\boxslash}$, since $|\text{level}_G(3)| < 4$. Table 1b lists all $G_{\boxslash}$.

Finally, we give the important subgraphs $G_{\dashv\lhd}$. We first consider chords within one level. Since each level contains less than four vertices each, this graph contains no $G_{\dashv\lhd}$ with all chords on the same level. Therefore, we consider $G_{\dashv\lhd}$ with directly encased chords. For $\text{level}_G(1)$ we have that v_1 directly encases $\{v_2, v_6, v_7\}$ while v_3 directly encases $\{v_4, v_5\}$. We have $N(v_1) \cap N(v_2) = \{v_3, v_4, v_5\}$ and since there is no 3-clique containing v_2 and two vertices of $\{v_3, v_4, v_5\}$ such that v_1 is adjacent to exactly one vertex of the 3-clique, there is no induced $G_{\dashv\lhd}$ containing v_1 and v_2. Further, we have $N(v_1) \cap N(v_6) \cap N(v_7) = \{v_8\}$, so $\{v_1, v_6, v_7, v_8\}$ induces a $G_{\dashv\lhd}$. For v_3 we have that $N(v_3) \cap N(v_4) \cap N(v_5) = \{v_1, v_2\}$, therefore $\{v_1, v_3, v_4, v_5\}$ and $\{v_2, v_3, v_4, v_5\}$ each induce a $G_{\dashv\lhd}$ that produce the same clauses for $\Phi(\mathcal{G})$. For $\text{level}_G(2)$ we have that v_2 encases no chords, v_4 directly encases $\{v_6, v_7\}$ and v_5 directly encases $\{v_6, v_7, v_8\}$. We have that $N(v_4) \cap N(v_5) \cap N(v_6) \cap N(v_7) = \{v_2\}$, therefore $\{v_2, v_4, v_5, v_6, v_7\}$ induces four $G_{\dashv\lhd}$, namely $\{v_2, v_4, v_6, v_7\}$, $\{v_2, v_5, v_6, v_7\}$, $\{v_2, v_4, v_5, v_6\}$ and $\{v_2, v_4, v_5, v_7\}$. For v_4 we also have $v_8 \in N(v_4) \cap N(v_6) \cap N(v_7)$, therefore $\{v_4, v_6, v_7, v_8\}$ also induces a $G_{\dashv\lhd}$. We note that the clauses for $\Phi(\mathcal{G})$ produced by $\{v_4, v_6, v_7, v_8\}$ is also gathered from $\{v_2, v_4, v_6, v_7\}$. Since for v_8 there is no 3-clique containing it such that v_5 is adjacent to exactly one vertex of that 3-clique, there is no induced $G_{\dashv\lhd}$ containing v_5 and v_8. Note that $\{v_2, v_4, v_5, v_8\}$ does not induce a $G_{\dashv\lhd}$, since no important subgraph contains both v_2 and v_8, as argued before. Since the chords in $\text{level}_G(3)$ do not encase any chords they do not induce any additional $G_{\dashv\lhd}$. All $G_{\dashv\lhd}$ are given in Table 1c.

Table 1 shows the clauses of the formula $\Phi := \Phi(\mathcal{G})$. Finally, Table 2 gives a satisfying truth

Table 2: Satisfying variable assignment for Φ.

True	v_1v_2, v_1v_7, v_2v_5, v_3v_4, v_4v_6, v_5v_7, v_5v_8
False	v_1v_6, v_4v_7, v_5v_6, v_3v_5

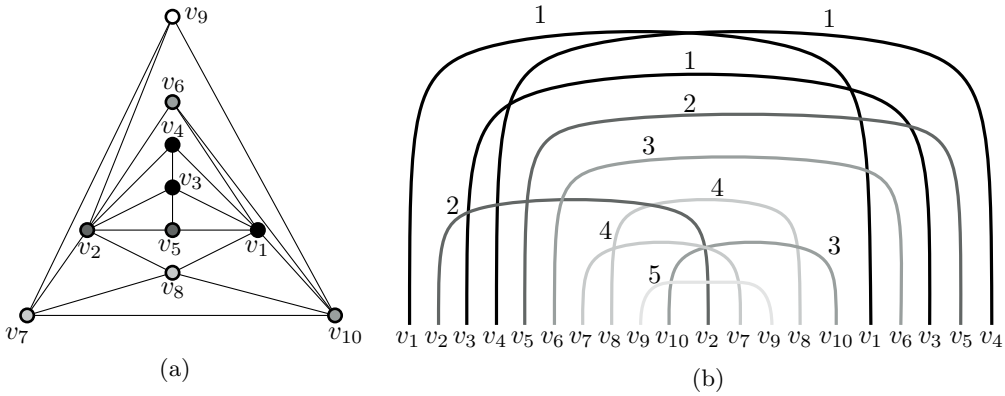


Figure 6: Counterexample for the thesis version of important subgraphs.

assignment for Φ. The underlined literals in Table 1 are those satisfied by that truth assignment. Since every clause contains a satisfied literal, Φ is satisfiable and the truth assignment defines a consistent auxiliary coloring function for $\mathcal{G}$. However, as G is not 3-colorable, this contradicts Unger's claim (1).

Counterexample to the Version of Unger's Thesis We note that the definition of important subgraphs subtly differs between the extended abstract [11] and Unger's PhD thesis [10]. Namely, in the thesis, important subgraphs are not defined only via direct encasing but for some types their vertices belong to (at most) two adjacent levels. While the counterexample above refutes the claim from the extended abstract, the graph G in Figure 6 refutes the analogous claim from Unger's thesis [10]. The graph is clearly not 3-colorable, since in any 3-coloring col we have $\text{col}(v_1) = \text{col}(v_2)$ and $\text{col}(v_1) = \text{col}(v_7)$ due to the subgraphs induced by $\{v_1, v_2, v_3, v_4\}$ and $\{v_1, v_7, v_8, v_{10}\}$. However, v_2 and v_7 are adjacent. In the following we show that there also exists a satisfiable formula $\Phi(\mathcal{G})$ for the family of subgraphs $\mathcal{G}$ of G.

For the important subgraph $G_{\dashv\triangleleft}$, the definition states that the non-adjacent vertices either belong to the same level or to two adjacent ones. We first consider those $G_{\dashv\triangleleft}$ for which the non-adjacent vertices belong to one level. Since each level contains only three vertices or less, there can be no $G_{\dashv\triangleleft}$ where are all vertices are in the same level. Therefore, we consider $G_{\dashv\triangleleft}$ with non-adjacent vertices on two levels. For $\text{level}_G(1)$ and $\text{level}_G(2)$, it is easy to verify that no subset of $\{v_1, v_2, v_3, v_4, v_5\}$ induces a $G_{\dashv\triangleleft}$ with the non-adjacent vertices in $\text{level}_G(1) \cup \text{level}_G(2)$. For $\text{level}_G(2)$ and $\text{level}_G(3)$ we have that the vertices $\{v_1, v_5, v_6, v_{10}\}$ induces a $G_{\dashv\triangleleft}$. For $\text{level}_G(3)$ and $\text{level}_G(4)$ we have that $\{v_2, v_6, v_7, v_8\}$ and $\{v_6, v_7, v_8, v_{10}\}$ each induce a $G_{\dashv\triangleleft}$ conveying the same information, namely v_6 must have the same color as either v_7 or v_8. Lastly, for $\text{level}_G(4)$ and $\text{level}_G(5)$ we have that $|\text{level}_G(4) \cup \text{level}_G(5)| < 4$, hence no subset of these vertices induces a $G_{\dashv\triangleleft}$.

Next, we consider the important subgraph $G_{\boxslash}$. These are defined such that the two non-

Table 3: Important subgraphs and the corresponding clauses for each of (a) $G_{\dashv}$, (b) $G_{\boxslash}$, and (c) $G_{\square}$. For the satisfying variable assignment from Table 4 all true literals are underlined.

Important Subgraph $G_{\dashv}$	Clauses
$\{v_1, v_5, v_6, v_{10}\}$	$(v_5v_{10} \vee \underline{v_5v_6}) \wedge (\neg\underline{v_5v_{10}} \vee \neg v_5v_6)$
$\{v_2, v_6, v_7, v_8\}$	$(\underline{v_6v_8} \vee v_6v_7) \wedge (\neg v_6v_8 \vee \neg\underline{v_6v_7})$
$\{v_6, v_7, v_8, v_{10}\}$	$(\underline{v_6v_8} \vee v_6v_7) \wedge (\neg v_6v_8 \vee \neg\underline{v_6v_7})$

(a)

Important Subgraph $G_{\boxslash}$	Clauses
$\{v_1, v_2, v_3, v_i\}$ for $i \in \{4, 5\}$	$\underline{v_1v_2}$
$\{v_1, v_7, v_8, v_{10}\}$	$\underline{v_1v_7}$
$\{v_2, v_{10}, v_7, v_i\}$ for $i \in \{8, 9\}$	$\underline{v_2v_{10}}$
$\{v_4, v_5, v_3, v_i\}$ for $i \in \{1, 2\}$	$\underline{v_4v_5}$
$\{v_6, v_8, v_1, v_{10}\}$	$\underline{v_6v_8}$
$\{v_8, v_9, v_7, v_i\}$ for $i \in \{2, 10\}$	$\underline{v_8v_9}$

(b)

Important Subgraph $G_{\square}$	Clauses
$\{v_1, v_2, v_3, v_i\}$ for $i \in \{6, 8\}$	$\underline{v_1v_2} \vee v_3v_i$
$\{v_1, v_2, v_i, v_j\}$ for $i \in \{4, 5\}$, $j \in \{5, 6, 8\}$, $i < j$	$\underline{v_1v_2} \vee v_iv_j$
$\{v_2, v_{10}, v_i, v_j\}$ for $i \in \{6, 8\}$, $j \in \{7, 8, 9\}$, $i < j$	$\underline{v_2v_{10}} \vee v_iv_j$

(c)

Table 4: Satisfying truth assignment for the second counterexample.

True	v_5v_6, v_6v_8, v_1v_2, v_1, v_7, v_2, v_{10}, v_4, v_5, v_6, v_8, v_8, v_9
False	v_5v_{10}, v_6v_7

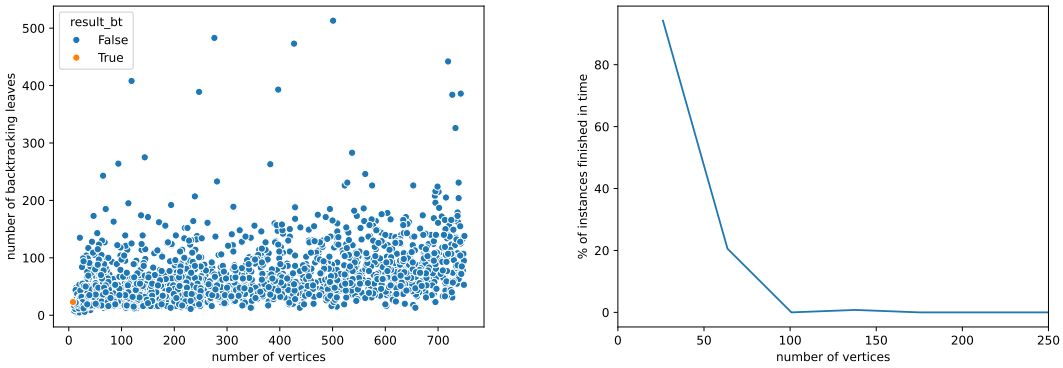
adjacent vertices u, v either directly encase each other or w.l.o.g. both endpoints of the chord corresponding to u lie to the left of both endpoints of the chord corresponding to v. The pairs of non-adjacent vertices that belong to (at least) one induced $G_{\boxslash}$ are $\{v_1, v_2\}$, $\{v_4, v_5\}$, $\{v_1, v_7\}$, $\{v_2, v_{10}\}$, $\{v_6, v_8\}$ and $\{v_8, v_9\}$. Each of these pairs contributes one clause to Φ.

Observe, that every chordless 4-cycle contains the non-adjacent pairs of vertices $\{v_1, v_2\}$ or $\{v_2, v_{10}\}$. Since these two pairs are already represented by the clauses contributed by $G_{\boxslash}$, which have to be set to `true`, every clause contributed by an induced $G_{\square}$ is already satisfied.

The important subgraphs $G_{\pentagon}$ and $G_{\bigcirc}$ are not present in G, since there are no induced cycles of length greater than 4.

Table 3 shows the clauses we obtain from the induced important subgraph and Table 4 gives a satisfying truth assignment for $\Phi \coloneqq \Phi(\mathcal{G})$.

Similar to before, the literals that are set to `true` by the truth assignment are underlined.



(a) (b)

Figure 7: (a) Number of leaves in Unger's backtracking tree. (b) Percentage of instances solved by regular backtracking within a one-hour time limit.

4 Unger's Backtracking Algorithm

In addition to the counterexample, we investigate the backtracking algorithm described by Unger. Let Φ_1 be the 2-SAT instance obtained from the important subgraphs $G_{\triangleleft}$, $G_{\square}$, $G_{\boxslash}$ and $G_{\pentagon}$. Checking whether Φ_1 is satisfiable can be done in polynomial time. For the SAT instance Φ_2 obtained from the remaining important subgraphs $G_{\pentagon}$ and C_k for $k \geq 6$ it is less clear how to solve it efficiently. Unger [11] proposes the following modified backtracking algorithm. When encountering a clause during the variable assignment whose literals are all set to `False`, we may jump in the backtracking tree to a variable which (1) results in a literal of that clause being set to `True` by flipping its assigned value and (2) the child that corresponds to this new assigned value is not included in the backtracking tree yet. We note that we rephrased property (2) from "[t]he new value of [the variable] is not already included in the backtracking tree" [11, p. 396]. If no such variable exists, the algorithm terminates and reports that there is no solution for Φ. If a solution is found, it is used to compute a 3-coloring.

Unger claims that this backtracking tree has at most $O(\log n)$ leaves. To verify this claim we implemented[2] his backtracking algorithm, ran it on randomly generated 3-colorable circle graphs and counted the number of leaves in the backtracking tree; see Figure 7a.

We evaluated Unger's backtracking algorithm on 2196 graphs ranging from 8 to 750 vertices. We generate 3-colored circle graphs by picking a random number within the range of 3 to 750 as the desired number of vertices and then inserting two endpoints of a chord representing vertex v_i into two randomly chosen distinct cells of an array. If after inserting the chord v_i the current coloring can be extended to v_i, i.e., there is at least one color left that is not yet used by any neighbor of v_i, v_i is kept and colored. Otherwise, it is discarded, and we repeat these steps for a new pair of randomly inserted endpoints. This is done until the desired number of vertices has been reached.

While the algorithm using the modified backtracking approach is reasonably fast, the number

[2]The source code as well as the used graph instances are available at https://gitlab.infosun.fim.uni-passau.de/bachmanp/3-coloring-code

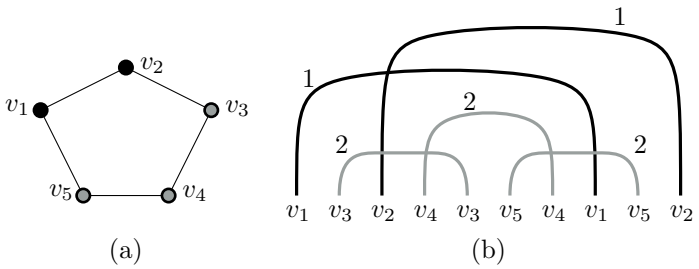


Figure 8: Graph for which Unger's backtracking might return the wrong result.

of leaves scatters quite a bit, e.g. some graphs produce backtracking trees with around 500 leaves even though they have only 250 to 500 vertices. This is likely not consistent with an upper bound of $O(\log n)$. More importantly, while all the graphs are 3-colorable, the backtracking algorithm reports no solution for all but one of the instances. In the following, we show how the algorithm may already fail for a cycle of length 5 if the order of variable assignments is poorly chosen.

Consider a cycle of length 5 and a corresponding chord diagram; see Figure 8. We show that the backtracking algorithm described by Unger can be executed such that no satisfying variable assignment is found, even though the graph is clearly 3-colorable. In this case the only important subgraph is the graph itself. We now give the clauses constructed for important subgraphs that are isomorphic to induced cycles as described in Unger's thesis [10, p. 87]. More generally, let $G = (V, E)$ with $V = \{v_1, \ldots, v_k\}$ be a circle graph isomorphic to a cycle C_k with $k \geq 5$ and let c_{aux} be an auxiliary coloring function for G with respect to $\mathcal{P}(G)$. The function h is recursively defined as follows:

$$
\begin{array}{ll}
h(v_x, v_{x+1}) \coloneqq \texttt{False} \text{ for } x \in \{1,2\} & \text{and} \\
h(v_x, v_{x+2}) \coloneqq c_{\text{aux}}(v_x, v_{x+2}) \text{ for } x \in \{1,2\} & \text{and} \\
h(v_x, v_i) = \begin{cases} h(v_x, v_{i-2}) & \text{if } c_{\text{aux}}(v_{i-2}, v_i) = \texttt{True} \\ \neg h(v_x, v_{i-2}) \wedge \neg h(v_x, v_{i-1}) & \text{if } c_{\text{aux}}(v_{i-2}, v_i) = \texttt{false} \end{cases} &
\end{array}
$$

for $x \in \{1,2\}$ and $x + 3 \leq i \leq k$. Unger shows that c_{aux} is realizable for G if and only if (i) $h(v_1, v_k) = \texttt{False}$ (ii) $h(v_1, v_{k-1}) = c_{\text{aux}}(v_1, v_{k-1})$ and (iii) $h(v_2, v_k) = c_{\text{aux}}(v_2, v_k)$. From this we infer the following formula $\Phi(G)$ for G isomorphic to C_5:

$$
\begin{array}{ll}
\underline{\neg h(v_1, v_2)} \wedge \underline{\neg h(v_2, v_3)} \wedge \underline{\neg h(v_1, v_5)} & \wedge \\
(h(v_1, v_3) \vee \neg(v_1, v_3)) \wedge \underline{(\neg h(v_1, v_3) \vee (v_1, v_3))} & \wedge \\
(h(v_2, v_4) \vee \neg(v_2, v_4)) \wedge \underline{(\neg h(v_2, v_4) \vee (v_2, v_4))} & \wedge \\
(h(v_1, v_4) \vee \neg(v_1, v_4)) \wedge \underline{(\neg h(v_1, v_4) \vee (v_1, v_4))} & \wedge \\
(h(v_2, v_5) \vee \neg(v_2, v_5)) \wedge (\neg h(v_2, v_5) \vee (v_2, v_5)) & \wedge \\
\Psi_{3,2} &
\end{array}
$$

with

$$
\begin{array}{lll}
\Psi_{k,l} \coloneqq & \bigwedge_{i=1}^{k} \bigwedge_{j=1}^{l} & \{(\neg(v_{i-2}, v_i) \vee \neg h(v_1, v_i) \vee h(v_1, v_{i-2})), \underline{(\neg(v_{i-2}, v_i) \vee h(v_1, v_i) \vee \neg h(v_1, v_{i-2}))}, \\
& & \underline{(\neg(v_{j-2}, v_j) \vee \neg h(v_2, v_j) \vee h(v_2, v_{j-2}))}, \underline{(\neg(v_{j-2}, v_j) \vee h(v_2, v_j) \vee \neg h(v_2, v_{j-2}))}, \\
& & \underline{((v_{i-2}, v_i) \vee \neg h(v_1, v_i) \vee \neg h(v_1, v_{i-2}))}, \underline{((v_{i-2}, v_i) \vee \neg h(v_1, v_i) \vee \neg h(v_1, v_{i-1}))} \\
& & \underline{((v_{i-2}, v_i) \vee h(v_1, v_i) \vee h(v_1, v_{i-2}) \vee h(v_1, v_{i-1}))}, \\
& & \underline{((v_{j-2}, v_j) \vee \neg h(v_2, v_j) \vee \neg h(v_2, v_{j-2}))}, \underline{((v_{j-2}, v_j) \vee \neg h(v_2, v_j) \vee \neg h(v_2, v_{j-1}))}, \\
& & \underline{((v_{j-2}, v_j) \vee h(v_2, v_j) \vee h(v_2, v_{j-2}) \vee h(v_2, v_{j-1}))}\}
\end{array}
$$

Table 5: Unsatisfied clauses and jumps performed during backtracking.

Pos. in backtracking tree	Contradicting clause	Jumping to value
(1)	$\neg h(v_1, v_5)$	$h(v_1, v_5)$
(2)	$\neg h(v_2, v_3)$	$h(v_2, v_3)$
(3)	$\neg h(v_1, v_2)$	$h(v_1, v_2)$
(4)	$\neg(v_3, v_5) \vee \neg h(v_2, v_5) \vee h(v_2, v_3)$	$h(v_3, v_5)$
(5)	$(v_3, v_5) \vee \neg h(v_2, v_5) \vee \neg h(v_2, v_4)$	$h(v_2, v_5)$
(6)	$\neg(v_3, v_5) \vee \neg h(v_1, v_3) \vee h(v_1, v_5)$	(v_3, v_5)
(7)	$\neg(v_2, v_4) \vee \neg h(v_1, v_4) \vee h(v_1, v_2)$	(v_2, v_4)
(8)	$(v_2, v_4) \vee \neg h(v_1, v_3) \vee \neg h(v_1, v_4)$	$h(v_1, v_3)$
(9)	$(v_2, v_4) \vee \neg h(v_2, v_4)$	$h(v_2, v_4)$
(10)	$(v_3, v_5) \vee h(v_2, v_5) \vee h(v_2, v_3) \vee h(v_1, v_4)$	n.a.

For readability, c_{aux} is omitted from these clauses and only the respective vertices are given. The clauses that are relevant to our counterexample are underlined. Since Unger does not specify an order in which the variables are assigned, we may assign values to a subset of variables in the following order:

$$h(v_2, v_5), h(v_1, v_4), h(v_2, v_4), h(v_1, v_3), h(v_1, v_5), h(v_2, v_3),$$
$$h(v_1, v_2), (v_3, v_5), (v_2, v_4)$$

We apply the following strategy for jumping in the backtracking tree, which is also consistent with Unger's description of his backtracking algorithm. Recall, that the vertices of the backtracking tree represent variables and their child edges correspond to their truth assignments. Further, when encountering a clause whose literals are all set to `False` during the variable assignment, we may jump in the backtracking tree to a variable of that clause that (1) results in a literal of that clause being set to `True` by flipping its assigned value and (2) the child edge that corresponds to this new assigned value does not exist, i.e. we have not tried both truth assignments at that vertex of the backtracking tree yet. For each variable we first set its value to `True` and check whether this assignment causes a clause to contain only `False` literals. If so, we flip the truth assignment of that variable to `False` and check, if any other clause now contains only `False` literals. If there still exists a clause whose literals are all set to `False`, we may pick a variable from that respective clause that lies on the path from the current variable to the root and is not currently set to `False`. If no such variable exists, there is no variable that satisfies properties (1) and (2) of Unger's description, and therefore the algorithm reports that the graph is not 3-colorable.

We apply this strategy to the variables listed above. The resulting backtracking tree is illustrated in Figure 9. The contradicting clauses and jumps in the backtracking tree that occur are shown in Table 5. For this specific order of variables, the algorithm runs out of variables to flip in clause (10) and therfore incorrectly reports that C_5 is not 3-colorable.

Since the modified backtracking algorithm may report incorrect results, we also evaluated a regular backtracking algorithm on the same instances. We found that the number of leaves of these backtracking trees grows exponentially. Figure 7(b) shows the percentage of solved instances within a time limit of one hour. Notably, starting at around 50 vertices, the algorithm barely

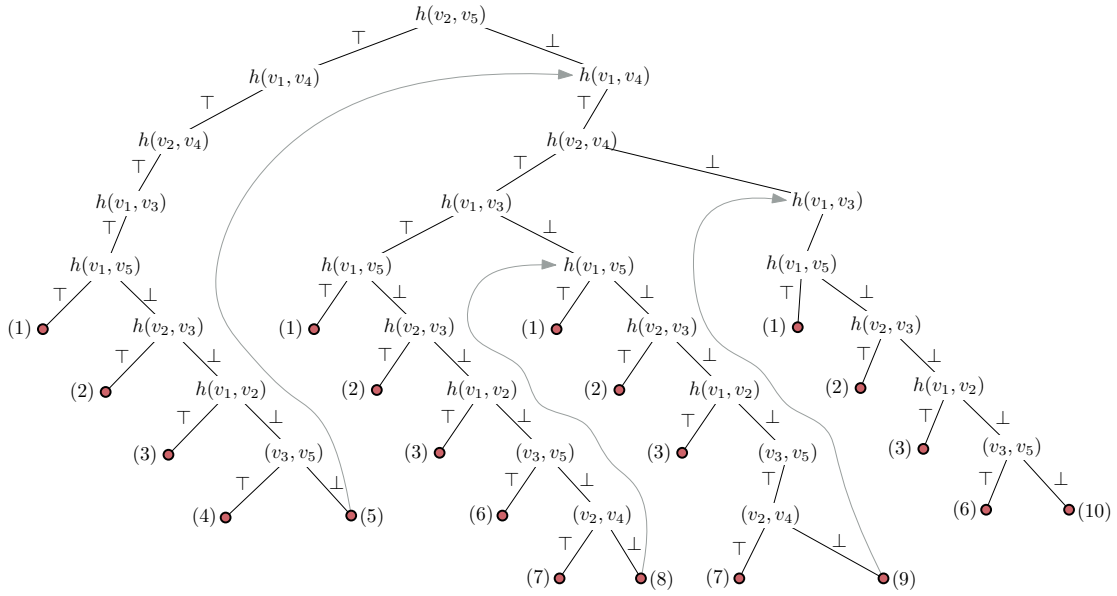


Figure 9: Backtracking tree for C_5.

manages to solve any instances. For readability, the Figure shows only the fraction of solved instances with up to 250 vertices as the backtracking algorithm does not terminate on any of the larger instances. This indicates the impracticality of a regular backtracking approach and that crucial insights are still missing to see how a backtracking approach could be modified to be more efficient on circle graphs.

Finally, we also implemented the coloring algorithm that uses the solution for Φ to color the graph. Our result is that less than 20% of the computed auxiliary coloring functions were realizable and resulted in valid colorings. This shows that claim (1) not only fails qualitatively due to the counterexample from Section 3 but also quantitatively for the vast majority of instances.

5 Conclusion

We have shown that the question of whether 3-coloring for circle graphs is possible in polynomial time should be considered open, even though Unger claimed to provide a polynomial time algorithm in a conference paper in 1992 [11]. To this end, we provided two counterexamples: one that contradicts the characterization in terms of the auxiliary coloring function and one that shows that the modified backtracking algorithm may fail to compute a correct solution. We further gave empirical evidence that displays a discrepancy to the claimed running time.

Considering the approach with important subgraphs, it appears that especially large induced cycles increase the difficulty, since for the other important subgraphs only a 2-SAT formula is constructed. Hence, an open question to investigate is the complexity of 3-coloring for circle graphs where no C_k with $k > 4$ is an induced subgraph.

We note that when considering all important subgraphs containing four vertices for the 2-SAT instance disregarding levels, we were not able to find a counterexample similar to the one in Section 3.

References

[1] P. Angelini, G. D. Battista, F. Frati, M. Patrignani, and I. Rutter. Testing the simultaneous embeddability of two graphs whose intersection is a biconnected or a connected graph. *Journal of Discrete Algorithms*, 14:150–172, 2012. doi:10.1016/j.jda.2011.12.015.

[2] M. Baur and U. Brandes. Crossing reduction in circular layouts. In J. Hromkovič, M. Nagl, and B. Westfechtel, editors, *Graph-Theoretic Concepts in Computer Science*, pages 332–343. Springer Berlin Heidelberg, 2004. doi:10.1007/978-3-540-30559-0_28.

[3] M. A. Bekos, T. Bruckdorfer, M. Kaufmann, and C. N. Raftopoulou. 1-Planar Graphs have Constant Book Thickness. In N. Bansal and I. Finocchi, editors, *Proceedings of the 23rd Annual European Symposium on Algorithms (ESA'15)*, volume 9294 of *Lecture Notes in Computer Science*, pages 130–141. Springer, 2015. doi:10.1007/978-3-662-48350-3_12.

[4] M. A. Bekos, M. Kaufmann, F. Klute, S. Pupyrev, C. N. Raftopoulou, and T. Ueckerdt. Four pages are indeed necessary for planar graphs. *J. Comput. Geom.*, 11(1):332–353, 2020. doi:10.20382/jocg.v11i1a12.

[5] F. Chung, F. Leighton, and A. 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. doi:10.1137/0608002.

[6] V. Dujmović, A. Pór, and D. R. Wood. Track layouts of graphs. *Discrete Mathematics & Theoretical Computer Science*, Vol. 6 no. 2:497–522, jan 2004. doi:10.46298/dmtcs.315.

[7] D. Eppstein. Three-colorable circle graphs and three-page book embeddings. https://11011110.github.io/blog/2014/08/09/three-colorable-circle-graphs.html, 2014. Accessed: 2023-06-6.

[8] S.-H. Hong and H. Nagamochi. Simpler algorithms for testing two-page book embedding of partitioned graphs. *Theoretical Computer Science*, 725:79–98, may 2018. doi:10.1016/j.tcs.2015.12.039.

[9] W. Unger. On the k-colouring of circle-graphs. In R. Cori and M. Wirsing, editors, *Proceedings of the 5th Annual Symposium on Theoretical Aspects of Computer Science (STACS'88)*, volume 294 of *Lecture Notes in Computer Science*, pages 61–72. Springer, 1988. doi:10.1007/BFb0035832.

[10] W. Unger. *Färbung von Kreissehnengraphen*. PhD thesis, University of Paderborn, Germany, 1990. URL: https://d-nb.info/920881181.

[11] W. Unger. The complexity of colouring circle graphs. In *Proceedings of the 9th Annual Symposium on Theoretical Aspects of Computer Science (STACS'92)*, pages 389–400. Springer Berlin Heidelberg, 1992. doi:10.1007/3-540-55210-3_199.

[12] M. Wattenberg. Arc diagrams: Visualizing structure in strings. In P. C. Wong and K. Andrews, editors, *Proceedings of the IEEE Symposium on Information Visualization (InfoVis'02)*, pages 110–116. IEEE Computer Society, 2002. doi:10.1109/INFVIS.2002.1173155.

[13] M. Yannakakis. Embedding planar graphs in four pages. *J. Comput. Syst. Sci.*, 38(1):36–67, 1989. doi:10.1016/0022-0000(89)90032-9.

[14] M. Yannakakis. Planar graphs that need four pages. *Journal of Combinatorial Theory, Series B*, 145:241–263, nov 2020. doi:10.1016/j.jctb.2020.05.008.