The Complexity of Colouring Circle Graphs

(extended abstract)

Walter Unger

Fachbereich 17
University of Paderborn
Warburger Straße 100
4790 Paderborn, Germany
Email: Walter.Unger@Uni-Paderborn.de

Abstract

We study the complexity of the colouring problem for circle graphs. We will solve the two open questions of [Un88], where first results were presented.

1. Here we will present an algorithm which solves the 3-colouring problem of circle graphs in time $O(n \log(n))$. In [Un88] we showed that the 4-colouring problem for circle graphs is NP-complete.
2. If the largest clique of a circle graph has size k then the $2 \cdot k - 1$-colouring is NP-complete. Such circle graphs are $2 \cdot k$-colourable [Un88].

Further results and improvements of [Un88] complete the knowledge of the complexity of the colouring problem of circle graphs.

Classification: algorithms and data structures, computational complexity

1 Introduction

A *circle graph* is an undirected graph which is an *intersection graph* of a set of chords in one circle. The vertex set of an intersection graph of a set of chords is that set of chords. Two vertices are joined by an edge iff the representing chords intersect each other. An example of such an intersection graph is given in Figure 1.

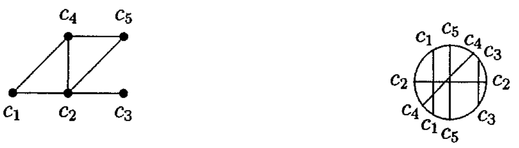


Figure 1: intersection graph of a circle graph

There are other graph classes defined as intersection graphs, such as interval graphs, path graphs and circular-arc graphs. For these and many other graph classes the complexity of colouring — and other — problems was investigated. For a survey of such graph classes see [Br90]. Before listing results for other problems of circle graphs we will describe the fields where the colouring problem for circle graphs arises.

- There is a relationship between chord colouring and the page-number of a graph. The page-number problem consists of arranging the vertices on the spline of a book so that edges are embedded without crossings on a minimum number of pages. If the vertices of the graph are already arranged on the spline of the book, then the remaining problem — the embedding of the edges on the pages — is identical to the k-colouring problem of the corresponding circle graph.

- The problem of realizing a given permutation using a minimum number of parallel stacks is modeled by the colouring of chords in a special circle graph, a permutation graph. This is presented in [EvIt71]. The number of stacks you have to use is the same as the number of colours you need to colour the corresponding permutation graph.
- The colouring of chords is closely related to some layer-assignment problems in VLSI. The layer-assignment to k layers of 2-point nets in a closed region is the same as the k-colouring of circle graphs.

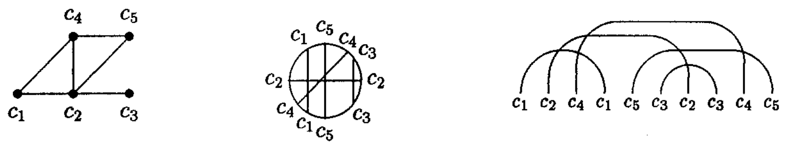
There are several papers concerned with the *recognition problem* for circle graphs [Bo75, Bo85, GaHsSu85, Na85]. The algorithm with the best running time was given by Spinrad [Sp88]: Circle graphs are recognizable in time $O(n^2)$ where n is the number of vertices. The first algorithm for the *clique problem* for circle graphs was presented by Gavril [Ga75] with a running time of $O(n^3)$. The time was improved by Rotem and Urrutia [RoUr81] to $O(n^2)$. They also presented an algorithm for the k-clique problem with a running time of $O(n \log(n))$. Wessel and Pöschel [WePo85] showed that every outerplanar graph is a circle graph. Also, permutation graphs are circle graphs. Peter Damaschke [Da89] showed that the *Hamiltonian path* problem for circle graphs is *NP*-complete. Also the *dominating set* problem is *NP*-complete due to Keil [Ke90].
The first result about the complexity of the *colouring problem* for circle graphs was given by Garey et al. [GaJoMiPa80]. They considered circular-arc graphs: the k-colouring problem for circular-arc graphs is solvable in linear time — with a quite large constant factor — and the colouring problem for circular-arc graphs is *NP*-complete. Using the last result they were able to show that the colouring problem for circle graphs is also *NP*-complete. First results on the complexity of the k-colouring problem of circle graphs were given in [Un88]. It was shown that the 4-colouring of circle graphs is *NP*-complete. Furthermore: if the largest clique has size k and $k \geqslant 3$ then the $\lceil 3/2 \cdot k \rceil$ colouring is *NP*-complete and a $2 \cdot k$ colouring is always possible. There remained the gap between $\lceil 3/2 \cdot k \rceil$ and $2 \cdot k$. The complexity of the 3-colouring of circle graphs was also not solved.
This paper will solve these open questions and will give a full description of the complexity of the k-colouring problem of circle graphs. Due to the quite large complexity of the proofs of most Theorems, we will not present their full version here. We will describe in detail the algorithm for the 3-colouring problem for circle graphs.
Section 2 is concerned with the formal definition of circle graphs and their representation. In Section 3 we will present our algorithm which solves the 3-colouring problem for circle graphs in time $O(n \log(n))$. The full description of the complexity of the k-colouring problem of circle graphs is presented in Section 4. Section 5 is concerned with g-segment graphs, which are subclasses of the class of circle graphs. Polynomial time algorithm for these g-segment graphs were also presented in [St90].
Throughout this paper we assume that circle graphs are given by their representation as a set of chords. The construction of that representation is possible in time $O(n^2)$ (see [Sp88]). The full version of this paper is presented in [Un90].

2 Representation of circle graphs and definitions

We will define in this Section the type of representation which we will use in this paper. Without loss of generality we assume that no two chords have a common endpoint on the circle. We will not use the representation as shown in Figure 1. In our representation:

- The circle is broken up and straightened.
- The arcs become now arcs above the bottom line representing the circle.

This is a direct transformation from that one being presented in Figure 1. This *circle graph representation* is shown in Figure 2. It is also known as the overlap graph model.



c_4
c_5
c_1
c_2
c_3
c_1 c_5 c_4 c_3 c_2 c_2 c_4 c_1 c_5 c_3
c_1 c_2 c_4 c_1 c_5 c_3 c_2 c_3 c_4 c_5

Figure 2: representation of a circle graph

In the following the arcs are still called chords. If a chord is named x we will denote its *left endpoint* reaching the bottom line with $\overleftarrow{x}$ and the right one with $\overrightarrow{x}$. With this notation we obtain the following formal definition of circle graphs.

Definition 1 (Circle graphs)
An undirected graph $G = (V, E)$ with $V = \{v_1, v_2, ..., v_n\}$ is a *circle graph* iff there exits a set of chords $C = \{(\overleftarrow{c_i}, \overrightarrow{c_i})\,;\ 1 \leqslant i \leqslant n\ \wedge\ \overleftarrow{c_i} < \overrightarrow{c_i}\,\}$ such that:

$$\{v_i, v_j\} = e \in E \ \text{ iff } \ c_i \otimes c_j$$

where the Boolean predicate $\otimes$ denotes the intersection of chords:

$$v \otimes w \ \text{ iff } \ \overleftarrow{v} < \overleftarrow{w} < \overrightarrow{v} < \overrightarrow{w} \ \text{ or } \ \overleftarrow{w} < \overleftarrow{v} < \overrightarrow{w} < \overrightarrow{v}$$

In the following we will only use the representation of a circle graph by a set of chords, thus a circle graph C is always of the form: $C = \{\,(\overleftarrow{c_i}, \overrightarrow{c_i})\,;\ 1 \leqslant i \leqslant n\,\}$.
Furthermore $\mathcal{N}_C(c)$ denotes the set of neighbours of c in C: $\mathcal{N}_C(c) = \{\,d \in C\,;\ d \otimes c\,\}$. □

In Figure 2 we see that some chords *cover* another and some chords are not covered by any other chord. From this we get the definition of the level structure of circle graphs.

Definition 2 (Level structure of circle graphs)
Let C be a circle graph. A chord x *covers* a chord y — denoted by: $x \rightsquigarrow y$ — iff

$$\overleftarrow{x} < \overleftarrow{y} < \overrightarrow{y} < \overrightarrow{x}$$

A chord x *directly covers* a chord y — denoted by: $x \rightarrow y$ — iff $x \rightsquigarrow y$ and there is no chord z with $x \rightsquigarrow z \wedge z \rightsquigarrow y$. The *level structure* is defined by recursion as follows:

$$\begin{aligned}
\mathtt{level}^{\downarrow}_{C}(1) &= \{\,c \in C\,;\ \nexists x \in C:\ x \rightsquigarrow c\,\} \\
\mathtt{level}^{\downarrow *}_{C}(i) &= \textstyle\bigcup_{j=1}^{i} \mathtt{level}^{\downarrow}_{C}(j) \\
\mathtt{level}^{\downarrow}_{C}(i+1) &= \mathtt{level}^{\downarrow}_{C \setminus \mathtt{level}^{\downarrow *}_{C}(i)}(1)
\end{aligned}$$

The number of levels is given by $\mathtt{LEV}(C)$, and by $\mathtt{index}^{\downarrow}_{C}(c)$ we denote the number of the level which c belongs to.

$$\begin{aligned}
\mathtt{LEV}(C) &= \max\{\,i\,;\ \mathtt{level}^{\downarrow}_{C}(i) \neq \emptyset\,\} \\
\mathtt{index}^{\downarrow}_{C}(c) &= i \ \text{ iff } \ c \in \mathtt{level}^{\downarrow}_{C}(i)
\end{aligned}$$

We call $\mathtt{level}^{\downarrow}_{C}(1)$ the top level of C, $\mathtt{level}^{\downarrow}_{C}(i)$ the i-th level of C and $\mathtt{level}^{\downarrow}_{C}(\mathtt{LEV}(C))$ the bottom level of C.
Furthermore let $\overset{\rightsquigarrow}{\mathcal{O}}_C\ (c) = \{\,d \in C\,;\ c \rightsquigarrow d\,\}$ be the set of chords covered by c. □

This level structure depends on the chosen representation.
A function $\mathtt{col} : C \longrightarrow \{1, 2, ..., k\}$ is a k-colouring iff $(a \otimes b\ \Rightarrow\ \mathtt{col}(a) \neq \mathtt{col}(b))$. With $\mathtt{COL}(C)$ we denote the minimal number of colours to colour C and with $\mathtt{CLI}(C)$ the size of the largest clique in C.

3 The 3-colouring problem for circle graphs

This Section contains the description of an algorithm for the 3-colouring problem of circle graphs. This algorithm will illustrate the fundamental difference between the 3-colouring problem and the 4-colouring problem of circle graphs. The latter one is $\mathcal{NP}$-complete [Un88]. Thus this algorithm is "optimal", because there is — assuming $P \neq NP$ — no polynomial time algorithm for the 4-colouring of circle graphs. We will not describe the full algorithm which solves the 3-colouring problem of a circle graph C in time $O(n\log(n))$. That algorithm is too complicated to be presented here. But the "basic version" of that algorithm will be described.
The algorithm has three phases:

- The first phase computes a set of "important subgraphs" of C. Each important subgraph C_I is checked for conditions necessary for a correct 3-colouring of C_I. These conditions are expressed in a boolean formula $\mathcal{F}$.
- In the second phase an assignment is computed for the variables of $\mathcal{F}$ such that $\mathcal{F}$ is satisfied. If $\mathcal{F}$ is not satisfiable then C is not colourable with 3 colours and the algorithm will finish.
- Otherwise, the third phase uses the assignment of the variables of $\mathcal{F}$ to compute a valid 3-colouring of C. This colouring is done level by level. In round i level $\mathtt{level}^{\downarrow}_{C}(i)$ is coloured.

The three phases will be described in detail below. Phases one and two have to gather enough information for the third phase. This information will be given by a function $\mathtt{val}_3 : \mathcal{D} \to$ boolean, where $\mathcal{D}$ is a subset of the set of pairs of chords. A pair of chords $\{a, b\}$ is in $\mathcal{D}$ iff the following condition holds:

$$\begin{array}{ll} \neg(a \otimes b) & \text{and} \\ \mathcal{N}_C(a) \cap \mathcal{N}_C(b) \neq \emptyset & \text{and} \\ \mathtt{index}^{\downarrow}_{C}(a) = \mathtt{index}^{\downarrow}_{C}(b) \text{ or } a \to b & \end{array}$$

If $\mathtt{val}_3(a, b)$ is $\mathtt{true}$ then the two chords a and b have to be coloured with one colour, otherwise they have to be coloured by two different colours.
We will illustrate this gathering of information and the colouring using $\mathtt{val}_3$ with the example shown in Figure 3.

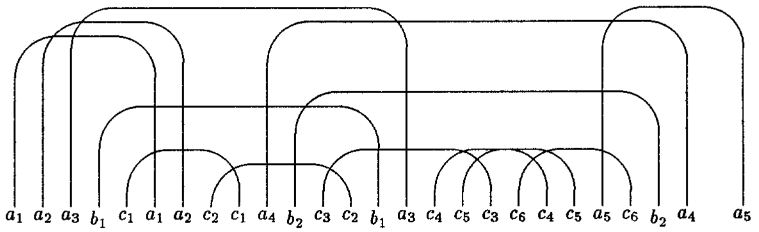

a_1 a_2 a_3 b_1 c_1 a_1 a_2 c_2 c_1 a_4 b_2 c_3 c_2 b_1 a_3 c_4 c_5 c_3 c_6 c_4 c_5 a_5 c_6 b_2 a_4 a_5

Figure 3: circle graph C_e

The circle graph C_e has the following three levels:

$$\begin{array}{llll} \mathtt{level}^{\downarrow}_{C_e}(1) = \{a_1, a_2, a_3, a_4, a_5\} & & \mathtt{level}^{\downarrow}_{C_e}(2) = \{b_1, b_2\} \\ \mathtt{level}^{\downarrow}_{C_e}(3) = \{c_1, c_2, c_3, c_4, c_5, c_6\} & & \end{array}$$

The function $\mathtt{val}_3$ has to be defined for the following domain:

$$\mathcal{D} = \left\{ \begin{array}{c} \{a_1,a_4\},\{a_2,a_4\},\{a_3,a_5\},\{a_3,b_1\},\{a_4,b_2\},\{b_1,c_1\},\{b_1,c_2\}, \\ \{b_2,c_3\},\{b_2,c_6\},\{c_1,c_3\},\{c_2,c_4\},\{c_2,c_5\},\{c_3,c_6\} \end{array} \right\}$$

We will now look for important subgraphs from which we will get some conclusions for the function $\texttt{val}_3$. This is done as follows: For every important subgraph there is a 3-colouring which obeys the function $\texttt{val}_3$. A 3-colouring of an important subgraph C_I obeys $\texttt{val}_3$ iff for all chords a, b in C_I $\texttt{val}_3(a,b) = (\texttt{col}(a) = \texttt{col}(b))$ holds.
The first important subgraph is $\{a_1, a_2, a_3, b_1\}$, which is also shown in Figure 4.

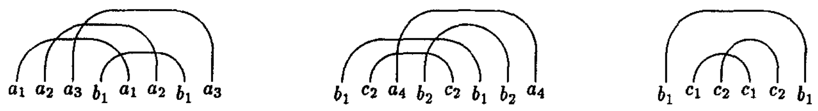

a_1 a_2 a_3 b_1 a_1 a_2 b_1 a_3
b_1 c_2 a_4 b_2 c_2 b_1 b_2 a_4
b_1 c_1 c_2 c_1 c_2 b_1

Figure 4: examples of important subgraphs

In a correct 3-colouring the chords a_3 and b_1 have to be coloured with the same colour, because of the two intersecting chords a_1, a_2. Thus we have to define: $\texttt{val}_3(a_3, b_1) = \texttt{true}$.
There are more values for $\texttt{val}_3$ that we have to define like the above one:

$$\texttt{val}_3(b_1,c_1) = \texttt{true} \quad \text{by important subgraph: } \{a_1,a_2,b_1,c_1\}$$
$$\texttt{val}_3(c_3,c_6) = \texttt{true} \quad \text{by important subgraph: } \{c_3,c_4,c_5,c_6\}$$

Another type of important subgraph is $\{a_4, b_1, b_2, c_2\}$. This subgraph — also seen in Figure 4 — forms a cycle of length four. It is not possible to have $\texttt{val}_3(a_4, b_2) = \texttt{val}_3(b_1, c_2) = \texttt{false}$, because then this subgraph must be coloured with four colours. We will express this by the following clause: $(\texttt{val}_3(a_4, b_2) \vee \texttt{val}_3(b_1, c_2))$. In C_e there are more of these cycles of length four. Therefore, we also get the following clauses:

$$\begin{array}{ll} (\texttt{val}_3(b_2,c_3) \vee \texttt{val}_3(b_1,c_3)) & (\texttt{val}_3(c_3,d_1) \vee \texttt{val}_3(b_1,c_3)) \\ (\texttt{val}_3(a_3,b_1) \vee \texttt{val}_3(a_4,b_2)) & (\texttt{val}_3(a_3,b_1) \vee \texttt{val}_3(b_2,c_3)) \\ (\texttt{val}_3(a_4,b_2) \vee \texttt{val}_3(a_3,a_5)) & \end{array}$$

The next type of observation is made with the important subgraph $\{b_1, c_1, c_2\}$, see Figure 4.
For these chords the values of $\texttt{val}_3(b_1, c_1)$ and $\texttt{val}_3(b_1, c_2)$ have to be defined, but both of them cannot be $\texttt{true}$. If both of them are true, we have to colour c_1 and c_2 with the same colour as b_1. But this is not possible, because c_1 and c_2 intersect each other. Thus our formula $\mathcal{F}$ will include the clause $(\neg\texttt{val}_3(b_1, c_1) \vee \neg\texttt{val}_3(b_1, c_2))$.
The last conditions we get for $\mathcal{F}$ result from the important subgraphs $\{a_3, a_5, b_2, c_3, c_4, c_5, c_6\}$ and $\{a_2, a_4, b_1, c_1, c_2\}$. From these we get some more complex clauses, which we will not state here. But we keep these subgraphs in mind for the evaluation of the formula $\mathcal{F}$.
In the next step we have to evaluate $\mathcal{F}$ to get a valid assignment to the function $\texttt{val}_3$. For this evaluation we have listed "many" clauses of length two and "some" conditions from two important subgraphs which we did not list. Thus we will use in our example just the clauses of length two. The evaluation of $\mathcal{F}$ is started with the following conclusions:

$$\begin{array}{lcl} \texttt{val}_3(b_1,c_1) = \texttt{true} & \Rightarrow & \neg\texttt{val}_3(b_1,c_2) \\ & \Rightarrow & \texttt{val}_3(a_4,b_2) \wedge \texttt{val}_3(b_2,c_3) \wedge \texttt{val}_3(c_3,d_1) \\ \texttt{val}_3(b_2,c_3) \wedge \texttt{val}_3(c_3,c_6) & \Rightarrow & \texttt{val}_3(b_2,c_6) \end{array}$$

The second conclusion is made using the subgraph $\{a_3, a_5, b_2, c_3, c_4, c_5, c_6\}$. We may define some values of $\texttt{val}_3$ without getting any conclusions:

$$\texttt{val}_3(d_1, d_3) \;=\; \texttt{true} \quad \text{and} \quad \texttt{val}_3(a_3, a_5) \;=\; \texttt{false}$$

From other values of *val* we get some conclusions:

$$\begin{aligned} \texttt{val}_3(a_2, a_4) = \texttt{false} &\;\Rightarrow\; \texttt{val}_3(a_1, a_4) \wedge \texttt{val}_3(a_2, c_2) \wedge \neg\texttt{val}_3(a_4, c_1) \\ \texttt{val}_3(a_3, c_4) = \texttt{false} &\;\Rightarrow\; \texttt{val}_3(a_3, c_5) \wedge \neg\texttt{val}_3(a_5, c_5) \end{aligned}$$

We will now use the information stored in function $\texttt{val}_3$ to compute a valid 3-colouring of C_e. This colouring will be done level by level. At first the chords a_i $(1 \leqslant i \leqslant 3)$ are coloured with colour i. We will see that from now on the function $\texttt{val}_3$ will define the colours of the other chords in a unique way.
Because of the value of $\texttt{val}_3(a_1, a_4)$ the chord a_4 is coloured with 1 and because of $\texttt{val}_3(a_3, a_5)$ a_5 is coloured with 2. Also the colouring of the next level b_1, b_2 is uniquely defined by $\texttt{val}_3$: $\texttt{col}(b_1) = 3$ and $\texttt{col}(b_2) = 1$. The chord c_1 has to be coloured with 3, because $\texttt{val}_3(b_1, c_1)$ holds. For the chord c_2 the only possible colour is 2, and this obeys function $\texttt{val}_3$. This technique is continued until C_e is coloured. The final colouring is seen in Figure 5.

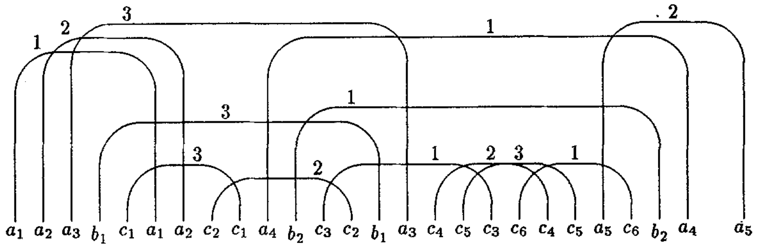

1
2
3
1
2
3
1
3
2
1
2
3
1
a_1 a_2 a_3 b_1 c_1 a_1 a_2 c_2 c_1 a_4 b_2 c_3 c_2 b_1 a_3 c_4 c_5 c_3 c_6 c_4 c_5 a_5 c_6 b_2 a_4 a_5

Figure 5: coloured example circle graph C_e

This example already gives quite a good description of the algorithm. There are three questions left:
How is it possible to get the claimed running time? Using the description from above this is not possible. We introduced the important subgraphs which form a cycle of length four. But there may be $\Omega(n^2)$ such important subgraphs and $\Omega(n^2)$ clauses from them. We have to be very careful in defining the clauses which have to be computed. But the full description of this will not fit into this paper. Hence we will describe a polynomial time algorithm.
How is it possible to evaluate the formula $\mathcal{F}$ in the claimed — or at least polynomial — time? For the part of $\mathcal{F}$ which consists of the clauses of length two, this is easy, but for the other clauses this seems to be hard.
We have to prove that the second phase stops iff the circle graph is not 3-colourable and that the third phase will always succeed.
We will answer the latter questions in the following part.

3.1 Phase 1: Computing formula $\mathcal{F}$

First we have to define what the important subgraphs of a circle graph C are. For the $O(n \log(n))$ time version of this algorithm this is rather complicated. Here we will give in Table 1 a "simplified" version of important subgraphs by listing them as undirected graphs.
We have to make the following remarks: The clauses will only be defined if the parameters of $\texttt{val}_3$ are in $\mathcal{D}$. The latter important subgraphs — the cycles of length five or more, denoted as "type 6" subgraphs — are not defined in the correct way by the above picture. To be more precise: A cycle C_l of length $l \geqslant 5$ is an important subgraph iff the following is true:

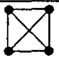
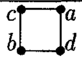
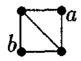
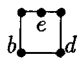
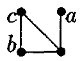
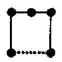
	subgraphs	clauses		subgraphs	clauses
1)		`false`	4)		$\mathtt{val}_3(a,b) \vee \mathtt{val}_3(a,c)$
2)		$\mathtt{val}_3(a,b)$	5)		$\neg\mathtt{val}_3(e,b) \vee \neg\mathtt{val}_3(e,d)$
3)		$\mathtt{val}_3(a,b) \neq \mathtt{val}_3(a,c)$	6)		"Some large expression"

Table 1: Important subgraphs and clauses

- There exists an i such that $C_I \subseteq \mathtt{level}^{\downarrow}_{C}(i) \cup \mathtt{level}^{\downarrow}_{C}(i+1)$.
- It is not possible using an extra cord c to split C_I into two different cycles. We will illustrate this using the example from Figure 3. If we insert a chord x into C_e such that $\{x\} = \mathcal{N}_{C_e}(c_4) \cap \mathcal{N}_{C_e}(c_5) \cap \mathcal{N}_{C_e}(b_2)$ holds then $\{a_3, a_5, b_2, c_3, c_4, c_5, c_6\}$ will not be an important subgraph. There will be two other important subgraphs: $\{a_3, b_2, c_3, c_4, c_5, x\}$ and $\{a_5, b_2, c_4, c_5, c_6, x\}$.

The rather tedious work of defining the clauses for such an important subgraph is omitted.
At this stage we have computed a formula $\mathcal{F}$. This formula contains many clauses of length two and some clauses from these type 6 subgraphs.
We used the set of important subgraphs like a "fishing net" which we throw over C to extract the information for $\mathcal{F}$. The important subgraphs are the meshes of the fishing net. To gather enough information, we have to be sure that the meshes are small enough.
Let I be the set of important subgraphs. We will define the connectivity graph $\mathcal{G}$ as follows:

$$\begin{aligned} \mathcal{G} &= (I, E) \qquad\qquad \text{with} \\ \{I, J\} \in E &\text{ iff } \exists a, b \in I \cap J : a \otimes b \end{aligned}$$

Without loss of generality we may assume that C is 2-connected. Then — using careful observation — we prove that $\mathcal{G}$ is connected. This will ensure that we are gathering enough information. Furthermore, it will also ensure that the third phase will compute a correct colouring of C.

3.2 Phase 2: Computing a solution for $\mathcal{F}$

The formula $\mathcal{F}$ consists of two parts. $\mathcal{F}_1$ contains the clauses of length two and $\mathcal{F}_2$ the clauses from the important subgraphs of type 6. The evaluation of $\mathcal{F}_1$ is possible in linear time, it is an instance of 2-SAT. If there is no assignment to $\mathtt{val}_3$ such that $\mathcal{F}_1$ is satisfied, then we are done. In this case C is not 3-colourable. From now on we assume that $\mathcal{F}_1$ is satisfiable.
The clauses of $\mathcal{F}_2$ are not so easy to evaluate. We have to use a backtracking technique. In a preprocessing step we will prepare the implications of $\mathcal{F}_1$ for later use.
Compute the dependencies of variables implied by $\mathcal{F}_1$ in $\mathcal{F}_2$. This preprocessing is done as follows: For each variable of the form $\mathtt{val}_3(a,b)$ which is in $\mathcal{F}_2$, assign the value `true` to it and compute all conclusions we get by using $\mathcal{F}_1$. If we get a conclusion for another variable $\mathtt{val}_3(c,d)$ which is also in $\mathcal{F}_2$, we will store this information in a formula $\mathcal{F}_3$ as:

$$\begin{aligned} \mathtt{val}_3(a,b) &\Rightarrow \mathtt{val}_3(c,d) \qquad \text{or as} \\ \mathtt{val}_3(a,b) &\Rightarrow \neg\mathtt{val}_3(c,d) \end{aligned}$$

The same is done for the assignment $\mathtt{val}_3(a,b) = \mathtt{false}$. We will use $\mathcal{F}_3$ together with $\mathcal{F}_2$ in the backtracking to compute a solution for $\mathcal{F}$.
We will use a modified backtracking, where the backtracking tree is stored. Thus our algorithm may jump from one node in the tree to another node to continue the computation at the second

node. This jumping is used if the assignment to the variables yields a contradiction. If a contradiction did occur, then we will try to resolve this contradiction. I.e. we will look for a variable $\mathtt{val}_3(a,b)$ with the following properties:

- This contradiction is broken up by changing the value $\mathtt{val}_3(a,b)$.
- The new value of $\mathtt{val}_3(a,b)$ is not already included in the backtracking tree.

If there is no such variable, then there is also no solution for $\mathcal{F}$, and C is not 3-colourable. If there is such a variable, our algorithm will jump in the backtracking tree.
Furthermore if the algorithm finds a solution to $\mathcal{F}$ it will continue with the third phase using this solution.
For this backtracking strategy we are able to prove the following Lemma:

Lemma 1 (Size of the backtracking tree)
The backtracking tree of the above algorithm has at most $O(\log(n))$ leaves. □

Because of the small number of leaves in the backtracking tree the algorithm for evaluating $\mathcal{F}$ will run in polynomial time. The algorithm is also correct, because we did just change the traversal of the backtracking tree, not the tree itself.

3.3 Phase 3: Using $\mathcal{F}$ for the colouring

At this stage we have computed a function $\mathtt{val}_3$ such that for each important subgraph of C there exists a colouring which obeys $\mathtt{val}_3$.
The colouring of the circle graph C using the function $\mathtt{val}_3$ is done level by level. In the i-th round level $\mathtt{level}^{\downarrow}_C(i)$ is coloured. Without loss of generality we assume that each level is connected and that the leftmost chord of a the i-th level has a neighbour in level $i-1$. Then each level is coloured in a left to right motion. We have to prove that this algorithm will never fail. This is done step by step with the following list of statements.

- If the value of $\mathtt{val}_3(a,b)$ is defined and the chords a and $c \in \mathcal{N}(a) \cap \mathcal{N}(b)$ are already coloured, then the value of $\mathtt{val}_3(a,b)$ determines a unique and correct colouring of the chord b.
- If two intersecting chords of an important subgraph C_I are coloured then the function $\mathtt{val}_3$ defines a correct and unique colouring of C_I.
- If the first two chords of the top level are coloured then the function $\mathtt{val}_3$ defines a correct and unique colouring of the top level.
- If $\mathtt{level}^{\downarrow *}_C(i)$ $(1 \leqslant i < \mathtt{LEV}(C))$ is coloured then the function $\mathtt{val}_3$ defines a correct and unique colouring of $\mathtt{level}^{\downarrow *}_C(i+1)$.

The proofs of these statements are done by a close observation of the structure of important subgraphs. This observation uses the connectivity graph defined above.
This also finishes the description of our algorithm and the proof of correctness.

4 The k-colouring of circle graphs: a full description

In this Section the complexity of the colouring problem of circle graphs is thoroughly investigated. We will investigate the following properties: 4-cycle free, 3-cycle free, bounded level size, bounded degree and bounded clique size. For all these properties we will describe the complexity of the corresponding colouring problem. Due to the bounded space we have to skip most of the proofs.
First we will consider circle graphs without cycles of length four. A cycle in a circle graph is the same as a cycle in the corresponding undirected graph.

<u>**Theorem 2**</u> (Colouring circle graphs without cycles of length four [Un88])
A circle graph with no cycle of length four is colourable with three colours. □

In the next step we consider circle graphs without cycles of length three (clique of size three).

<u>**Theorem 3**</u> (Colouring circle graphs without cycles of length three [Un88])
A circle graph with no cycle of length three is colourable with four colours. □

The above Theorems are tight, because there is a circle graph without cycles of length three which is not colourable with three colours and there is a circle graph without cycles of length four which is not colourable with two colours.
We will now consider the colouring of circle graphs with bounded level depth. First let us recall some simple definition: A proper interval graph is an interval graph which does not have two intervals I, J with: $I \subseteq J$. A graph G is perfect iff for each node induced subgraph G' of G $\mathtt{COL}(G') = \mathtt{CLI}(G')$ holds.

<u>**Lemma 4**</u> (Circle graphs with $\mathtt{LEV}(C) = 1$)
The class of circle graphs with only one level is the same as the class of proper interval graphs. Thus this class is perfect and the colouring problem is solvable in time $O(n\log(n))$.

<u>**Proof:**</u>
A circle graph C with $\mathtt{LEV}(C) = 1$ does not contain two chords c, c' with $c \rightsquigarrow c'$. We will assign to each chord $c_i \in C$ the interval $I_i = [\overleftarrow{c}_i, \overrightarrow{c}_i]$. Then two chords c_i, c_j intersect iff the corresponding intervals I_i, I_j have a nonempty intersection. The interval graph defined in this way is proper, because there are no intervals I_i, I_j with $I_i \supseteq I_j$. Otherwise $c_i \rightsquigarrow c_j$ would be true. ■

<u>**Theorem 5**</u> (k-colouring circle graphs with $\mathtt{LEV}(C) = d$)
The k-colouring problem for circle graphs with bounded level depth is solvable in time $O(n\log(n))$.

<u>**Proof:**</u>
Finding a clique of size $k+1$ is possible in time $O(n\log(n))$ [RoUr81]. Because of this we may assume without loss of generality: $\mathtt{LEV}(C) = d$ and $\mathtt{CLI}(C) \leqslant k$.
Let $\overleftrightarrow{\mathcal{T}}_C\,(x)$ be the set of chords covering a point $x \in \mathbb{R}$: $\overleftrightarrow{\mathcal{T}}_C\,(x) = \{c \in C\,;\, \overleftarrow{c} < x < \overrightarrow{c}\}$. The clique size is bounded by k, thus for any level $\mathtt{level}_C^{\downarrow}(i)$ and any $x \in \mathbb{R}$ $|\overleftrightarrow{\mathcal{T}}_{\mathtt{level}_C^{\downarrow}(i)}\,(x)| \leqslant k$ will hold. Then $|\overleftrightarrow{\mathcal{T}}_C\,(x)| \leqslant k \cdot d$ holds for all $x \in \mathbb{R}$. Thus a line-sweep algorithm has to store a finite amount of information and will solve the problem. ■

If k is not fixed — is a part of the input — the situation changes:

<u>**Theorem 6**</u> (The colouring of circle graphs with $\mathtt{LEV}(C) \geqslant 7$)
The colouring of circle graphs with at least 7 levels is NP-complete. □

The next lemma — an improved version from [Un88] — is devoted to degree bounded circle graphs.

<u>**Lemma 7**</u> (Circle graphs with bounded degree)
A circle graph C with degree at most d has a d-seperator[1]. □

In [Un88] the existence of a $2 \cdot d$-seperator was presented and the following Theorem was concluded.

<u>**Theorem 8**</u> (Colouring circle graphs with bounded degree [Un88])
The k-colouring problem for circle graphs where the degree is at most d is solvable in time $O(n\log(n))$. □

<u>**Theorem 9**</u> (The 3-colouring problem for circle graphs)
The 3-colouring of circle graphs is solvable in time $O(n\log(n))$.

[1]A graph has a d-seperator iff there exist d nodes such that the deletion of these d nodes slit the graph into two parts which are not connected and each part contains at least $1/3$ of the nodes of the graph.

<u>Proof:</u>
See Section 3 ■

<u>Theorem 10</u> (The 4-colouring problem for circle graphs [Un88])
The 4-colouring problem for circle graphs with $\mathtt{CLI}(C) = 3$ is NP-complete. □

Using a circle graph C_k with clique-size k which contains an independent set which uses 2 colours in any $\lceil (2/3) \cdot k \rceil$-colouring we were able to prove in [Un88]: The $\lceil (2/3) \cdot k \rceil$-colouring problem is NP-complete for circle graph with clique-size k and for all $k \geqslant 3$.
We have now constructed a circle graph C'_k with clique-size k which contains an independent set which uses 2 colours in any $2 \cdot k - 1$-colouring. Thus we are able to close the gap:

<u>Theorem 11</u> (The $2 \cdot k - 1$-colouring problem for circle graphs with $\mathtt{CLI}(C) = k$)
The $2 \cdot k - 1$-colouring problem for circle graphs with $\mathtt{CLI}(C) = k$ is NP-complete. □

<u>Theorem 12</u> (The $2 \cdot k$-colouring problem for circle graphs with $\mathtt{CLI}(C) = k$ [Un88])
Let C be a circle graph with $k = \mathtt{CLI}(C)$. Then $\mathtt{COL}(C) \leqslant 2 \cdot k$ holds and a $2 \cdot k$-colouring is computable in time $O(n \log(n))$. □

5 Segment graphs

Circle graphs were defined by a set of chords on one circle, but we may replace the circle by a g-polygon. In this g-polygon we will place chords and define an intersection graph in the same way as for circle graphs. By splitting the g-polygon at one corner and straightening the g-polygon we will get the following formal definition of these g-segment graphs.

<u>Definition 3</u> (g-segment graphs)
A circle graph C is called a *g-segment graph* iff there exist $g-1$ points $e_1, e_2, ...e_{g-1}$ such that for all $c \in C$ there is one e_i such that $\overleftarrow{c} < e_i < \overrightarrow{c}$ holds. □

We start with a note on 2-segment graphs. The class of 2-segment graphs is the same as the class of circle graphs which have an equator. A circle graph C has an equator iff we are able to add a single chord which crosses each other chord of C. Using the well known result that these graph class is the same as the permutation graphs we have the following Lemma.

<u>Lemma 13</u> (2-segment graphs)
The class of 2-segment graphs is the same as the class of permutation graphs. Thus this class is perfect. □

<u>Lemma 14</u> (g-segment graphs with $\mathtt{CLI}(C) = k'$)
There is a function $f(g, k')$ such that for each g-segment graph C with $\mathtt{CLI}(C) = k'$ there is a g-segment graph $C' \subseteq C$ with: $\mathtt{COL}(C') = \mathtt{COL}(C)$ and $|C'| \leqslant f(g, k')$. This subgraph C' is computable in time $O(n \log(n))$. □

Using this we are able to solve the k-colouring problem for g-segment graphs.

<u>Theorem 15</u> (k-colouring g-segment graphs with $\mathtt{CLI}(C) = k'$)
The k-colouring of g-segment graphs with $\mathtt{CLI}(C) = k'$ is solvable in time $O(n \log(n))$. □

6 Conclusions

We presented in this article a nearly complete description of the complexity of the colouring problems of circle graph. Table 2 shows this in a compact version.
There are two problems still open. One is a minor one: How many levels must a circle graph have, such that the colouring problem is NP-complete? The second open problem is the complexity of the colouring problem for g-segment graphs.

circle graph C	x-Colouring Problem					
	$x=3$	$x=4$	$x=k$	$x=2k-1$	$x=2k$	$x=\epsilon$
C without restrictions	$\mathcal{P}$	$\mathcal{NPC}$	$\mathcal{NPC}$	$\mathcal{NPC}$	$\mathcal{NPC}$	$\mathcal{NPC}$-
$\mathtt{CLI}(C)=k\geqslant 3$	$\mathcal{P}$	$\mathcal{NPC}$	$\mathcal{NPC}$	$\mathcal{NPC}$	$\mathcal{P}+$	$\mathcal{NPC}$
degree$(C)=k$	$\mathcal{P}$	$\mathcal{P}$	$\mathcal{P}$	$\mathcal{P}$	$\mathcal{P}$	$\mathcal{P}$
C without 4-cycle	$\mathcal{P}+$	$\mathcal{P}+$	$\mathcal{P}$	$\mathcal{P}$	$\mathcal{P}$	$\mathcal{P}$
C without 3-cycle	$\mathcal{P}$	$\mathcal{P}+$	$\mathcal{P}$	$\mathcal{P}$	$\mathcal{P}$	$\mathcal{P}$
C is a 2-segment graph	$\mathcal{P}*$	$\mathcal{P}*$	$\mathcal{P}*$	$\mathcal{P}*$	$\mathcal{P}*$	$\mathcal{P}*$
C is a g-segment graph	$\mathcal{P}$	$\mathcal{P}$	$\mathcal{P}$	$\mathcal{P}$	$\mathcal{P}$	open
$\mathtt{LEV}(C)=1$	$\mathcal{P}*$	$\mathcal{P}*$	$\mathcal{P}*$	$\mathcal{P}*$	$\mathcal{P}*$	$\mathcal{P}*$
$2\leqslant \mathtt{LEV}(C)\leqslant 6$	$\mathcal{P}$	$\mathcal{P}$	$\mathcal{P}$	$\mathcal{P}$	$\mathcal{P}$	open
$\mathtt{LEV}(C)\geqslant 7$	$\mathcal{P}$	$\mathcal{P}$	$\mathcal{P}$	$\mathcal{P}$	$\mathcal{P}$	$\mathcal{NPC}$

Entry	Description of Entry
$\mathcal{P}$	Solvable in polynomial time
$\mathcal{P}+$	Colouring always possible and solvable in polynomial time
$\mathcal{P}*$	Solvable in polynomial time because that class is perfect
$\mathcal{NPC}$	$\mathcal{NP}$-complete
$\mathcal{NPC}$-	$\mathcal{NP}$-complete, not shown here or in [Un88]
open	open Problem

Table 2: The complexity of the colouring of circle graphs

Acknowledgment

Many discussions with B. Monien accompanied this work. His help is gratefully acknowledged.

References

[Bo75] *A. Bouchet*, **Reducing prime graphs and recognizing circle graphs**, Combinatorica, 1975

[Bo85] *A. Bouchet*, **Characterizing and recognizing circle graphs**, Proc. of the 6th Yugoslav Seminar on Graph Theory, Dubrovnik 1985

[Br90] *Andreas Brandstädt*, **Special graph classes — a survey (preliminary version)**, Forschungsergebnisse Friedrich-Schiller-Universität Jena, Sektion Mathematik, 1990

[Da89] *Peter Damaschke,* **The Hamilton Cycle Probelm is $\mathcal{NP}$-Complete for Circle Graphs,** Technical Report, University of Jena, O-6900 Jena, Germany

[EvIt71] *S. Even and A. Itai*, **Queues, stacks and graphs**, Theory of Machines and Computations, 1971, 71-86

[Fr84] *H. de Fraysseix*, **A Characterization of Circle Graphs**, Europ. J. Combinatorics, Vol 5, 1984, 223-238

[GaHsSu85] *C.P. Gabor and W.-L. Hsu and K.J. Supowit*, **Recognizing Circle Graphs in Polynomial Time**, FOCS, 85

[GaJoMiPa80] *M.R. Garey and D.S. Johnson and G.L. Miller and Ch.H. Papadimitriou*, **The Complexity of Coloring Circular Arcs and Chords**, SIAM Alg. Disc. Meth., 1. No. 2., 1980, 216-227

[Ga75] *F. Gavril*, **Algorithms for a Maximum Clique and A Maximum Independent Set of a Circle Graph**, Networks 3, 1975, 261-273

[GuLeLe82] *U.I. Gupta and D.T. Lee and J.Y.T. Leung*, **Efficient Algorithms for Interval Graphs and Circular-Arc Graphs**, Networks, 12, 1982, 459-467

[Ke90] *J.M. Keil,* **The Dominating Set Problem for Circle Graphs is *NP*-Complete,** Personal commonications with A. Brandstädt

[Le84] *J.Y.-T. Leung*, **Fast Algorithms for Generating All Maximal Independent Sets of Interval, Circular-Arc and Chordal Graphs**, Journal of Algorithms 5,, 1984, 11, 22-35

[Na85] *W. Naji*, **Reconnaissance des graphes de cordes**, Discr. Math. 54, 1985, 329-337

[OrBoBo82] *J.B. Orlin and M.A. Bonuccelli and D.P. Bovet*, **An $O(n^2)$ Algorithm for Coloring Proper Circular Arc Graphs**, SIAM Alg. Disc. Meth., 2. No. 2., June 1982, 88-93

[RePoUr82] *R.C. Read and D. Rotem and J. Urrutia*, **Orientation of Circle Graphs**, Journal of Graph Theory, 6, 1982, 325-341

[RoUr81] *D. Rotyem and J. Urrutia*, **Finding Maximum Cliques in Circle Graphs**, Networks, 11, 1981, 269-278

[Sp88] *J.P. Spinrad*, **Recognition of Circle Graphs**, Manuscript 1988, Dept. of CS, Vanderbilt Univ., Nashville, TN

[St90] *L. Stewart and E. Elmallah and J. Culberson*, **Polynomial algorithms on *k*-polygon graphs**, Proc. 21th SE Conf. Combin. Graph Theory and Computing, Boca Raton, Florida 1990

[Tu80] *A. Tucker*, **An Efficient Test for Circular-Arc Graphs**, SIAM J. Comput. Vol. 9. No. 1., 1980, 1-24

[Un90] *W. Unger*, **Färbung von Kreissehnengraphen**, Ph.D. Thesis, University of Paderborn, Germany

[Un88] *W. Unger*, **On the k-colouring of Circle Graphs**, Lecture Notes in Computer Science, 294, Springer Verlag, proc. STACS 88 Bordeaux, pp 61-72

[WePo85] *W. Wessel and R. Pöschel*, **On Circle Graphs**, Graphs, Hypergraphs and Applications, 1985, 207-210

[Ya86] *M. Yannakakis*, **Four Pages are necessary and sufficient for Planar Graphs**, ACM, 1986, 104-108