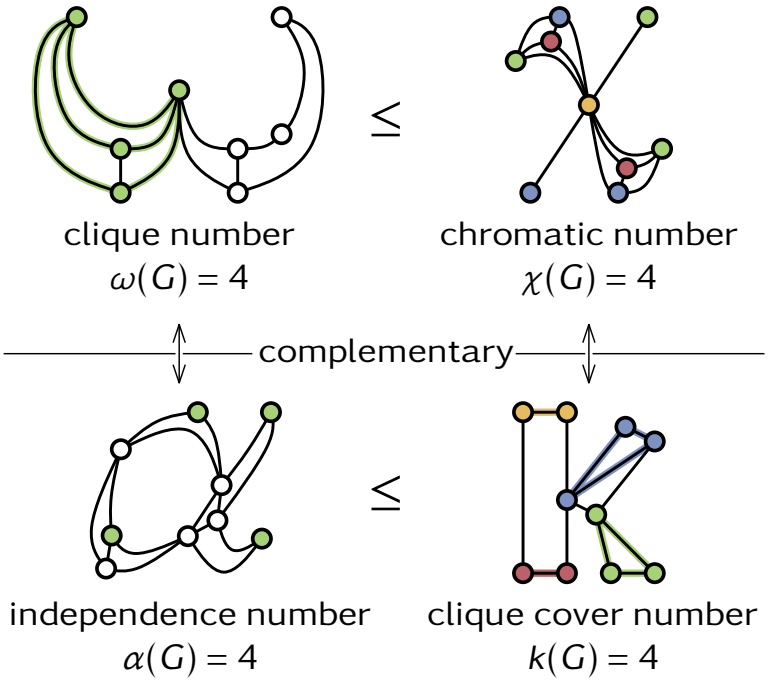
Algorithmic Graph Theory and Perfect Graphs

Lecture notes for the summer semester 2018

Prof. Dr. Ignaz Rutter

Summer semester 2018



Foreword

This write-up was first published in parallel to the lecture "Algorithmic Graph Theory" in the winter semester 2014/15 at the Karlsruhe Institute of Technology (KIT). For the lecture in the winter semester 2020/21 at the University of Passau, the write-up will be revised and extended accompanying the lecture. It is based on the book "Algorithmic Graph Theory and Perfect Graphs" by Martin C. Golumbic, published by Elsevier as part of the series *Annals of Discrete Mathematics*. However, from time to time I have omitted topics, streamlined them, and formulated proofs differently. The lecture notes do not claim to be complete and are in particular *not* intended to replace lecture attendance. Even though I try very hard, I cannot guarantee the correctness of the content. At this point, I would also like to thank all lecture participants in the winter semester 2014/15 at KIT and Thomas Bläsius, who held the accompanying exercise, who proofread the write-up very thoroughly, which made it possible to eliminate many errors, and who designed the fancy "Cheat Sheet" that adorns the title page.

Of course, further corrections and suggestions for improvement are always welcome, for example by email to `rutter@fim.uni-passau.de`.

Ignaz Rutter, 30 January 2021

Contents

Chapter 1

Intro

In this chapter, we first review some basic notions from graph theory and other fundamental areas. Subsequently, first examples of graph classes, graph representations and algorithmic problems are presented, which are prototypical for the rest of the lecture.

Generally, this script assumes familiarity with basic concepts of complexity theory (in particular O notation and NP-completeness) and graph theory. Nevertheless, some concepts are repeated for the sake of completeness and to unify the notation.

1.1 Fundamental definitions and notation

Sets

Two sets A and B are *disjoint* if $A \cap B = \emptyset$. In this case for $C = A \cup B$ we also write $C = A + B$ and mean that $C = A \cup B$ and $A \cap B = \emptyset$ holds.

Relations

An *binary relation* on a set X is a mapping $R\colon X \to P(X)$, where $P(X)$ denotes the power set of X. Equivalently, R can be thought of as a set $\mathcal{R} \subseteq X \times X$ by defining

$$(x, y) \in \mathcal{R} \text{ exactly if } y \in R(x).$$

A relation can satisfy one or more of the following properties:

Symmetry: $x \in R(y) \Rightarrow y \in R(x) \quad \forall x, y \in X$,

Asymmetry: $x \in R(y) \Rightarrow y \notin R(x) \quad \forall x, y \in X$,

Reflexivity: $x \in R(x) \quad \forall x \in X$,

Irreflexivity: $x \notin R(x) \quad \forall x \in X$,

Transitivity: $z \in R(y), y \in R(x) \Rightarrow z \in R(x) \quad \forall x, y, z \in X$.

Such a binary relation is an *equivalence relation* if it is reflexive, symmetric and transitive. A binary relation R is a strict partial order if it is irreflexive and transitive is. (Show: Then R is also asymmetric).

Graphs

A graph G consists of a *vertex set* V and an irreflexive binary relation on V. Here, the binary relation is represented by the mapping $\mathrm{Adj}\colon V \to P(V)$. For a vertex $v \in V$, we denote by $\mathrm{Adj}(v)$ the set of vertices adjacent to v. Equivalently, the relation can be represented by a set $E \subseteq V \times V$. An ordered pair $(u, v) \in E$ is called *edge* from u to v. Obviously $(u, v) \in E$ holds exactly if $v \in \mathrm{Adj}(u)$.

The given definition of graphs ensures that a graph contains at most one edge (u, v) for every two vertices u and v. Moreover, it follows from irreflexivity that $(v, v) \notin E$ for all $v \in V$. Thus, the graphs occurring in the following are always directed, but do not contain multiple edges or so-called *loops* of the form (v, v).

For a vertex, we define its *neighborhood* $N(v) = \{v\} + \mathrm{Adj}(v)$. In what follows, we abbreviate the notation for edges $(u, v) \in E$ to $uv \in E$. Two edges are called *adjacent* if they have a common endpoint.

Let $G = (V, E)$ be a graph with vertex set V and edge set E. The graph $G^{-1} = (V, E^{-1})$ with

$$E^{-1} = \{(v, u) \mid (u, v) \in E\}$$

is called *inverse graph* of G. Obviously $uv \in E$ holds exactly if $vu \in E^{-1}$. The *symmetric closure* of G is the graph $\hat{G} = (V, \hat{E})$ with $\hat{E} = E \cup E^{-1}$. A graph is called *undirected* if its adjacency relation is symmetric, that is, if $E^{-1} = E$ holds.

A graph $H = (V, F)$ is *oriented* if its adjacency relation is asymmetric, that is, if $F \cap F^{-1} = \emptyset$. Moreover, if $F + F^{-1} = E$, then H (or F) is called *orientation* of G.

Let $G = (V, E)$ be an undirected graph. The *complement* of G is the graph $\overline{G} = (V, \overline{E})$ with

$$\overline{E} = \{(u, v) \in V \times V \mid u \neq v \text{ and } (u, v) \notin E\}.$$

That is, two vertices are adjacent in $\overline{G}$ exactly when they are not adjacent in G. A graph is *complete* if every pair u, v of vertices is adjacent with $u \neq v$. The complete graph with n vertices is denoted by K_n.

A graph $H = (V', E')$ is an *subgraph* of a graph $G = (V, E)$ if it holds that both $V' \subseteq V$ and $E' \subseteq E$. Two types of subgraphs are of particular importance, namely those generated by a set of vertices or edges. Let $S \subseteq E$ be a set of edges. The subgraph *spanned* by S is

the graph $H = (V_S, S)$ with $V_S = \{v \in V \mid \exists (u,v) \in S \vee \exists (v,u) \in S\}$. A vertex set $A \subseteq V$ *induces* a subgraph $G_A = (A, E_A)$ with

$$E_A = \{(u,v) \in E \mid u,v \in A\}.$$

Not every subgraph of a graph G is also an induced subgraph.

In the following, we define a set of subgraphs and associated metrics that provide insights into the structure of a graph. The computation of such structures and the associated metrics, as well as their interaction, will constitute a considerable part of the lecture. Thus, it is important to internalize these central concepts as well as the associated notation as much as possible. Let $G = (V, E)$ be an undirected graph in the following.

An *clique* is a set of vertices $A \subseteq V$ that induces a complete graph, that is $G_A \cong K_r$ for some $r \in \mathbb{N}$. Here $r = |A|$ is the *size* of the clique. A clique of size r is also called a r-*clique*. Obviously, a single vertex is always a 1-clique. A clique is called *(inclusion wise)maximal* if it is not contained in any larger clique. It is called *cardinality-maximal* if G does not contain any clique with larger cardinality.

We denote the size of a cardinality-maximal clique in G by $\omega(G)$; $\omega(G)$ is called *clique number* of G. A *clique cover* of size k is a partition of $V = A_1 + A_2 + \cdots + A_k$ such that each A_i is a clique. We denote by $k(G)$ the size of a smallest possible clique cover of G; $k(G)$ is called *clique cover number* of G.

An *independent set* $X \subseteq V$ is a set of vertices which are pairwise nonadjacent. We denote by $\alpha(G)$ the size of a cardinality-maximal independent set in G; $\alpha(G)$ is called *independence number* of G.

An *(proper)* c-*coloring* is a partition of the vertices $V = X_1 + X_2 + \cdots + X_c$ such that each X_i is an independent set. We can then color the vertices in X_i with color i, ensuring that nonadjacent vertices get different colors. We say that G is c-colorable. We denote by $\chi(G)$ the smallest c such that G is c-colorable; $\chi(G)$ is called *chromatic number* of G.

Obviously, $\omega(G) \leq \chi(G)$ (vertexes of the same clique need different colors) and $\alpha(G) \leq k(G)$ (vertexes of an independent set must be in different cliques of the cover) hold. Moreover, $\omega(G) = \alpha(\overline{G})$ and $\chi(G) = k(\overline{G})$ holds.

For any graph $G = (V, E)$ we define the *outdegree* $d^+(x) = |\operatorname{Adj}(x)|$ and the *indegree* $d^-(x) = |\{y \in V \mid x \in \operatorname{Adj}(y)\}|$. It is $\sum_{v \in V} d^+(v) = \sum_{v \in V} d^-(v) = |E|$. Vertices with indegree 0 are called *source*, vertices with output degree 0 are called *sink*. Vertices with in- and outdegrees 0 are called *isolated vertices*. For undirected graphs, $d^-(v) = d^+(v)$ holds for each vertex $v \in V$. We simply denote this number as the *degree* of v and write $d(v) = d^-(v) = d^+(v)$.

An *undirected path* of length l is a sequence of vertices $[v_0, \ldots, v_l]$ with $v_{i-1}v_i \in E$ or $v_iv_{i-1} \in E$ for $i = 1, 2, \ldots, l$. An *directed path* of length l is a sequence of vertices $[v_0, \ldots, v_l]$ with $v_{i-1}v_i \in E$ for $i = 1, 2, \ldots, l$. A path is called *simple* if it does not contain any vertex multiple times.

A graph G is *connected* if there is an undirected path between every two vertices. It

is *strongly connected* if between every two vertices x and y there exists a directed path from x to y. An *connected component* of an undirected graph is an inclusionwise maximal connected subgraph.

Similarly, an *(directed) cycle* of length $l+1$ is a sequence of vertices $[v_0, \dots, v_l, v_0]$ with $v_{i-1}v_i \in E$ for $i = 1, \dots, l$ and $v_l v_0 \in E$. A cycle is called *simple* if the v_i are pairwise different, i.e., if $v_i \neq v_j$ for $i \neq j$.

An *chord* in a simple cycle $[v_0, \dots, v_l, v_0]$ is an edge $v_i v_j \in E$ where i and j differ modulo $\ell + 1$ by more than 1.

An undirected graph $G = (V, E)$ is *bipartite* if its vertices can be partitioned into two disjoint independent sets, that is, $V = I_1 + I_2$ and each edge in E has an endpoint in I_1 and an endpoint in I_2. We then write $G = (I_1, I_2, E)$. Obviously, G is bipartite exactly if $\chi(G) \leq 2$. A bipartite graph $G = (I_1, I_2, E)$ is *complete* if for every two vertices $x \in I_1$ and $y \in I_2$ the edge xy is contained in E.

The following graphs will appear repeatedly.

K_n: the complete graph with n vertices.

C_n: the cycle of length n without chords.

P_n: the path of length n without chords.

$K_{n,m}$: the complete bipartite graph with $m + n$ vertices partitioned into independent sets of sizes n and m.

$K_{1,n}$ the *star graph* with $n + 1$ vertices.

mK_n: m disjoint copies of K_n.

1.2 Intersection graphs

Let $\mathcal{F}$ be a family of non-empty sets. The *intersection graph* of $\mathcal{F}$ is the graph with set of vertices $\mathcal{F}$ in which two vertices $F, F' \in \mathcal{F}$ are adjacent if and only if $F \cap F' \neq \emptyset$. If one allows $\mathcal{F}$ to contain arbitrary sets, then any graph can be obtained as the intersection graph of a suitable set system (exercise). In the the following we want to restrict the kind of sets which can be contained in $\mathcal{F}$. If such a restriction is given, then this gives rise to two natural questions, namely the *characterization* of those graphs which have such an intersection representation, and the *detection problem* of deciding for a given graph, whether it has a representation with the respective constraint.

In the following, some types of intersection representations are presented as examples. An intersection graph of intervals of a linearly ordered set (say $\mathbb{R}$) is called an *interval graph*; see Figure 1.1. If all intervals have length 1, it is a *unit interval graph.* A *proper interval graph* is the intersection graph of a family of intervals with the property that no

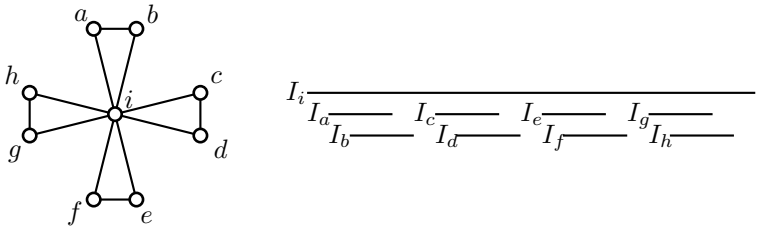


Figure 1.1: Windmill graph and corresponding interval representation.

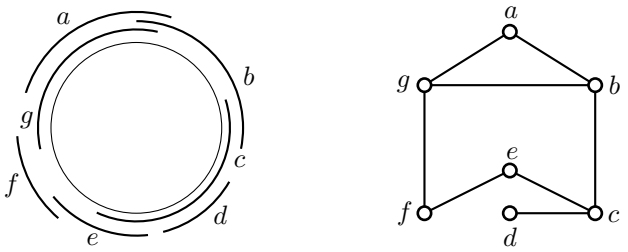


Figure 1.2: A circular-arc graph

interval properly contains another. It can be shown that unit interval graphs and proper interval graphs form the same class of graphs (exercise).

A relaxation of this concept is obtained by considering intervals on a finite line segment and identifying the end points of the line segment with each other, so that it forms a cycle. The intervals then become arcs. We now allow as sets also circular arcs which contain or pass over the connecting point, and thus obtain the set of *circular-arc graph*; see Figure 1.2. Analogous to the situation with interval graphs, a *proper circular-arc graph* is the intersection graph of a family of circular arcs such that none of the circular arcs properly contains another.

A completely different generalization of interval graphs is obtained by first considering that interval graphs are exactly intersection graphs of subpaths of a path. Starting from this, one can generalize the representation by considering, for example, path-in-tree or tree-in-tree intersection graphs.

A further completely different kind of graph is obtained by considering two parallel lines and choosing n points on each of the lines and n line segments, each connecting vertices on different lines, so that a matching is induced between the points. The intersection graph of the lines is called a *permutation graph*; see Figure 1.3. If, on the other hand, we have $2n$ vertices arbitrarily distributed on a cycle and a set of n line segments, each forming pairs of vertices, these line segments are chords of the cycle and we obtain so-called *circle-graphs*; see Figure 1.4.

Remark 1.1. One can show that the class of proper circular-arc graphs (i.e., no interval contains another) is a proper subset of the circular-arc graphs. That is, every proper circular-arc graph is also a circle graph, but there are also circle graphs which are not

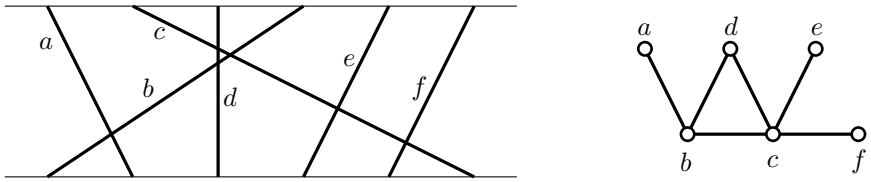


Figure 1.3: A permutation graph.

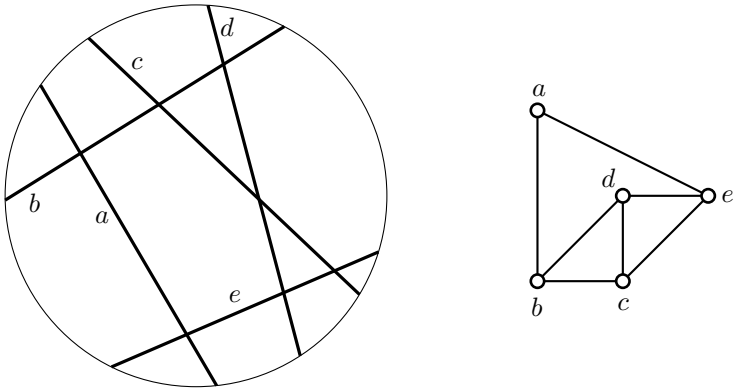


Figure 1.4: A circle-graph

(proper) circular-arc graphs.

1.3 A short outlook on interval graphs

An undirected graph is an *interval graph* if there is a bijective mapping of its vertices to a set $\mathcal{I}$ of intervals in $\mathbb{R}$ such that two vertices are adjacent if and only if the associated intervals have a non-empty intersection. We then call $\mathcal{I}$ an *interval representation* of G.

An application. The following problem arises when assigning rooms to lectures. There is a set of lectures, each with fixed times, and a set of rooms. The task now is to assign a room to each lecture so that at no time two or more lectures take place in the same room.

This problem can be modeled quite naturally as a graph problem. For this purpose, we consider the graph $\mathsf{G} = (\mathsf{V}, \mathsf{E})$, which contains one vertex per lecture and in which two lectures are connected if and only if they overlap with each other at some point in time. To obtain a valid room assignment, it is now sufficient to find a proper coloring of this graph, where the colors then correspond to the individual rooms.

Normally, such modeling is not particularly helpful, since it is NP-hard to decide whether a given number of colors is sufficient to find a proper coloring. In this case, however, it is easy to see that the constructed graph G has further structure; G is an interval graph,

since it has been defined precisely as the intersection graph of the time intervals in which the lectures take place. As it will turn out, the coloring problem can be solved efficiently on interval graphs.

Fundamental characteristics. Interval graphs have a number of basic properties that play a similar role for many other classes of intersection graphs.

Definition 1.2. A graph property is called *hereditary* if the fact that G has the property implies that any induced subgraph of G also has that property.

For example, the property of being an interval graph is a *hereditary property.*

Proposition 1.3. *An induced subgraph of an interval graph is an interval graph.*

Proof. Let $\{I_v\}_{v\in V}$ be an interval representation of $G = (V, E)$. Then $\{I_v\}_{v\in X}$ is an interval representation of the induced subgraph G_X. □

Remark 1.4. The above proof does not use in any way that the sets of the intersection representation are intervals. In fact all intersection graphs are hereditary families.

Definition 1.5. A graph is called *chordal* if every cycle with length greater than three has a chord.

Proposition 1.6. *Interval graphs are chordal.*

Proof. Let G be an interval graph with chordless cycle $[v_0, v_1, \ldots, v_{l-1}, v_0]$ with $l > 3$. Denote I_k the interval corresponding to vertex v_k. Choose $p_i \in I_{i-1} \cap I_i$ for $i = 1, \ldots, l-1$. Since I_{i-1} and I_{i+1} do not intersect, the p_i form a strictly increasing or strictly decreasing sequence. It follows that I_0 and I_{l-1} do not intersect; a contradiction to the existence of the edge v_0v_{l-1}. □

This gives us a first simple criterion to exclude certain graphs from being interval graphs; namely, if they are not chordal, such as the graph in Figure 1.5. On the other hand, there are graphs that are chordal but still have no interval representation, such as the graph in Figure 1.6. (Why?)

Definition 1.7. A graph is called *transitively orientable* if its edges can be oriented such that the resulting graph (V, F) satisfies the following conditions:

$$ab \in F \text{ and } bc \in F \quad \text{implies} \quad ac \in F \qquad \forall a, b, c \in V$$

An undirected graph which is transitive orientable is also called *comparabilitygraph.* Figure 1.7 shows two examples.

Proposition 1.8. *The complement graph of an interval graph is transitively orientable.*

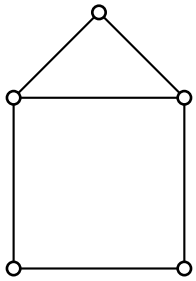


Figure 1.5: A non-chordal graph.

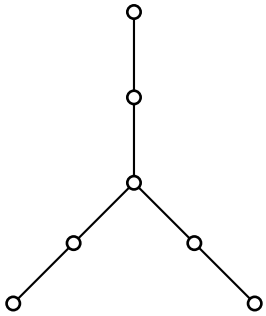


Figure 1.6: A chordal graph that is not an interval graph.

Proof. Let $\{I_v\}_{v \in V}$ be an interval representation of $G = (V, E)$. We define an orientation of the complement $\overline{G} = (V, \overline{E})$ as follows:

$$xy \in F \Leftrightarrow I_x < I_y \quad \forall xy \in \overline{E}$$

Here $I_x < I_y$ means that the interval I_x lies entirely to the left of the interval I_y. Obviously, the orientation F is transitive, since $I_x < I_y < I_z$ implies that $I_x < I_z$. So F is a transitive orientation of $\overline{G}$. □

Similar to the chordal graphs, however, there are again graphs whose complements are comparability graphs, but which are not interval graphs. So this condition is also necessary but not sufficient. As we will see later, however, the conditions are sufficient together.

Theorem 1.9 (Gilmore, Hoffman 1964)**.** *An undirected graph G is an interval graph if and only if it is chordal and its complement $\overline{G}$ is a comparability graph.*

Looking at the example graphs in figures 1.1, 1.3, 1.4, 1.5, and 1.7, we see that they can be colored with three colors. On the other hand, each of them contains a 3-clique. That is, for these graphs the clique number coincides with the chromatic number. This motivates the definitions of the following properties.

Definition 1.10. A graph G is called χ-perfect if for every induced subgraph G_A of G $\chi(G_A) = \omega(G_A)$ holds.

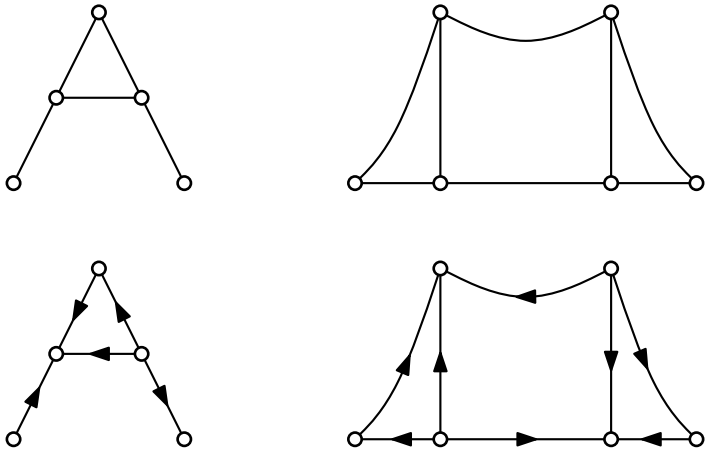


Figure 1.7: Comparability graphs with transitive orientations.

Definition 1.11. A graph G is called α-perfect if for every induced subgraph G_A of G $\alpha(G_A) = k(G_A)$ holds.

Chapter 2

Pefect Graphs

Let's consider again the following graph parameters:

$\omega(G)$, the *clique number* von G: the size of a largest complete subgraph of G.
$\chi(G)$, the *chomatic number* of G: is the minimum number of colors with which G can be properly colored. (equivalent: the minimum number of independent sets needed to cover all vertex of G).
$\alpha(G)$, the *independence number* von G: the maximum number of vertices in an independent set in G.
$k(G)$, the *clique-cover-number* of G: the minimum number of complete subgraphs needed to cover all the vertices of G.

The intersection of a clique and an independent set contains at most one element. Therefore,

$$\omega(G) \leq \chi(G)$$

and

$$\alpha(G) \leq k(G).$$

These inequalities are dual to each other, since $\alpha(G) = \omega(\overline{G})$ and $k(G) = \chi(\overline{G})$.

Now let $G = (V, E)$ be an undirected graph. In the following we consider graphs which satisfy the following properties

$$\omega(G_A) = \chi(G_A) \qquad \text{for all } A \subseteq V \tag{P1}$$

and

$$\alpha(G_A) = k(G_A) \qquad \text{for all } A \subseteq V. \tag{P2}$$

Such graphs are called *perfect*. According to the above duality, it is clear that a graph satisfies property (P1) exactly if its complement graph $\overline{G}$ satisfies property (P2). In the following we will prove a much stronger statement, namely that the statements (P1) and (P2) are equivalent.

2.1 The "'Perfect Graph Theorem"'

In this section we show that the statements (P1) and (P2) are equivalent. For the proof we consider the additional property

$$\omega(G_A)\cdot\alpha(G_A) \geq |A| \qquad \text{for all } A \subseteq V \tag{P3}$$

and show that it is equivalent to both (P1) and (P2). This property is motivated by the fact that, on the one hand, it is symmetric with respect to the formation of complements. On the other hand, the statement is not surprising, since it says that the maximal clique and the maximal independent set cannot both be small compared to the graph size. If both numbers were small, and if the graph G nevertheless satisfied (P1) and (P2), respectively, then G could be covered with a few small independent sets and with a few small cliques, respectively, and thus would itself be small.

An important role will be played by the so-called node multiplication, which allows us to "inflate" certain parts of the graph without changing its fundamental properties.

Definition 2.1. Let $G = (V, E)$ be a graph and v a vertex of G. The graph $G \circ v$ is the graph obtained from G by adding a new vertex v' which is connected to all neighbors of v.

Lemma 2.2. *Let $G = (V, E)$ be a graph. For $x \neq y \in V$ holds $(G \circ x) - y = (G - y) \circ x$.*

Proof. Exercise. □

Definition 2.3. In general, let $x_1, \ldots, x_n$ be the vertices of G and $h = (h_1, h_2, \ldots, h_n)$ be a vector with $h_i \in \mathbb{N}_0$. The graph $H = G \circ h$ is constructed by replacing each vertex x_i with an independent set of h_i vertices $x_i^1, \ldots, x_i^{h_i}$ and connecting x_i^s with x_j^t exactly when x_i and x_j are adjacent in G. We say that H is obtained by *vertex multiplication* from G.

Remark 2.4. The definition explicitly allows $h_i = 0$. In this case H contains no copy of x_i. In particular, any induced subgraph of G can be obtained by multiplication by an appropriate $0/1$ vector.

Lemma 2.5. *Let G be a graph with vertices $x_1, \ldots, x_n$. If $h \in \mathbb{N}_0^n$ is a vector with $h_i = 0$ and h' is the vector obtained from h by omitting the i-th component, then*

$$G \circ h = (G - x_i) \circ h'.$$

If h is a vector with $h_i > 0$ and $h' = h - e_i$ (e_i is the i-th unit vector), then

$$G \circ h = (G \circ x_i) \circ h'.$$

Proof. Exercise. □

Lemma 2.6. *Let* H *be a graph obtained by vertex multiplication from* G*. Then the following statements hold:*

(i) If G *satisfies property (P1), then* H *also satisfies (P1).*

(ii) If G *satisfies property (P2), then* H *also satisfies (P2).*

Proof. The proof is done by induction on the number of vertices. Obviously, the statement holds if G has only one vertex. We now consider a graph G and assume that statements (i) and (ii) hold for all graphs with fewer vertices than G. Let $H = G \circ h$. If one of the coordinates of h is zero, say $h_i = 0$, then H is obtained by vertex multiplication from $G - x_i$. Since G satisfies property (P1) [resp. property (P2)] so does $G - x_i$. Thus (i) and (ii) follow from the induction hypothesis.

So we can assume that for each coordinate $h_i \geq 1$ holds. Since any multiplication can be split into individual steps according to Lemma 2.5, it suffices to show the result for $H = G \circ x$. Let x' be the vertex added by multiplication by x. Since any proper induced subgraph of $G \circ x$ can be obtained by vertex multiplication from a proper induced subgraph of G, the respective statement holds by induction. Thus it suffices to show that $\omega(G \circ x) = \chi(G \circ x)$ (statement (i)), respectively, that $k(G \circ x) = \alpha(G \circ x)$ (statement (ii))

Suppose G satisfies (P1). Since x and x' are not adjacent, $\omega(G \circ x) = \omega(G)$ holds. Consider a coloring of G with $\omega(G)$ colors. Color x' with the color of x. We obtain a coloring of $G \circ x$ with $\omega(G \circ x)$ colors. So statement (i) holds for $G \circ x$.

Suppose G satisfies (P2). Then show that $\alpha(G \circ x) = k(G \circ x)$. Let $\mathcal{K}$ be a clique cover of G with $|\mathcal{K}| = k(G) = \alpha(G)$ and let $K_x \in \mathcal{K}$ be the clique with $x \in K_x$. We distinguish two cases.

Case 1: x *is contained in a cardinality-maximal independent set* S *of* G, i.e. $|S| = \alpha(G)$. Then $S \cup \{x'\}$ is an independent set of $G \circ x$, i.e. $\alpha(G \circ x) = \alpha(G) + 1$.

On the other hand, $\mathcal{K} \cup \{\{x'\}\}$ is a clique cover of $G \circ x$. Thus,

$$k(G \circ x) \leq k(G) + 1 = \alpha(G) + 1 = \alpha(G \circ x) \leq k(G \circ x).$$

Thus $\alpha(G \circ x) = k(G \circ x)$ holds.

Case 2: No cardinality-maximal independent set of G *contains* x. Then $\alpha(G \circ x) = \alpha(G)$ holds. For any cardinality-maximal independent set S in G and any clique $K \in \mathcal{K}$, $|S \cap K| = 1$ holds (exercise!). In particular, this holds for K_x. But since $x \notin S$ it even holds that every cardinality-maximal independent set S has exactly one element in common with the set $D = K_x \setminus \{x\}$. So $\alpha(G_{V-D}) = \alpha(G) - 1$ holds. This implies that

$$k(G_{V-D}) = \alpha(G_{V-D}) = \alpha(G) - 1 = \alpha(G \circ x) - 1.$$

If we now consider a clique cover of G_{V-D} of size $\alpha(G \circ x) - 1$ together with the additional clique $D \cup \{x'\}$, we obtain a clique cover of $G \circ x$. Thus, $k(G \circ x) = \alpha(G \circ x)$. □

Lemma 2.7. *Let* G *be an undirected graph for which every proper induced subgraph satisfies property (P2) and let* H *be a graph obtained by vertex multiplication from* G*. If* G *satisfies property (P3), then this is also true for* H*.*

Proof. We shall give a proof by contradiction. Let H be a graph with minimal number of vertices, which is formed by multiplying vertices of G but does not satisfy property (P3). Then it holds that

$$\omega(H)\alpha(H) < |X|, \tag{2.1}$$

where X denotes the set of vertices of H, but property (P3) holds for every proper induced subgraph of H.

As in the previous lemma, we can assume that each vertex has been multiplied by at least 1 and that a vertex u has been multiplied by $h \geq 2$. Let $U = \{u^1, \ldots, u^h\}$ be the vertices of H corresponding to u. Due to the minimality of H, H_{X-u^1} satisfies property (P3), so

$$\begin{aligned} |X| - 1 = |X - u^1| &\leq \omega(H_{X-u^1})\alpha(H_{X-u^1}) && \text{[as per (P3)]} \\ &\leq \omega(H)\alpha(H) \\ &\leq |X| - 1 && \text{[as per (2.1)]} \end{aligned}$$

So equality holds everywhere and we define

$$\begin{aligned} p &= \omega(H_{X-u^1}) = \omega(H), \\ q &= \alpha(H_{X-u^1}) = \alpha(H). \end{aligned}$$

Then $pq = |X| - 1$ holds. Since H_{X-U} can be obtained by vertex multiplication from $G - u$, by Lemma 2.6 H_{X-U} satisfies (P2). Thus, the graph H_{X-U} can be covered with q complete subgraphs $K_1, \ldots, K_q$ of H. Without loss of generality, we can assume that the K_i are pairwise disjoint and sorted by size in non-ascending order. It holds that

$$\sum_{i=1}^{q} |K_i| = |X - U| = |X| - h = pq - (h - 1).$$

Since $|K_i| \leq p$ at most $h - 1$ of the K_i can contribute less than p to the sum. So

$$|K_1| = |K_2| = \cdots = |K_{q-h+1}| = p.$$

Let H' be the subgraph of H induced by $X' = K_1 \cup \cdots \cup K_{q-h+1} \cup \{u^1\}$. Then

$$|X'| = p(q - h + 1) + 1 < pq + 1 = |X|.$$

Due to the minimality of H, it follows that

$$\omega(H')\alpha(H') \geq |X'|.$$

Further, it holds that $p = \omega(H) \geq \omega(H')$, so

$$\alpha(H') \geq |X'|/p > q - h + 1.$$

Let S' be an independent set of size $q - h + 2$ in H'. It holds that $u^1 \in S'$, otherwise S' would contain two vertices of a clique. But then $S = S' \cup U$ is an independent set with $q + 1$ vertices in H. This is a contradiction to the definition of q. □

With this all important preliminary work is done and we come to the proof of the "'Perfect Graph Theorem"'.

Theorem 2.8. *For an undirected graph* $G = (V, E)$ *the following statements are equivalent:*

$$\omega(G_A) = \chi(G_A) \qquad \text{for all } A \subseteq V \qquad \text{(P1)}$$
$$\alpha(G_A) = k(G_A) \qquad \text{for all } A \subseteq V \qquad \text{(P2)}$$
$$\omega(G_A) \cdot \alpha(G_A) \geq |A| \qquad \text{for all } A \subseteq V \qquad \text{(P3)}$$

Proof. For the proof, we can assume that the theorem holds for all graphs that have fewer vertices than G.

(P1)⇒(P3): Suppose G_A can be colored with $\omega(G_A)$ colors. Since there are at most $\alpha(G_A)$ vertices of each color, it follows $\omega(G_A)\alpha(G_A) \geq |A|$.

(P3)⇒(P1): Suppose $G = (V, E)$ satisfies property (P3). By induction, every proper subgraph of G satisfies property (P1)–(P3). Thus it suffices to show that $\omega(G) = \chi(G)$.

Suppose there was an independent set S in G with $\omega(G_{V-S}) < \omega(G)$. Then we could color G_{V-S} with $\omega(G) - 1$ colors and S with one additional color, obtaining a coloring with $\omega(G)$ colors, i.e. $\omega(G) = \chi(G)$.

Thus, we can assume that for any independent set S, the graph G_{V-S} contains a $\omega(G)$-clique $K(S)$. Let $\mathcal{S}$ be the set of all independent sets of G and note that $S \cap K(S) = \emptyset$. For each $x_i \in V$, let h_i be the number of cliques $K(S)$ containing x_i. Let $H = (X, F)$ be the graph obtained from G by multiplying x_i by h_i. From Lemma 2.7 it follows that

$$\omega(H)\alpha(H) \geq |X|.$$

On the other hand it holds that

$$|X| = \sum_{x_i \in V} h_i \tag{2.2}$$

$$= \sum_{S \in \mathcal{S}} |K(S)| = \omega(G)|\mathcal{S}|, \tag{2.3}$$

$$\omega(H) \leq \omega(G), \tag{2.4}$$

$$\alpha(H) = \max_{T \in \mathcal{S}} \sum_{x_i \in T} h_i \tag{2.5}$$

$$= \max_{T \in \mathcal{S}} \left[\sum_{S \in \mathcal{S}} |T \cap K(S)| \right] \tag{2.6}$$

$$\leq |\mathcal{S}| - 1 \tag{2.7}$$

Altogether it follows that

$$\omega(H)\alpha(H) \leq \omega(G)(|\mathcal{S}| - 1) < |X|$$

- a contradiction.

(P2)⇔(P3): From the already proven implications it follows that

$$G \text{ satisfies (P2)} \Leftrightarrow \overline{G} \text{ satisfies (P1)} \tag{2.8}$$

$$\Leftrightarrow \overline{G} \text{ satisfies (P3)} \Leftrightarrow G \text{ satisfies (P3).} \tag{2.9}$$

□

Corollary 2.9. *A graph* G *is perfect if and only if its compliment* $\overline{G}$ *is perfect.*

Corollary 2.10. *A graph* G *is perfect if and only if every graph* H *that can be obtained by multiplying vertices from* G *is perfect.*

2.2 p-Critical and Partitionable Graphs (⋆)

Attention: Sections marked with (⋆) have not been discussed in the lecture.

It would be desirable to have a characterization of perfect graphs by forbidden substructures (cf. Kuratowski's theorem for planar graphs). Since the property of being perfect is hereditary, a characterization by means of forbidden induced subgraphs lends itself to this. For this purpose, one is interested in the smallest possible non-perfect graphs. This motivates the following definition.

Definition 2.11. An undirected graph G is called p-*critical* if it is minimally imperfect, that is, G is not perfect, but any proper induced subgraph of G is perfect.

In particular, for a p-critical graph G

$$\alpha(G - x) = k(G - x) \quad \text{and} \quad \omega(G - x) = \chi(G - x)$$

holds for each vertex x. In the following, we will study the structure of such graphs in more detail and thus formulate a conjecture about their exact structure. The corresponding conjecture is called 'Strong Perfect Graph Conjecture" and has been proved by now (namely in 2006), it states that the p-critical graphs are exactly the odd length cycles and their complements. However, the whole proof covers about 170 pages and is thus not suitable for the lecture. Instead, we want to prove at least some structure results for p-critical graphs and thus make the conjecture plausible.

Theorem 2.12. *If G is a p-critical graph, then*

$$n = \alpha(G)\omega(G) + 1,$$

and for all vertices x of G,

$$\alpha(G) = k(G - x) \quad \textit{and} \quad \omega(G) = \chi(G - x).$$

Proof. By theorem 2.8, since G is p-critical, $n > \alpha(G)\omega(G)$ and $n-1 \leq \alpha(G-x)\omega(G-x)$ holds for all vertices x. So

$$n - 1 \leq \alpha(G - x)\omega(G - x) \leq \alpha(G)\omega(G) < n.$$

It follows $n - 1 = \alpha(G)\omega(G)$ as well as $\alpha(G) = \alpha(G - x) = k(G - x)$ and $\omega(G) = \omega(G - x) = \chi(G - x)$. □

We now raise to definition the properties found in theorem 2.12.

Definition 2.13. Let $\alpha, \omega \geq 2$ be arbitrary integers. An undirected graph G with n vertices is called (α, ω)-*partitionable* if $n = \alpha\omega + 1$ and further for each vertex x of G it holds that

$$\alpha = k(G - x), \qquad \omega = \chi(G - x).$$

.

Theorem 2.12 shows that every p-critical graph G is (α, ω)-partitionable with $\alpha = \alpha(G)$ and $\omega = \omega(G)$. In fact, a more general statement holds.

Remark 2.14. Let G be a (α, ω)-partitionable graph and x be any vertex in G. Then $G - x$ has exactly $\alpha\omega$ vertices, has chromatic number ω and clique cover number α. Thus, a ω coloring of $G - x$ partitions the vertices into ω independent sets, one of which must have at least size α. Similarly, a minimal clique cover of $G - x$ partitions the vertices into α cliques, one of which must have size at least ω.

Theorem 2.15. *Let G be a (α, ω)-partitionable graph. Let $\alpha = \alpha(G)$ and $\omega = \omega(G)$.*

Proof. Let $G = (V, E)$ be (α, ω)-partitionable. By remark 2.14, $\alpha \leq \alpha(G)$ and $\omega \leq \omega(G)$ hold. For the inverse, consider an independent set S of maximal size in G and let $y \in V - S$. Then S is a maximal independent set of $G - y$, so

$$\alpha(G) = |S| = \alpha(G - y) \leq k(G - y) = \alpha\,.$$

Thus $\alpha(G) \leq \alpha$.

For ω consider analogously a maximal clique K of G and $y \in V - K$. Then K is a maximal clique of $G - y$, so

$$\omega(G) = |K| = \omega(G - y) \leq \chi(G - y) = \omega\,.$$

It follows that $\omega(G) \leq \omega$. Altogether this yields $\alpha = \alpha(G)$ and $\omega = \omega(G)$. □

Theorem 2.15 shows that the numbers α and ω are uniquely determined for a partitionable graph. So in what follows we use only the term *partitionable* and assume that $\alpha = \alpha(G)$ and $\omega = \omega(G)$.

Remark 2.16. The class of p-critical graphs is a proper subset of the partitionable graphs, which in turn is a proper subset of the imperfect graphs.

Lemma 2.17. *If G is a partitionable graph with n vertices, the following statements hold:*

(i) *G contains a set of n maximal cliques $K_1, K_2, \ldots, K_n$ that cover each vertex of G exactly $\omega(G)$ times;*

(ii) *G contains a set of n maximal independent sets $S_1, S_2, \ldots, S_n$ that cover each vertex of G exactly $\alpha(G)$ times; and*

(iii) *$K_i \cap S_j = \emptyset$ if and only if $i = j$.*

Proof. Choose a maximal clique K of G and for each vertex $x \in K$ choose a minimal clique cover $\mathcal{K}_x$ of $G - x$. By remark 2.14, all cliques in $\mathcal{K}_x$ are cliques of size ω. Now let A be the $n \times n$ matrix whose first row is the characteristic vector of clique K, and whose other rows are the characteristic vectors of all cliques in $\mathcal{K}_x$ for each $x \in K$. (Note: these are ω distinct clique covers $\mathcal{K}_x$, each consisting of α cliques. So, together with the row for K, that's a total of $\omega\alpha + 1 = n$ rows).

Each vertex $y \notin K$ is covered exactly once by $\mathcal{K}_x$ for each $x \in K$. Each vertex $z \in K$ is covered once by K and exactly once by each $\mathcal{K}_x$ with $x \neq z$. Thus, each vertex is covered exactly ω times. For each row a_i of A, denote K_i the corresponding clique with characteristic vector a_i. We can write property (i) as $\mathbf{1}A = \omega\mathbf{1}$, where $\mathbf{1}$ denotes the row vector where all entries are 1. (Note: it remains to show that the K_i are pairwise distinct).

We first proceed with the construction of S_i. For each i, choose a vertex $v \in K_i$ and denote by $\mathcal{S}$ an optimal coloring (minimal overlap with independent sets!) of $G - v$. By remark 2.14, $\mathcal{S}$ consists of ω disjoint independent sets of size α. Obviously, each of these sets contains at most one vertex of K_i, and moreover $K_i - v$ has only $\omega - 1$ vertices, so there exists an $S_i \in \mathcal{S}$ with $K_i \cap S_i = \emptyset$. On the other hand, if $j \neq i$, then $K_j \cap S_i \neq \emptyset$ holds (so $|K_j \cap S_i| = 1$), since otherwise $\mathcal{S}$ would not be a valid coloring. This proves property (iii).

Now let b_i denote the characteristic vector of S_i and let B be the $n \times n$ matrix whose rows are the b_i ($i = 1, \ldots, n$). According to the above observations it holds that

$$a_i b_j^T = \begin{cases} 0 & i = j \\ 1 & \text{otherwise.} \end{cases}$$

We now consider the matrix AB^T. According to the equation above, $AB^T = J - I$, where I is the unit matrix and J denotes the matrix whose entries are all 1. This matrix is not singular, so neither are A and B, from which it follows again that the K_i and the S_i must be pairwise distinct. From this property (i) follows directly.

Furthermore,

$$\mathbf{1}B = \mathbf{1}BA^T(A^T)^{-1} = \mathbf{1}(J - I)(A^T)^{-1} = [(n-1)/\omega]\mathbf{1} = \alpha\mathbf{1}\,,$$

which proves property (ii). □

In fact, it can be shown that the cliques and independent sets constructed in Lemma 2.17 contain all maximal independent sets and all maximal cliques of G.

Lemma 2.18. *A partitionable graph G contains exactly n maximal cliques and n maximal independent sets.*

Proof. Let A and B be the matrices whose rows form the characteristic vectors of the cliques and independent sets from Lemma 2.17. We assume that c is the characteristic vector of a maximal clique in G. We will show that c is a row of A.

First, $A(\omega^{-1}J - B^T) = \omega^{-1}AJ - AB^T = J - AB^T = I$ and consequently

$$A^{-1} = \omega^{-1}J - B^T\,.$$

Let us consider a solution t of the equation $tA = c$. Then

$$t = cA^{-1} = c(\omega^{-1}J - B^T) = \omega^{-1}cJ - cB^T = \mathbf{1} - cB^T\,.$$

. Consequently, t is a $(0,1)$ vector.

Moreover,

$$t\mathbf{1}^T = (\mathbf{1} - cB^T)\mathbf{1}^T = n - cB^T\mathbf{1}^T = n - \alpha c\mathbf{1}^T = n - \alpha\omega = 1\,.$$

So t is a unit vector and hence c is a row of A. The statement for maximal independent sets follows analogously. □

Theorem 2.19. *Let* G *be an undirected graph with* n *vertices and let* $\alpha = \alpha(G)$ *and* $\omega = \omega(G)$*. The graph* G *is partitionable if and only if the following conditions hold:*

(i) $n = \alpha\omega + 1$*;*

(ii) G *has exactly* n *maximal cliques and* n *maximal independent sets;*

(iii) each vertex of G *is contained in exactly* ω *maximal cliques and in exactly* α *maximal independent sets;*

(iv) each maximum clique intersects all but one maximum independent set and vice versa.

Proof. If G is partitionable, then the statements (i)–(iv) follow from the previous lemmas. Thus it remains to show that a graph G satisfying properties (i)–(iv) is partitionable.

According to conditions (ii)–(iv), we can set up the matrices A and B as before, so that

$$AJ = JA = \omega J, \quad BJ = JB = \alpha J, \quad AB^T = J - I\,.$$

Now let x_i be a vertex of G and let h_i^T be the corresponding column in A. Because $A^TB = J - I$, it follows that $h_iB = 1 - e_i$. That is, h_i selects ω rows of B (which are independent sets!) that together cover $G - x_i$. Thus, $\chi(G - x_i) \leq \omega$ holds. Analogously, it can be shown that $k(G - x_i) \leq \alpha$ for all x_i. But since $n - 1 = \alpha\omega$, $\chi(G - x_i) = \omega$ and $k(G - x_i) = \alpha$ must hold. So G is partitionable. □

Corollary 2.20. *Every* p*-critical graph satisfies properties (i)–(iv) of theorem 2.19.*

2.3 The strong "Perfect Graph Conjecture" ($\star$)

Attention: Sections marked with ($\star$) have not been discussed in the lecture.

Der ungerade Kreis C_{2k+1} ist für $k \geq 2$ kein perfekter Graph. Offensichtlich gilt $\alpha(C_{2k+1}) = k$ und $k(C_{2k+1}) = k + 1$ (bzw. $\omega(C_{2k+1}) = 2$ und $\chi(C_{2k+1}) = 3$). Andererseits ist jeder echte Subgraph von C_{2k+1} perfekt. Die ungeraden Kreise (und damit auch deren Komplemente) sind also p-kritisch.

Die starke "Perfect Graph Conjecture" besagt, dass die obigen Graphen in der Tat die einzigen p-kritischen Graphen sind, ein Graph also genau dann perfekt ist, wenn er keinen der obigen Graphen als induzierten Subgraph enthält. Die Vermutung lässt sich auf folgende äquivalente Arten formulieren:

SPGC$_1$ Ein ungerichteter Graph ist genau dann perfekt wenn er keinen induzierten Subgraph enthält, der zu C_{2k+1} oder $\overline{C}_{2k+1}$ mit $k \geq 2$ isomorph ist.

SPGC$_2$ Ein ungerichteter Graph G ist genau dann perfekt wenn jeder ungerade Kreis der Länge mindestens 5 in G oder $\overline{G}$ eine Sehne hat.

SPGC$_3$ Die einzigen $\mathfrak{p}$-kritischen Graphen sind C_{2k+1} und $\overline{C}_{2k+1}$ für $k \geq 2$.

Im englischen heißen die Graphen C_{2k+1} und $\overline{C}_{2k+1}$ auch *odd hole* und *odd anti-hole*. Wir haben bereits gesehen, dass $\mathfrak{p}$-kritische Graphen ein außerordentlich hohes Maß an Symmetrie aufweisen. Insbesondere, besagt der vorherige Abschnitt dass, wenn G ein $\mathfrak{p}$-kritischer Graph ist und $\alpha = \alpha(G)$ und $\omega = \omega(G)$, dann müssen die folgenden Aussagen erfüllt sein.

1. $n = \alpha\omega + 1$
2. Jeder Knoten liegt in genau ω maximalen Cliquen.
3. Jeder Knoten liegt in genau α maximalen unabhängigen Mengen.
4. G hat genau n maximale Cliquen (der Größe ω).
5. G hat genau n maximale unabhängige Mengen (der Größe α).
6. Die maximalen Cliquen und maximalen unabhängigen Mengen lassen sich mit $K_1, \ldots, K_n$ und $S_1, \ldots, S_n$ durchnummerieren, sodass $|K_i \cap S_j| = 1 - \delta_{ij}$, wobei δ_{ij} das Kronecker-Delta bezeichnet.

Offensichtlich ist jeder $\mathfrak{p}$-kritische Graph zusammenhängend. Es ist leicht zu sehen, dass C_n der einzige zusammenhängende Graph mit n Knoten ist, für den gilt, dass $\omega = 2$ und der genau n Kanten (maximale Cliquen) hat und für den jeder Knoten zu genau zwei Kanten inzident ist (jeder Knoten ist in genau zwei maximalen Cliquen enthalten). Damit können wir die Vermutung nochmals neu formulieren.

SPGC$_4$ Es gibt keinen $\mathfrak{p}$-kritischen Graphen mit $\alpha > 2$ und $\omega > 2$.

In Anlehnung an die Familien C_{2k+1} und $\overline{C}_{2k+1}$ lässt sich eine weitere Familie von partitionierbaren Graphen konstruieren. Der Graph C_n^d hat Knoten $v_1, \ldots, v_n$ und v_i und v_j sind durch eine Kante verbunden genau dann wenn i und j sich um höchstens d unterscheiden. Dabei werden die Indices modulo n betrachtet. Man kann leicht sehen, dass $C_{\alpha\omega+1}^{\omega-1}$ ein (α, ω)-partitionierbarer Graph ist. Für $\omega = 2$ erhalten wir so die Familie C_{2k+1}, für $\alpha = 2$ die Familie $\overline{C}_{2k+1}$. Allerdings sind diese Graphen für $\alpha > 2$ und $\omega > 2$ nicht $\mathfrak{p}$-kritisch.

Theorem 2.21 (ohne Beweis.)**.** *Für $\alpha \geq 3$ und $\omega \geq 3$ sind die partitionierbaren Graphen $C_{\alpha\omega+1}^{\omega-1}$ nicht $\mathfrak{p}$-kritisch.*

Damit lässt sich die Vermutung nochmals äquivalent umformulieren.

SPGC$_5$ Ist G $\mathfrak{p}$-kritisch mit $\alpha(G) = \alpha$ und $\omega(G) = \omega$, so enthält G einen induzierten Subgraphen, der isomorph ist zu $C_{\alpha\omega+1}^{\omega-1}$.

Inzwischen konnte die obige Vermutung gezeigt werden. Allerdings ist der Beweis extrem umfangreich.

Theorem 2.22 ("'Strong perfect graph theorem"', 2006)**.** *Ein ungerichteter Graph ist genau dann perfekt wenn er keinen induzierten Subgraph enthält, der zu* C_{2k+1} *oder* $\overline{C}_{2k+1}$ *mit* $k \geq 2$ *isomorph ist.*

Chapter 3

Chordal Graphs

One of the first graph classes for which it could be shown that all graphs contained therein are perfect were the chordal graphs. Indeed, the insight that chordal graphs satisfy both property (P1) and property (P2) provided the basis for the conjecture that the two properties may be equivalent. In this sense, the study of chordal graphs forms the basis for the study of perfect graphs.

Definition 3.1. A graph G is *chordal* if every cycle of length greater than 3 has a chord.

This is equivalent to the requirement that G has no induced subgraph that is isomorphic to C_n with $n > 3$. Thus it is obvious that being chordal is a hereditary property. In chapter 1 it was already mentioned that interval graphs are a special class of chordal graphs. Figure 3.1 shows two graphs of which the left one is chordal, but the right one is not.

3.1 Characterization

In the following, we will elaborate a characterization of chordal graphs, which will also be crucial from an algorithmic point of view.

Definition 3.2. A vertex x of G is called *simplicial* if its nonadjacent vertices $\mathrm{Adj}(x)$

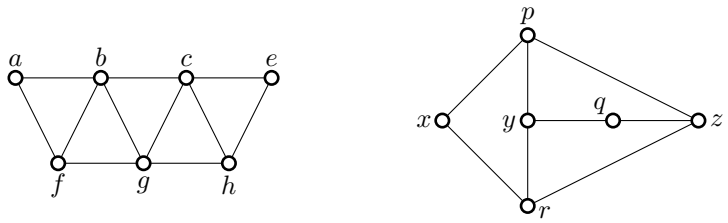


Figure 3.1: Two example graphs, the left one is chordal, the right one is not.

form a complete subgraph of G, that is, $\mathrm{Adj}(x)$ is a (not necessarily maximal) clique.

In the left graph in Figure 3.1, the vertices a and e are the only simplicial vertices. The right graph has no simplicial vertex.

Definition 3.3. Let $G = (V, E)$ be an undirected graph and let $\sigma = [v_1, \ldots, v_n]$ be a vertex ordering. The order σ is a *perfect elimination scheme* if each vertex v_i is a simplicial vertex of the induced subgraph $G_{\{v_i,\ldots,v_n\}}$. Equivalently, each of the sets $X_i = \{v_j \in \mathrm{Adj}(v_i) \mid j > i\}$ is complete.

Obviously, the right graph in Figure 3.1 does not have a perfect elimination scheme, since it does not have a simplicial vertex. For the left graph, for example, $[a, f, b, g, c, h, e]$ is a perfect elimination scheme. However, this is by no means unique; in fact, the left graph has 96 different perfect elimination schemes.

EA set of vertices $S \subseteq V$ is an *(vertex)separator* for two nonadjacent vertices a and b (also called an a–b-*separator*) if a and b are in different connected components of $G - S$.

Lemma 3.4. *If G is a chordal graph, then every (inclusion) minimal vertex separator induces a complete subgraph of G.*

Proof. Let S be an (inclusion) minimal a–b-separator of G with connected components G_A and G_B of G_{V-S} containing a and b, respectively. Due to the minimality of S, each vertex of S has a neighbor in A and a neighbor in B. For each pair of vertices $x, y \in S$, there is a path $[x, a_1, \ldots, a_r, y]$ and a path $[y, b_1, \ldots, b_t, x]$ with $a_i \in A$ and $b_i \in B$ such that these paths have minimal length. Then the cycle $[x, a_1, \ldots, a_r, y, b_1, \ldots, b_t, x]$ is simple and has length at least 4; thus it must have a chord. However, since S is a separator, $a_i b_j i s \notin E$. Moreover, it follows from the minimality of r and t that $a_i a_j \notin E$ and $b_i b_j \notin E$ for $i < j - 1$. Therefore, the only possible chord is $xy \in E$. □

Remark 3.5. It further follows that $r = t = 1$, which in turn implies that for every two vertices $x, y \in S$ there are vertices in A and B that are nonadjacent to both x and y.

In fact, it can even be shown that there is at least one vertex in each of A and B that is adjacent to all vertices of the separator. (Exercise!)

Lemma 3.6. *Every chordal graph $G = (V, E)$ has one simplicial vertex. Moreover, if G is not a clique, then G has two nonadjoint simplicial vertices.*

Proof. If G is complete, the statement holds trivially. So suppose that G contains two nonadjoint vertices a and b and the statement holds for all graphs that have fewer vertices than G. Let S be an (inclusion) minimal vertex separator for a and b and let G_A and G_B be the connected components of G_{V-S} containing a and b, respectively. By induction, either G_{A+S} contains two nonadjacent simplicial vertices (in which case at least one of them is in A, since S induces a complete set by Lemma 3.4) or G_{A+S} is itself complete, and thus every vertex in A is simplicial in G_{A+S}. Since $\mathrm{Adj}(A) \subseteq A + S$, a

simplicial vertex of G_{A+S} in A is also simplicial in G. Using the same argument, one shows that B also contains a simplicial vertex. □

Theorem 3.7. *Let G be an undirected graph. The following statements are equivalent:*

(i) G *is chordal.*

(ii) G *has a perfect elimination scheme.*

(iii) Each (inclusion) minimal vertex separator induces a complete subgraph of G.

Proof. **(i) ⇒ (iii):** is exactly the statement of Lemma 3.4.

(iii) ⇒ (i): Let $[a, x, b, y_1, y_2, \ldots, y_k, a]$ with $k \geq 1$ be a simple cycle of $G = (V, E)$. Every minimal a–b-separator contains x and y_i for some i. But then $xy_i \in E$ is a chord of the cycle.

(i) ⇒ (ii): According to Lemma 3.6, a chordal graph G has a simplicial vertex x. Since the graph $G_{V-\{x\}}$ is chordal and smaller than G, by induction hypothesis it has a perfect elimination scheme. The concatenation of $[x]$ and this scheme then yields a perfect elimination scheme for G.

(ii) ⇒ (i): Let C be a simple cycle of G and let x be a vertex of C with minimal index in a perfect elimination scheme. Since $|\operatorname{Adj}(x) \cap C| \geq 2$, the simpliciality of x at the time of its removal guarantees a chord in C.

□

Remark 3.8. If G has a perfect elimination scheme and v is a simplicial vertex of G, then there also exists a perfect elimination scheme of G starting with v.

3.2 Erkennung chordaler Graphen

Theorem 3.7 and Remark 3.8 together directly yield an efficient algorithm for detecting chordal graphs by iteratively searching for and removing a simplicial vertex. If all vertices can be removed from the graph in this way, the deletion order results in a perfect elimination scheme. However, if the algorithm stops before this, we have found an induced subgraph that does not contain any simplicial vertex. Thus, the graph is not chordal. The main problem with this approach, which goes back to Fulkerson and Gross, is efficiency. For example, finding a simplicial vertex is easy to implement in $O(n + m^2)$ time. However, using the procedure for n steps results in a total running time of $O(n^2 + nm^2)$. In the following, we will give a more efficient algorithm that solves the recognition problem for chordal graphs in linear time.

The basic intuition for this is the following. The above procedure for constructing a perfect elimination scheme lets us choose between at least two simplicial vertices at

Eingabe : Unordered graph $G = (V, E)$.
Ausgabe : Vertex ordering σ.
1 Assign each vertex the label $\emptyset$;
2 **für** $i \leftarrow n$ **bis** 1 **tue**
3 | select an unnumbered vertex v with largest label;
4 | $\sigma(i) \leftarrow v$;
5 | **for** each unnumbered vertex $w \in \mathrm{Adj}(v)$ add i to label(w);
6 **Ende**

Algorithmus 1 : LexBFS

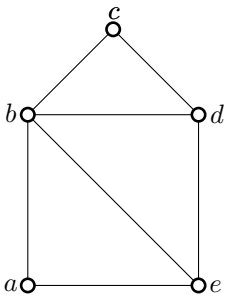


Figure 3.2: Chordal graph

any time. Thus, we are free to choose one vertex v_n to keep until the end. Similarly, however, we can now choose another vertex v_{n-1} adjacent to v_n and keep it for the $n-1$th position. If we were to continue in this way, we would create a perfect elimination scheme 'backwards', so to speak. This is exactly the idea followed by the algorithm presented below, which goes back to Rose, Tarjan and Lueker.

The algorithm uses a so-called *lexicographic breadth-first search*. This is a breadth-first search in which a special rule is used to resolve ambiguities in the selection of the next vertex to visit. Here, the queue used to implement breadth-first search contains a set of unordered subsets (vertices whose order is arbitrary are in the same set). Occasionally, these sets are refined, i.e., a set is split into multiple subsets and an ordering of the subsets is specified, but no two sets are ever reordered. The procedure is described in Algorithm 1. Here, each vertex has a *label*, which consists of a set of numbers listed in descending order. The vertices are then sorted lexicographically in the queue according to their labels.

Figure 3.3 exemplifies the flow of a lexicographic breadth-first search on the graph from Figure 3.1. It turns out that the returned sequence $[c, d, e, b, a]$ is a perfect elimination scheme. This is not a coincidence and in fact is always the case when the input graph is chordal.

For the proof, let $L_i(x)$ denote the label of x at the time of choosing the i-th vertex (line 3 in the algorithm). The index i is decreased in each run. Thus, the order of the vertices with different labels is always preserved. The following properties apply:

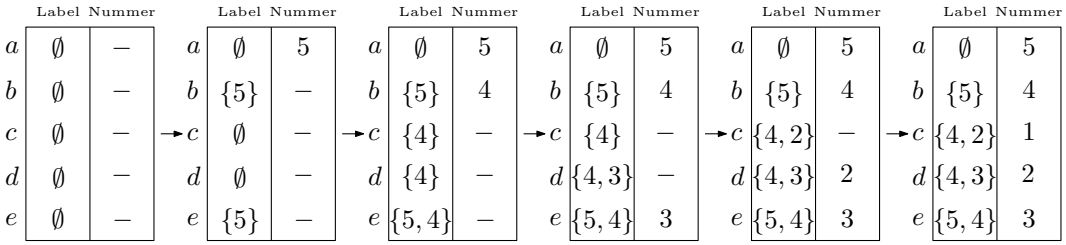


Figure 3.3: Chordal graph

(L1) $L_i(x) \leq L_j(x) \quad \forall j \leq i$

(L2) $L_i(x) < L_i(y) \Rightarrow L_j(x) < L_j(y) \quad \forall j \leq i$

(L3) If $\sigma^{-1}(a) < \sigma^{-1}(b) < \sigma^{-1}(c)$ and $c \in \mathrm{Adj}(a) - \mathrm{Adj}(b)$ hold, then there exists a $d \in \mathrm{Adj}(b) - \mathrm{Adj}(a)$ with $\sigma^{-1}(c) < \sigma^{-1}(d)$.

Property (L1) states that labels may only become larger, but not smaller. Property (L2) states that once a label of a vertex is smaller than the label of another vertex, it remains so in subsequent iterations. Property (L3) describes a condition under which there must exist a suitable vertex d that was numbered before c. The reasoning here is as follows: If there were no such vertex, a and b would have to have the same label at the time c was numbered. However, due to the adjacency of a to c and the fact that b is not adjacent to c, this would result in a getting a larger label than b and, since according to the other properties the order of the labels does not change, would thus have to be numbered before b. This is a contradiction.

We now show that lexicographic breadth-first search can be used to detect chordal graphs.

Theorem 3.9. *An undirected graph $G = (V, E)$ is chordal if and only if the vertex ordering σ computed by Algorithm 1 is a perfect elimination scheme.*

Proof. For $|V| = n = 1$ the statement obviously holds. We assume that the statement holds for all graphs with less than n vertices and consider an ordering σ obtained by applying algorithm 1 to a chordal graph G. By induction, it suffices to show that $x = \sigma(1)$ is a simplicial vertex of G.

So we assume that x is not simplicial. That means there exist vertices $x_1, x_2 \in \mathrm{Adj}(x)$ with $x_1x_2 \notin E$. Here we choose these vertices such that x_2 is maximal with respect to σ. Now the vertices $x = x_0, x_1, x_2$ just satisfy the property (L3), so there is a vertex x_3 adjacent to x_1 but not to x_0. Again, we choose x_3 maximal with respect to σ with this property. Since by assumption the graph is chordal, the edge x_2x_3 cannot exist either. We now proceed inductively. We have constructed a sequence $x_0, x_1, x_2, x_3, \ldots, x_m$ satisfying the following properties.

(1) $x_0x_i \in E \Leftrightarrow i \leq 2$,

(2) For $i, j > 0$ it holds that $x_i x_j \in E \Leftrightarrow |i - j| = 2$,

(3) $\sigma^{-1}(x_0) < \sigma^{-1}(x_1) < \sigma^{-1}(x_2) < \cdots < \sigma^{-1}(x_m)$,

(4) For $j \geq 3$ x_j is the maximal vertex with respect to σ with $x_{j-2}x_j \in E$ but $x_{j-3}x_j \notin E$.

We have just constructed the sequence for $m = 3$ and now show how to continue the construction. The vertices x_{m-2}, x_{m-1} and x_m satisfy property (L3), so there exists a vertex x_{m+1} with $\sigma^{-1}(x_{m+1}) > \sigma^{-1}(x_m)$ which is adjacent to x_{m-1} but not x_{m-2}. We choose x_{m+1} with this property such that it is maximal with respect to σ and show that the sequence $x_0, x_1, \ldots, x_{m+1}$ satisfies the above properties.

First, x_{m+1} is by definition adjacent to x_{m-1} and not adjacent to x_{m-2}. If x_{m+1} were adjacent to x_{m-3}, we could apply property (L3) to x_{m-3}, x_{m-2} and x_{m+1} and obtain a vertex larger than x_{m+1} and adjacent to x_{m-2} but not to x_{m-3}. This would contradict the maximality of x_m in (4). Thus, it follows that x_{m+1} is not adjacent to x_{m-3}. If x_{m+1} were adjacent to one of the vertices $x_0, \ldots, x_{m-4}$ or x_m, then (1) and (2) would yield a cycle without chords. But this would contradict the fact that G is chordal.

So the above inductive procedure continues indefinitely. But since the graph is finite, this leads to a contradiction. The vertex x is therefore simplicial. From this it follows directly that for chordal graphs a perfect elimination scheme is found. The converse follows from theorem **??**. □

To obtain an efficient algorithm for recognizing chordal graphs with linear runtime, two things are still necessary. First, we need to show that lexicographic breadth-first search can be implemented with linear runtime, and second, we need a procedure to decide in linear runtime whether a given vertex ordering is a perfect elimination scheme.

3.2.1 Implementation of LexBFS

Instead of computing the labels of the vertices, we simply keep the vertices ordered by their labels. For this purpose, we define for a label l the set S_l of the not yet numbered vertices with label l. We use a queue Q that contains the non-empty sets S_l in the correct order. Each of the sets is stored as a doubly linked list. Initially, Q contains only a single set, namely $S_\emptyset = V$. For each vertex v, we store a pointer $\mathrm{SET}(v)$ that points to the set containing v. Moreover, v also stores a pointer to its position in the list of $S_{label(v)}$, so that v can be removed from the set in constant time. The mappings σ and σ^{-1} are represented by simple arrays. In addition, to control the flow, we need a flag $\mathrm{FLAG}(S_l)$ for each of the sets, initially set to 0, and a list FIXLIST of sets S_l for which cleanup is required.

With such a data structure, it is easy to find a vertex with maximal label. We choose as v in line 3 of algorithm 1 any vertex in the last set in Q and remove this vertex from $\mathrm{SET}(v)$. If $\mathrm{SET}(v)$ becomes empty as a result of this, we remove it from Q. We

```
1  für each unnumerated vertices w ∈ Adj(v) tue
2  |  wenn FLAG(SET(w)) = 0 dann
3  |  |  Create new set S′ and add it to Q directly after SET(w);
4  |  |  FLAG(SET(w)) ← 1; FLAG(S′) ← 0; Pointer to SET(w) in FIXLIST;
5  |  Ende
6  |  Let S′ be the set directly after SET(w) in Q;
7  |  Remove w from SET(w); Add w to S′;
8  |  SET(w) ← S′
9  Ende
10 für each set S in FIXLIST tue
11 |  FLAG(S) ← 0;
12 |  wenn S leer dann
13 |  |  Remove S from Q;
14 |  Ende
15 |  Remove S from FIXLIST;
16 Ende
```

Algorithmus 2 : Pseudocode of the update step from line 5 of algorithm 1.

now set σ and σ^{-1} for the vertex v. We then iterate over the neighbors of v and move them into the new resulting sets. Here, a vertex $w \in \mathrm{Adj}(v)$ with label l is to be moved into the set $S_{l \cdot \sigma^{-1}(v)}$ that immediately follows S_l in Q. To avoid creating the same set multiple times, we use FLAG(SET(w)). We check if the FLAG is set, and only if the flag is still 0, we create the set and set the flag to 1. After that, the target set is in Q directly after SET(w) in any case. To guarantee that all flags are reset at the end, we also add a pointer to SET(w) in FIXLIST. The actual moving of the vertices is easily done in constant time. Following this, we iterate once over the sets of FIXLIST, set their associated flags back to 0, and remove sets that have become empty from Q. Algorithm 2 shows the procedure as pseudo-code. It is not hard to see that the entire update takes only $O(|\mathrm{Adj}(v)|)$ time. It follows that the total running time of the detection algorithm is in $O(|V| + |E|)$.

Theorem 3.10. *Algorithm 1 can be implemented to compute a lexicographic breadth-first search on an undirected graph* $G = (V, E)$ *with* $O(|V| + |E|)$ *time and space.*

3.2.2 Identification of perfect elimination schemes

We can now use lexicographic breadth-first search to determine a possible vertex ordering for a graph that is a perfect elimination scheme exactly when the graph is chordal. Thus, we must finally check whether the computed vertex ordering is a perfect elimination scheme. To prevent this step from becoming a bottleneck in the identification of chordal graphs, we need an algorithm that decides for a graph with a given vertex ordering whether it is a perfect elimination scheme. Of course, one can be naive about this and always check whether, for each vertex, its subsequent neighbors form a clique. However,

this requires at least quadratic running time. On the other hand, it is quite obvious that this will check many neighborhoods multiple times.

The basic idea behind the following algorithm is to go through the vertices in ascending order of numbering, check some neighborhoods, and then remove them. To check if the neighborhood of the first vertex x forms a clique, we assign the neighbor y of x with the smallest number the task to check if it is connected to all neighbors of x. This is not done until the vertex y is processed. At this point, vertex y may have been given the task of checking neighborhoods by other vertices.

Let us now consider the time at which y is processed. If y is not adjacent to all neighbors of x, then the subsequent neighbors of x do not form a clique and the vertex ordering is not a perfect elimination scheme. Otherwise, the neighborhood of x without y in the residual graph is a subset of the neighborhood of y. Thus, if y is simplicial, this implies the simpliciality of x. Thus, no further steps are necessary to treat x.

Algorithm 3 describes the procedure as pseudeocode. The removal of processed vertices is not done explicitly, but implicitly by only considering vertices whose index in σ is larger than the current iteration.

```
 1 Boolean procedure PERFEKT(σ)
 2 Beginn
 3 |  for each vertex v do A(v) ← ∅;
 4 |  für i ← 1 bis n − 1 tue
 5 |  |  v ← σ(i);
 6 |  |  X ← {x ∈ Adj(v) | σ^{-1}(v) < σ^{-1}(x)};
 7 |  |  If X = ∅ then go to line 10;
 8 |  |  u ← σ(min{σ^{-1}(x) | x ∈ X});
 9 |  |  concatinate X − {u} to A(u);
10 |  |  wenn A(v) − Adj(v) ≠ ∅ dann
11 |  |  |  return false;
12 |  |  Ende
13 |  Ende
14 |  return true;
15 Ende
```

Algorithmus 3 : Procedure to check if a given vertex ordering σ is a perfect elimination scheme.

Arrays are used to evaluate σ and σ^{-1} in constant time, doubly linked lists are used for $\mathrm{Adj}(v)$ and $A(v)$. The jump in line 7 is performed exactly $j-1$ times, where j denotes the number of connected components. The set $A(u)$ may contain duplicates. Algorithm 4 shows how to perform the test in line 10 in $O(|\mathrm{Adj}(v)| + |A(v)|)$ time, using an array of size n whose entries are initially 0.

```
1 for w ∈ Adj(v) do TEST(w) ← 1;
2 für w ∈ A(v) tue
3 |   wenn TEST(w) = 0 dann
4 |   |   return non-empty;
5 |   Ende
6 Ende
7 for w ∈ Adj(v) do TEST(w) ← 0;
8 return empty;
```

Algorithmus 4 : Adjacency test in line 10 of algorithm 3.

Thus the entire algorithm runs with time and memory proportional to

$$|V| + \sum_{v \in V} |\operatorname{Adj}(v)| + \sum_{u \in V} |A(u)|\,.$$

Here $|A(u)|$ denotes the size of the list $A(u)$ at the time u is processed. It is not hard to see that the middle summand dominates the last summand, and therefore the last two summands are in $O(|E|)$. Thus, the algorithm has linear running time.

Theorem 3.11. *Algorithm 3 correctly checks whether the input ordering σ is a perfect elimination scheme. The algorithm can be implemented with running time and memory requirements proportional to $|V| + |E|$.*

Proof. The statements about the complexity have already been proven. It remains to show the correctness of the procedure.

The algorithm returns the statement "'false'" in the $\sigma^{-1}(u)$-th iteration exactly when there are three vertexes u, v, w ($\sigma^{-1}(v) < \sigma^{-1}(u) < \sigma^{-1}(w)$) where u in line **??** is defined by the $\sigma^{-1}(v)$-th iteration, and $u, w \in \operatorname{Adj}(v)$ but u is not adjacent to w.

Obviously, in the case where the answer is "'false"', there is no perfect elimination scheme. Conversely, suppose that σ is not a perfect elimination scheme and yet the algorithm returns 'true'. Let v be the vertex of maximum index $\sigma^{-1}(v)$ such that $X = \{w \mid w \in \operatorname{Adj}(v)$ and $\sigma^{-1}(v) < \sigma^{-1}(w)\}$ is not complete; i.e., the last vertex in the ordering that is not simplicial at the time of its removal. Let u be the vertex of X defined in line 8 by the $\sigma^{-1}(v)$-th iteration. Then (in line 9) $X - \{u\}$ is added to $A(u)$. Since the algorithm does not terminate, each vertex $x \in X - \{u\}$ is adjacent to u. Furthermore, due to the maximality of $\sigma^{-1}(v)$, the vertex u is simplicial, so in particular every two vertices from $X - \{u\}$ are adjacent. So X is complete, which contradicts the assumption. □

Corollary 3.12. *The recognition problem for chordal graphs can be solved in linear time.*

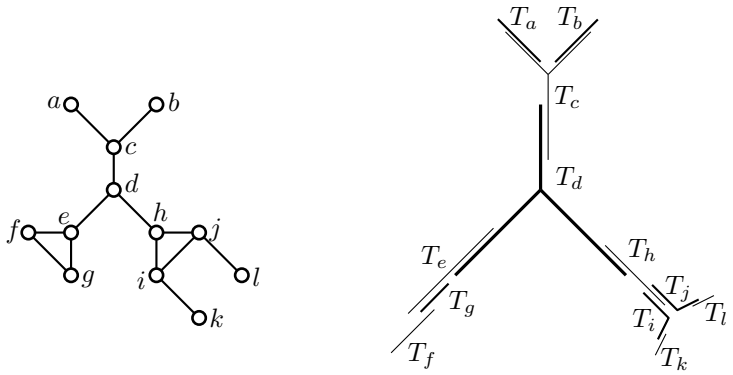


Figure 3.4: Chordal graph and a representation as a intersection graph of subtrees of a tree.

3.3 Chordal Graphs as Intersection Graphs

As we saw in chapter 1, interval graphs form a real subset of chordal graphs. This naturally leads to the question of whether chordal graphs can be described as intersection graphs of a topological family, which are somewhat more general than intervals of a straight line. In this section we show that a graph is chordal if and only if it is an intersection graph of a family of subtrees of a tree; see Figure 3.4.

A family $\{T_i\}_{i\in I}$ of subsets of a set T satisfies the so-called *Helly property* if the fact that for a subset $J \subseteq I$ it holds that $T_i \cap T_j \neq \emptyset$ for all $i, j \in J$ implies that $\cap_{j\in J} T_j \neq \emptyset$.

Proposition 3.13. *A family of subtrees of a tree satisfies the Helly property.*

Proof. We assume $T_i \cap T_j \neq \emptyset$ for all $i, j \in J$. We first consider the case where there are three vertices a, b, c of T such that for every two of the vertices there exists a tree T_j with $j \in J$ containing both of them. Let S be the set of indices s such that T_s contains at least two of the three points, and let P_1, P_2, P_3 be the simple paths connecting a to b, b to c, and c to a. Since T is a tree, $P_1 \cap P_2 \cap P_3 \neq \emptyset$ follows. However, each $T_s (s \in S)$ contains at least one of the paths P_i. So

$$\bigcap_{s\in S} T_s \supseteq P_1 \cap P_2 \cap P_3 \neq \emptyset\,.$$

We now prove the proposition by induction. We assume the proposition

$$T_i \cap T_j \neq \emptyset \text{ for all } i, j \in J \Rightarrow \bigcap_{j\in J} T_j \neq \emptyset$$

holds for index sets J of size at most k. The statement holds for $k = 2$.

Consider a family of subtrees $\{T_{i_1}, \ldots, T_{i_{k+1}}\}$. According to the induction hypothesis, vertices a, b, c of T exist such that

$$a \in \bigcap_{j=1}^{k} T_{i_j}\,, \quad b \in \bigcap_{j=2}^{k+1} T_{i_j}\,, \quad c \in T_{i_1} \cap T_{i_{k+1}}\,.$$

Each of the trees T_{i_j} contains at least two of the vertices a, b, c. Using the statement from the first paragraph of the proof, it follows $\bigcap_{j=1}^{k+1} T_{i_j} \emptyset$. □

Theorem 3.14. *Let $G = (V, E)$ be an undirected graph. The following statements are equivalent:*

(i) G *is chordal.*

(ii) G *is an intersection graph of a family of subtrees of a tree.*

(iii) There is a tree $T = (\mathcal{K}, \mathcal{E})$ whose vertex set $\mathcal{K}$ are the maximal cliques of G, such that each of the induced subgraphs $T_{\mathcal{K}_v}$ $(v \in V)$ is connected (i.e., a subtree), where $\mathcal{K}_v$ is the set of cliques in $\mathcal{K}$ containing v.

Proof. (iii) ⇒ (ii) We assume there is a tree $T = (\mathcal{K}, \mathcal{E})$ satisfying the properties in (iii). Let $v, w \in V$. The vertices v and w are adjacent in G exactly if there exists a maximal clique $A \in \mathcal{K}$ containing both vertices. This is again the case exactly if $\mathcal{K}_v \cap K_w \neq \emptyset$, i.e. exactly if $T_{\mathcal{K}_v} \cap T_{\mathcal{K}_w} \neq \emptyset$. So G is intersection graph of family of subtrees $\{T_{\mathcal{K}_v} \mid v \in V\}$.

(ii) ⇒ (i) Let $\{T_v\}_{v \in V}$ be a family of subtrees of a tree T such that $vw \in E$ exactly if $T_v \cap T_w \neq \emptyset$.

We assume that G contains a cycle $[v_0, v_1, \ldots, v_{k-1}, v_0]$ with $k > 3$ that has no chord. Let T_i denote the tree corresponding to v_i. Then $T_i \cap T_j \neq \emptyset$ holds exactly if i and j differ modulo k by at most 1. (In the following, all indices are implicitly computed modulo k).

Choose a point a_i from $T_i \cap T_{i+1}$ for $i = 0, \ldots, k-1$. Let b_i be the last common point of the unique paths from a_i to a_{i-1} and from a_i to a_{i+1}. These paths lie in T_i and in T_{i+1}, respectively; thus b_i lies in $T_i \cap T_{i+1}$. Let P_{i+1} be the simple path connecting b_i to b_{i+1}. Obviously, $P_i \subseteq T_i$ holds. So $P_i \cap P_j = \emptyset$ holds if i and j differ modulo k by more than 1. Moreover, $P_i \cap P_{i+1} = \{b_i\}$ for $i = 0, \ldots, k-1$. Consequently, $\bigcup_i P_i$ is a simple cycle in T. This contradicts the definition of a tree.

(i) ⇒ (iii) The proof is done by induction on the size of G. We assume that the theorem holds for all graphs that have fewer vertices than G.

If G is complete, T is a single vertex and the statement is trivial. If G is not connected with connected components $G_1, \ldots, G_k$, then by induction assumption for each of the graphs G_i there exists a corresponding tree T_i satisfying the assertions of (iii). We connect any vertex of T_i to any vertex of T_{i+1} to obtain the asserted tree satisfying property (iii) for G.

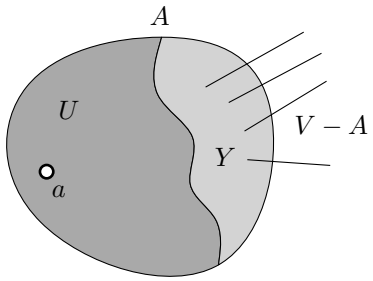


Figure 3.5: Illustration of the clique A, its two subsets U and Y, and its complement $V - A$.

In the following, we consider the case where G is neither complete nor disjoint. Let a be a simplicial vertex of G and let $A = \{a\} \operatorname{Adj}(a)$. Obviously, A is a maximal clique of G.

Let $U = \{u \in A \mid \operatorname{Adj}(u) \subset A\}$ and $Y = A - U$. It holds that $a \in U$; see figure 3.5. Since G is not complete, there are vertices in $V - A$, and moreover, since G is connected, and vertices in U are not adjacent to vertices in $V - A$, Y cannot be empty either. Thus the sets U, Y and $V - A$ are not empty.

We now consider the induced subgraph $G' = G_{V-U}$, which is chordal and has fewer vertices than G. By induction hypothesis, let T' be a tree whose vertex set $\mathcal{K}'$ is the set of maximal cliques of G' such that for each vertex in $V - U$ the vertex set $\mathcal{K}'_v = \{X \in \mathcal{K}' \mid v \in X\}$ induces a connected subgraph (subtree!) of T'.

Remark. If Y is a maximal clique of G', then $\mathcal{K} = \mathcal{K}' + \{A\} - \{Y\}$, otherwise $\mathcal{K} = \mathcal{K}' + \{A\}$ holds.

Let B be a maximal clique of G' containing Y. We distinguish two cases depending on whether $B = Y$ or not.

Case 1. If $B = Y$, we obtain T from T' by renaming the vertex B to A.

Case 2. If $B \neq Y$, we obtain T from T' by attaching the new vertex A as a leaf to the vertex B.

In both cases $\mathcal{K}_u = \{A\}$ for all $u \in U$ and $\mathcal{K}_v = \mathcal{K}'_v$ for all $v \in V - A$. By premise, these sets each induce a subtree of T. So we only need to check the sets $\mathcal{K}_y$ with $y \in Y$. In case 1, $\mathcal{K}_y = \mathcal{K}'_y + \{A\} - \{B\}$, which induces the same subtree as $\mathcal{K}'_y$, since we only renamed one vertex. In case 2, $\mathcal{K}_y = \mathcal{K}'_y + \{A\}$, which obviously induces a subtree. Thus T is the tree we are looking for and the theorem is proven. □

3.4 Perfectness

We will now show that chordal graphs are perfect. For this we use the fact that vertex separators in chordal graphs form cliques (Lemma 3.4).

Theorem 3.15. *Let S be a separator of a connected undirected graph $G = (V, E)$ and let $G_{A_1}, \ldots, G_{A_t}$ be the connected components of G_{V-S}. If S is a clique,*

$$\chi(G) = \max_i \chi(G_{S+A_i})$$

and

$$\omega(G) = \max_i \omega(G_{S+A_i}) .$$

Proof. Obviously, $\chi(G) \geq \chi(G_{S+A_i})$ holds for all i. On the other hand, we can initially color G_S arbitrarily, and extend the chosen coloring (where each vertex has its own color) to a coloring for each of G_{S+A_i} that uses at most $\chi(G_{S+A_i})$ colors. Overall, this gives a coloring of G with at most $\max_i \chi(G_{S+A_i})$ colors.

Similarly, $\omega(G) \geq \omega(G_{S+A_i})$ certainly holds for all i. On the other hand, no clique X in G can contain vertices from A_i and A_j with $j \neq i$, since S is a separator. Therefore, every clique X must be completely contained in one of the subgraphs G_{S+A_i}, so equality follows. □

Corollary 3.16. *Let S be a separator of a connected graph $G = (V, E)$ and let $G_{A_1}, \ldots, G_{A_t}$ be the connected components of G_{V-S}. If S is a clique and each subgraph G_{S+A_i} is perfect, then G is perfect.*

Proof. We assume that the statement holds for all graphs which have fewer vertices than G. Thus $\omega(G') = \chi(G')$ holds for all proper induced subgraphs G' of G and it suffices to show that $\omega(G) = \chi(G)$.

For each of the subgraphs $\chi(G_{S+A_i}) = \omega(G_{S+A_i})$ holds due to perfectness, so in particular $\max_i \chi(G_{S+A_i}) = \max_i \omega(G_{S+A_i})$. Thus, by the previous theorem, $\chi(G) = \omega(G)$ holds. □

Theorem 3.17. *Chordal graphs are perfect.*

Proof. Let G be a chordal graph. We show the statement by induction on the number of vertices and assume that the statement holds for all graphs that have fewer vertices than G. We can assume without loss of generality that G is connected but not a clique. Then there is a separator S in G that decomposes G into connected components $G_{A_1}, \ldots, G_{A_t}$ with $t \geq 2$. Each of the graphs $G_{S+A_1}, \ldots, G_{S+A_t}$ is chordal and hence perfect by induction hypothesis. By lemma 3.4 S forms a clique and thus by corollary 3.16 G is also perfect. □

3.5 Algorithms for chordal graphs

In the following, we will give efficient algorithms for the problems Coloring, Clique, Independent Set and Clique Cover on chordal graphs. In the following, let $G = (V, E)$ be a chordal graph and σ be a perfect elimination scheme for G. The first observation, going back to Fulkersson and Gross, is that any maximal clique of the form $\{v\} \cup X_v$, where

$$X_v = \{x \in \mathrm{Adj}(v) \mid \sigma^{-1}(v) < \sigma^{-1}(x)\}.$$

On the one hand, each of the sets $\{v\} \cup X_v$ is complete. On the other hand, if A is any maximal clique, we consider the first vertex w from A in the vertex ordering σ. Then $A = \{w\} \cup X_w$. In particular, the next result follows from this.

Proposition 3.18. *A chordal graph with n vertices has at most n inclusion maximal cliques. Equality holds if and only if the graph has no edges.*

It is easy to modify algorithm 3 such that in each step the set $\{v\} \cup X_v$ is output as well. However, not all of these sets are maximal. To decide which of these cliques are indeed maximal, it suffices to observe that the clique $\{v\} \cup X_v$ is *not* maximal exactly if in a previous step all vertices in X_v were added to $A(v)$ at once. (Proof: Exercise!)

Since σ is a perfect elimination scheme, at each step the set of vertices added to $A(v)$ is a subset of X_v. Thus, it is sufficient to store the size of the largest sets added in this way. By comparing this number with the size of $|X_v|$, we can then decide, when processing the vertex v, whether $\{v\} \cup X_v$ is a maximal clique and therefore must be output. Since chordal graphs are perfect, the size of the largest clique is also equal to the chromatic number. Algorithm 5 shows the pseudocode for this procedure.

Theorem 3.19. *Algorithm 5 correctly computes the chromatic number and all maximal cliques of a chordal graph $G = (V, E)$ in $O(|V| + |E|)$ time.*

The proof can be carried out similarly to the proof of theorem 3.11. (Exercise!)

Next, we deal with the computation of $\alpha(G)$. Since G is perfect, $\alpha(G) = k(G)$, and we also want to specify an independent set and a clique cover of this size at the same time.

To do this, we inductively define a sequence of vertices $y_1, y_2, \ldots, y_t$ by $y_1 = \sigma(1)$ and $y_i = \sigma(\min\{\sigma^{-1}(v) \mid \sigma^{-1}(v) > \sigma^{-1}(y_i), v \notin X_{y_1} \cup X_{y_2} \cup \cdots \cup X_{y_{i-1}}\})$ as the first successor of y_i in the vertex ordering σ, which is not contained in any of the sets X_{y_j} with $j < i$. In particular,

$$V = \{y_1, \ldots, y_t\} \cup X_{y_1} \cup \cdots \cup X_{y_t}.$$

The following theorem holds.

Theorem 3.20. *The set $\{y_1, \ldots, y_t\}$ is a maximal independent set of G and the set of sets $Y_i = \{y_i\} \cup X_{y_i}$ $(i = 1, 2, \ldots, t)$ is a minimum clique cover of G.*

```
1  Procedure CLIQUES(σ)
2  Beginn
3      χ ← 1;
4      for each vertex ν do S(ν) ← 0
5      für i ← 1 bis n tue
6          ν ← σ(i);
7          X ← {x ∈ Adj(ν) | σ^{-1}(ν) < σ^{-1}(x)};
8          If Adj(ν) = ∅ then print {ν};
9          If X = ∅ then go to line 16;
10         u ← σ(min{σ^{-1}(x) | x ∈ X});
11         S(u) ← max{S(u), |X| − 1};
12         wenn S(ν) < |X| dann
13             print {ν} ∪ X;
14             χ ← max{χ, 1 + |X|};
15         Ende
16     Ende
17     print "The chromatic number is " χ;
18 Ende
```

Algorithmus 5 : Procedure to enumerate all maximum cliques and calculate the chromatic number.

Proof. The set $\{y_1, y_2, \ldots, y_t\}$ is independent, since $y_j y_i \in E$ with $j < i$ implies that $y_i \in X_{y_j}$, which would contradict the definition of y_i. So $\alpha(G) \geq t$ holds. On the other hand, each of the sets $Y_i = \{y_i\} \cup X_{y_i}$ is a clique, and therefore $\{Y_1, \ldots, Y_t\}$ is a cover of G with cliques. So $\alpha(G) = k(G) = t$ holds, and we have indeed found a maximal independent set and a minimal clique cover. It is not difficult to implement the algorithm with linear running time. □