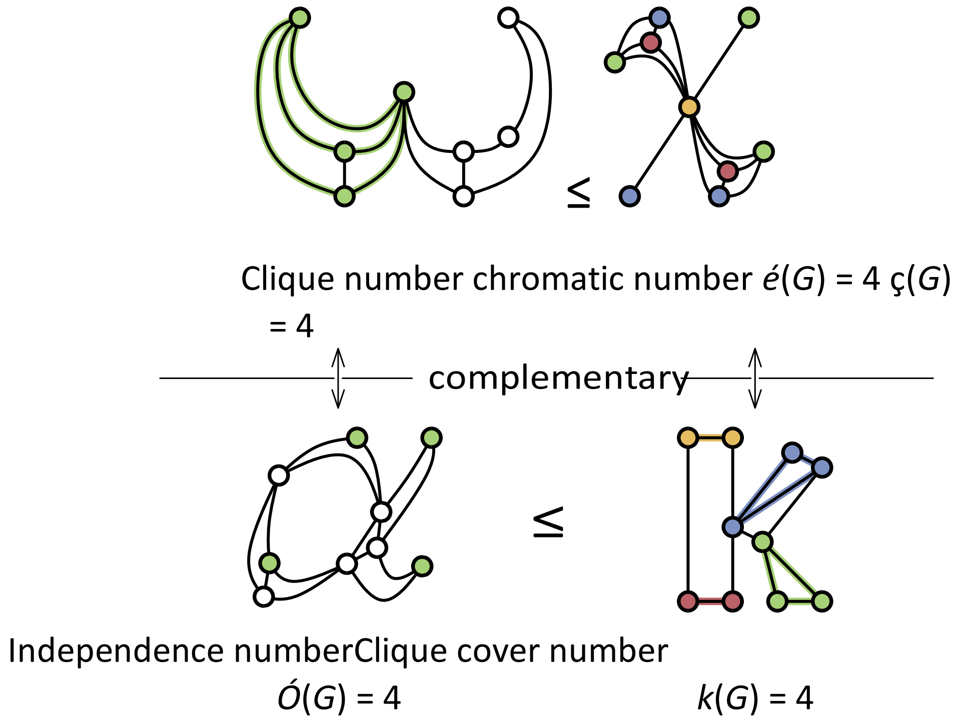
Algorithmic Graph Theory and Perfect Graphs

Note-taking for the lecture in the summer semester 2018

Prof. Dr. Ignaz Rutter

Summer semester 2018


≤
Clique number chromatic number *é*(*G*) = 4 ç(*G*) = 4
complementary
≤
Independence numberClique cover number
Ó(*G*) = 4
k(*G*) = 4

Foreword

This paper was first written 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 summer term 2018 at the University of Passau, the paper will be revised and extended in parallel to the lecture. It is mainly based on the book "Algorithmic Graph Theory and Perfect Graphs" by Martin C. Golumbic, published by Elsevier in the series *Annals of Discrete Mathematics*. However, I have occasionally omitted topics, streamlined, and worded proofs differently. The lecture notes do not claim to be complete and in particular are *not* meant to replace lecture attendance. Even if I try 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 2014/2015 at KIT and Thomas Bläsius, who held the associated exercise, 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, March 13, 2018

Table of Contents

Chapter 1

Introduction

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.

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

1.1Basic definitions and notation

Quantities

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

Relations

power set of X. Equivalently, R can be taken to be a set→R ⊆ X×X A *binary relation* on a set X is a mapping R: X P(X), where P(X) can be the by defining

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

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

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

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

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. (Show: Then R is also asymmetric).

Graphs

A graph G consists of a *set of nodes* V and an irreflexive binary relation For a node v ∈ V , we

denote by Adj(v) the set of adjacents→ to⊆ × v on V. Here, the binary relation is represented

by the mapping Adj: V P(V).

Node. Equivalently, the relation can be represented by a set E V V . An ordered pair (u,v) ∈ E

is called an *edge* from u to v. Obviously, (u,v) ∈ E holds exactly if v ∈ Adj(u).

The present definition of graphs ensures that a graph contains at most one edge (u,v) for every two nodes u and v. Moreover, it follows from irreflexivity that (v,v) ∉ E for all v ∈ V. Thus, the graphs occurring in the following are always directed, but contain neither multiple edges nor so-called *loops of* the form
(v,v).

For a node, we define its *neighborhood* N(v) = {v} + Adj(v). In the following we shorten the notation for edges (u,v) ∈ E to uv ∈ 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 the *inverse graph* of G. Obviously uv ∈ E holds exactly if vu ∈ E^{-1}. The *symmetric closure of* G is the graph G^ = (V,E^) with E^ = E ∪ 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 ∩ F^{-1} = ∅. 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

$$\overline{E} = \{(u,v) \in V \times V \mid u =6 \quad v \text{ and } (u,v) \in/ E\}.$$

That is, two nodes are adjacent in $\overline{G}$ if and only if they are not adjacent in G. A graph is *complete* if every pair u,v of nodes with u =6 v is adjacent. The complete graph with n nodes is ${}_n$ denoted by K.

A graph H = (V^0,E^0) is a *subgraph of* a graph G = (V,E) if it holds that both V^0 ⊆ V and E^0 ⊆ E. Two types of subgraphs are of particular importance, namely those generated by a set of nodes or edges. Let S ⊆ E be a set of edges. The subgraph *spanned* by S is the graph H = (V_S,S) with V_S = {v ∈ V | ∃(u,v) ∈ S ∨ ∃(v,u) ∈ S}. A set of nodes A ⊆ 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 their 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. In the following, let G = (V,E) be an undirected graph.

A *clique* is a set of nodes A ⊆ V that induces a complete graph, that is $G_A \overset{\sim}{=} K_r$ for an r ∈ N. Where r = |A| is the *size of* the clique. A clique of size r is also called an r-clique. Obviously, a single node is always a 1-clique. A clique is called *(inclusion-)maximal* if it is not contained in any larger clique. It is called *cardinality-maximal* if G contains no clique of larger cardinality.

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

An *independent set* $X \subseteq V$ is a set of nodes that are pairwise non-adjoint. We denote by α(G) the size of a cardinality-maximal independent set in G; α(G) is called the *independence number of* G.

A *(real)* c-coloring is a partition of nodes $V = X_1 + X_2 + --- + X_c$ such that each X_i is an independent set. We can then color the nodes in X_i with color i, ensuring that adjoint nodes get different colors. We say that G is c-colourable. We denote by χ(G) the smallest c such that G is c-colourable; χ(G) is called the *chromatic number of* G.

Obviously, $\omega(G) \leq \chi(G)$ (nodes of the same clique require different colors) and $\alpha(G) \leq k(G)$ (nodes of an independent set must be in different

Cliques of the overlap lie). Moreover, $\omega(G) = \alpha(\overline{G})$ and $\chi(G) = k(\overline{G})$ holds.

For any graph G = (V,E), we define the *output degree* $d^+(x) = |\operatorname{Adj}(x)|$ and the *input degree* d^-

P (x) = $|\{y \in V \mid x \in \operatorname{Adj}(y)\}|$. It holds that Pv $_{\in V}$ $d^+(v)$ = $_{v \in V}$ d-(v) = |E|. Nodes with input

degree 0 are called *source*, nodes with output degree 0 are called *sink*. Nodes with input and output degree 0 are called *isolated nodes*. For undirected graphs, $d^-(v) = d^+(v)$ for each node $v \in V$. 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 nodes $[v_0,...,v_l]$ with $v_{i-1} v_i \in E$ or $v_i v_{i-1} \in E$ for i = 1,2,...,l. A *directed path of* length l is a sequence of nodes $[v_0,...,v_l]$ with $v_{i-1}v_i \in E$ for i = 1,2,...,l. A path is called *simple* if it does not contain any node more than once.

A graph G is *connected* if there is an undirected path between every two nodes. It is *strongly connected* if there exists a directed path from x to y between every two vertices x and y. A *coherence component of* an undirected graph is an inclusion-maximal coherent subgraph.

Analogously, a *(directed) circle of* length l+1 is a sequence of nodes $[v_0,...,v_l,v_0]$ with $v_{i-1}v_i \in E$ for i = 1,...,l and $v_l v_0 \in E$. A circle is called *simple* if the v_i are pairwise distinct, i.e., if v_i =6 v_j for i =6 j.

A *chord* in a simple circle [v $_0$,...,v $_l$,v $_0$] is an edge $_i$v$_j$ ∈ E where i and j differ modulo ` + 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 if and only if χ(G) ≤ 2. A bipartite graph G = (I $_1$,I $_2$,E) is *complete* if for every two vertices x ∈ I $_1$ and y ∈ I, $_2$ the edge xy is contained in E .

The following graphs will come up again and again.

K $_n$: the complete graph with n nodes.

C $_n$: the circle 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 $_n$.

1.2Cut graphs

Let F be a family of nonempty sets. The *intersection graph of* F is the graph with node set F in which two nodes F,F^0 ∈ F are adjoint exactly when F∩F ^{0} =6 ∅.

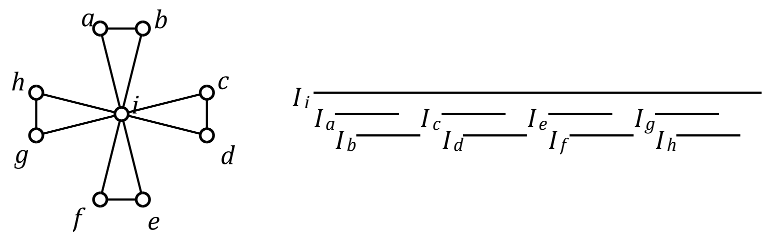


Figure 1.1: Windmill graph and associated interval representation.

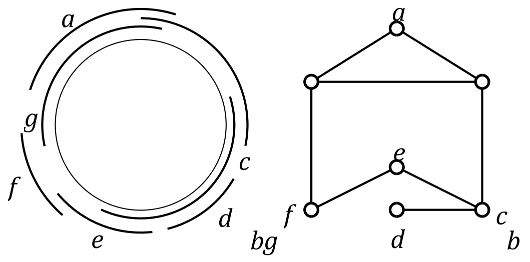


Figure 1.2: A circular arc graph.

If one allows F to contain arbitrary sets, then any graph can be obtained as an intersection graph of a suitable set system (exercise). In the following, we therefore want to restrict the kind of sets that can be contained in F . If one has given such a restriction, two natural questions arise from this, namely the *characterization of* those graphs which have such a cut representation, and the *recognition problem of* deciding for a given graph whether it has a representation with the respective restriction.

In the following, some types of intersection representations are presented as examples. An intersection graph of intervals of a linearly ordered set (say 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 interval contains another proper interval. One can show 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 path and identifying the endpoints of the path with each other so that it forms a circle. The intervals then become arcs. We now allow as sets also arcs which contain or pass over the connecting point and thus obtain the set of circular-arc graph*s*; see Figure 1.2. Analogous to the situation with interval graphs, a *true circular-arc graph* is the intersection graph of a family of arcs such that none of the arcs contains another true one.

A completely different generalization of interval graphs is obtained by first considering that interval graphs are precisely intersection graphs of subpaths of a path-

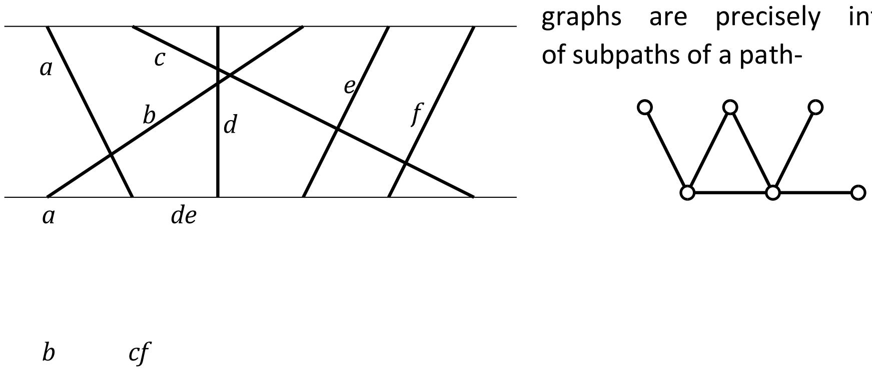


Figure 1.3: A permutation graph.

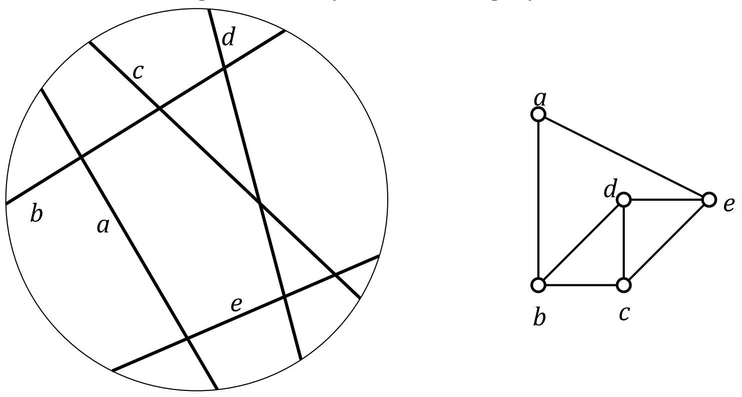


Figure 1.4: A circular chord graph.

of the. Starting from this, one can generalize the representation by looking at path-in-tree or even tree-in-tree graphs.

Again, a completely different kind of graph is obtained by considering two parallel straight lines and choosing n points on each of the lines and n routes, each connecting nodes on different straight 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 nodes arbitrarily distributed on a circle and a set of n lines, each forming pairs of nodes, then these lines are chords of the circle and we obtain so-called circle *graphs*; see Figure 1.4.

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

1.3A brief look at interval graphs

An undirected graph is an *interval graph* if there is a bijective mapping of its nodes to a set I of intervals in R such that two nodes are exactly

are adjacent if the associated intervals have a non-empty intersection. We then call 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 naturally as a graph problem. For this purpose, we consider the graph G = (V,E), which contains one node per lecture and in which two lectures are connected exactly if they overlap in time at some point. In order to obtain a valid space assignment, it now suffices to find a true coloring of this graph, where the colors then correspond to the individual spaces.

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 true 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 will turn out, the coloring problem can be solved efficiently on interval graphs.

Basic Properties. 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 the 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 $\{I_v\}$ be $_{v\in V}$ 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 cut representation are intervals. In fact, all cut graphs are hereditary families.

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

Proposition 1.6. *interval graphs are chordal.*

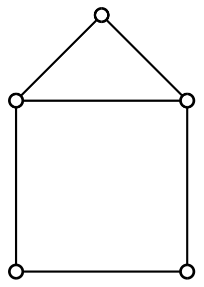


Figure 1.5: A non-chordal graph.

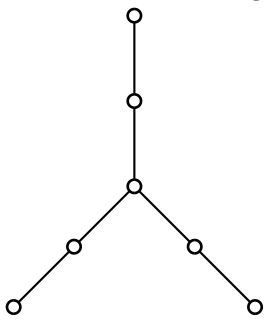


Figure 1.6: A chordal graph that is not an interval graph.

Proof. Let G be an interval graph with chordless circle $[v_0, v_1, \ldots, v_{l-1}, v_0]$ with $l > 3$. Denote I_k the interval corresponding to the node 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 ascending or strictly descending sequence. It follows that I_0 and I_{l-1} do not intersect; a contradiction to the existence of the edge $v_0 v_{l-1}$. □

This gives us a first simple criterion to rule out for certain graphs that they are 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:

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

An undirected graph that is transitively orientable is also called a *comparability graph*. Figure 1.7 shows two examples.

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

1.3. A BRIEF OVERVIEW OF INTERVAL GRAPHS

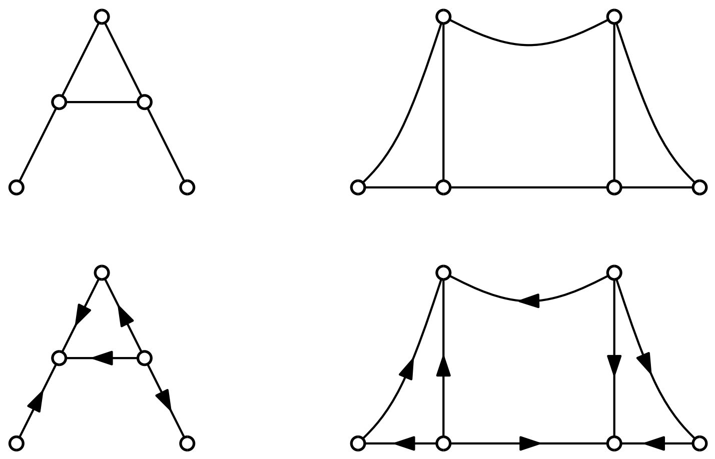


Figure 1.7: Comparability graphs with transitive orientations.

Proof. Let $\{I_v\}_{v\in V}$ be an interval representation of $G = (V,E)$. We define an ori-

entation of the complement $\overline{G} = (V,\overline{E})$ as follows:

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

Here $I_x < I_y$ implies that the interval I_x is completely 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 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 in- if and only if.*

terval graph 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 holds $\chi(G_A) = \omega(G_A)$.

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

Chapter 2

Perfect graphs

Let's look again at the following graph parameters:

$\omega(G)$, the *clique number of* G: the size of a largest complete subgraph of G.

$\chi(G)$, the *chromatic number* of G: the minimum number of colors with which G can be genuinely colored. (equivalent: the minimum number of independent sets needed to cover all nodes of G).

$\alpha(G)$, the *independence number* of G: the maximum number of nodes in an independent set in G.

$k(G)$, the *clique cover number* of G: the minimum number of complete subgraphs needed to cover all nodes of G.

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

$$\omega(G) \le \chi(G)$$

and $\alpha(G) \le k(G)$.

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 deal with graphs that satisfy the following properties

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

and

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

Such graphs are called *perfect*. According to the above duality it is clear that a graph is exactly

then property (P1) is satisfied if its complement graph $\overline{G}$ satisfies property (P2). In the following we want to prove a much stronger statement, namely that the statements (P1) and (P2) are equivalent.

2.1The Perfect Graph Theorem

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

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

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 complementation. 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 the graph G would nevertheless satisfy (P1) and (P2), respectively, then G could be covered with a few small independent sets and with a few small cliques, respectively, and would thus 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 basic 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 ^{0} adding a new vertex v that is connected to all neighbors of v.

Lemma 2.2 Let G = (V,E) *be a graph. For* x =6 y ∈ V *holds* $(G \circ x) - y = (G - y) \circ x$.

Evidence. Practice. □

Definition 2.3 In general, let $x_1,\dots,x_n$ be the nodes of G and let $h = (h_1,h_2,\dots,h_n)$ be a vector with $h_i \in$ N0. The graph $H = G \circ h$ is constructed by $^{h}{}_{i}$ replacing eachi node x_i by an independent set of h_i nodes x^1_i ,...,x and $^s{}_i{}^t{}_j$ connecting x to x exactly when x_i and x_j are adjacent in G . We say that H is obtained by *node 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 with a suitable 0/1-vector.

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

$$G \circ h = (G - x_i) \circ h^0.$$

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

$$G \circ h = (G \circ x_i) \circ h^0.$$

Evidence. Practice. □

2.1. THE PERFECT GRAPH THEOREM

Lemma 2.6. *Let* H *be a graph obtained from* G *by node multiplication. 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 by induction on the number of nodes. Obviously, the statement holds if G has only one node. We now consider a graph G and assume that statements (i) and (ii) hold for all graphs with less nodes 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 node 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.

Thus, we can assume that for any coordinate $h_i \geq 1$. Since any multiplication can be divided into individual steps according to Lemma 2.5, it suffices to show the result for $H = G \circ x$. Let x be ^{0}the node added by multiplication by x. Since any real induced subgraph of $G \circ x$ can be obtained from a real induced subgraph of G by node multiplication, 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^0 are not adjacent, $\omega(G \circ x) = \omega(G)$ holds. Consider a coloring of G with $\omega(G)$ colors. Color x^0 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). Show that $\alpha(G \circ x) = k(G \circ x)$. Let $\mathbb{K}$ be a clique cover of G with $|K| = k(G) = \alpha(G)$ and let $K_x \in \mathbb{K}$ 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 $S \cup \{x^0\}$ is an independent set of $G \circ x$, i.e. $\alpha(G \circ x) = \alpha(G)+1$.

On the other hand, $\mathbb{K} \cup \{\{x^0\}\}$ is a clique cover of $G \circ x$. Thus it holds

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

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 every cardinality-maximal independent set S in G and every clique $K \in \mathbb{K}$, $|S \cap K| = 1$ (exercise!) holds. In particular, this holds for K_x . But since x /∈ S, it even holds that every cardinality-maximal independent set S has exactly one element with the set

$D = K_x \setminus \{x\}$ has in common. So $\alpha(G_{V-D}) = \alpha(G) - 1$. This implies that

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

Now, taking a clique cover of G_{V-D} of size $\alpha(G \circ x) - 1$ together with the additional clique $D \cup \{x^0\}$, we obtain a clique cover of $G \circ x$. Thus, $k(G \circ x) = \alpha(G \circ x)$ holds. □

Lemma 2.7. *Let* G *be an undirected graph for which every real induced subgraph satisfies property (P2) and let* H *be a graph obtained from* G *by node multiplication. If* G *satisfies property (P3), then so does* H*.*

Proof. We give a proof by contradiction. To this end, let H be a graph with minimal number of nodes obtained by node multiplication from G, but not satisfying property (P3). Then holds

$$\omega(H)\alpha(H) < |X|, \qquad (2.1)$$

where X denotes the set of nodes of H, but property (P3) holds for any real subgraph of H. As in the previous lemma, we can assume that each node has been multiplied by at least 1 and that a node u has been multiplied by $h \geq 2$. Let $U = \{u^1,...,u^h\}$ be the nodes of H corresponding to u. Because of the minimality of H satisfies

H_{X-u^1} property (P3), i.e.

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

So equality applies everywhere and we define

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

Then $pq = |X|-1$. Since H_{X-U} can be obtained from G-u by node multiplication, H_{X-U} satisfies property (P2) by Lemma 2.6. Thus, the graph H_{X-U} can be covered by q complete subgraphs $K_1,...,K_q$ of H. Without restriction, we can assume that the K_i are pairwise disjoint and non-increasing according to their

size are sorted. It applies

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

$i=1$

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

$$|K_1| = |K_2| = \dots = |K| = _{q-h+1}p.$$

Let H be ^{0} the subgraph of H induced by $X^0 = K_1 \cup \dots \cup Kq\text{-}h\text{+}1 \cup \{u^1\}$. Then holds.

$$|X^0| = p(q - h + 1) + 1 < pq + 1 = |X|.$$

Due to the minimality of H it follows that

$$\omega(H0)\alpha(H0) \geq |X0|.$$

Moreover, $p = \omega(H) \geq \omega(H^0)$, i.e.

$$\alpha(H^0) \geq |X^0|/p > q - h + 1.$$

Now let S be ^{0} an independent set of size q-h+2 in H^0. Let $u^1 \in S^0$, otherwise S would contain ^{0} two nodes of a clique. But then $S = S^0 \cup U$ is an independent set with q + 1 nodes in H. This is a contradiction to the definition of q. □

With all the important preliminaries done, 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:*

$\omega(G_A) = \chi(G_A)$ *for all* $A \subseteq V$ (P1)

$\alpha(G_A) = k(G_A)$ *for all* $A \subseteq V$ (P2)

$\omega(G_A)\text{-}\alpha(G_A) \geq |A|$ *for all* $A \subseteq V$ (P3)

Proof. We can assume for the proof 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 it is at most

$\Rightarrow \alpha(G_A)$ nodes of each color, it follows $\omega(G)_A\alpha(G_A) \geq |A|$.

(P3) (P1): Assume that $G = (V,E)$ satisfies property (P3). By induction, that satisfies $\omega(G) = \chi(G)$.

$\Rightarrow$any real subgraph of G properties (P1)-(P3). So it suffices to show,

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 an 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 an $\omega(G)$-clique K(S). Let 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 be $_i$ the number of cliques K(S) containing x. Let $_i$ $H = (X,F)$ be the graph obtained by multiplication from G by multiplying x $_i$ by h. It $_i$ follows from Lemma 2.7 that.

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

On the other hand

$$|X| = {}_{x_i} \quad h_i \qquad (2.2)$$

$$x \in V$$

$$= X \; |K(S)| = \omega(G)|S|, \qquad (2.3)$$

$$S \in S$$

$$\omega(H) \leq \omega(G), \qquad (2.4)$$

$$\alpha(H) = \max_{T \in \mathcal{S}} X_i \quad h_i \qquad (2.5)$$

$$x \in T$$

"#

$$= \quad {}_T \quad \cap \quad \max X \; |TK \quad (S)| \qquad (2.6)$$

$$\in S$$
$$S \in S \qquad \leq |\mathcal{S}| - 1$$

(2.7)

Overall, it follows that

$$\omega(H)\alpha(H) \leq \omega(G)(|S| - 1) < |X|,$$

a contradiction.

(P2)$\Leftrightarrow$(P3): From the implications already proved it follows.

G fulfilled (P2) G fulfilled (P1) (2.8)

⇔⇔ G fulfilled (P3) ⇔ G fulfilled (P3). (2.9)

□

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

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

2.2 p-critical and partitionable graphs

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. So for this 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, i.e., G is not perfect, but every real induced subgraph of G is perfect.

In particular, for a p-critical graph G

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

for each node x. In the following, we want to 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 in the meantime (namely in 2006), it says that the p-critical graphs are exactly the circles of odd length and their complements. However, the whole proof is about 170 pages long and thus not suitable for lecture. Instead, we will prove at least some structure results for p-critical graphs to make the conjecture plausible.

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

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

and for all nodes x *of* G*,*

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

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 nodes x. Thus

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

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 nodes is called (α,ω)-*partitionable* if $n = \alpha\omega + 1$ and further for every node x of G holds

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

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

Theorem 2.15. *Let* G *be an* (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 converse, 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, i.e.

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

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

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

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

It follows that $\omega(G) \leq \omega$. Altogether this gives $\alpha = \alpha(G)$ 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 α = α(G) and ω = ω(G).

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

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

(i) G *contains a set of* n *maximal cliques* K $_1$,K ,...$_2$,K , $_n$*covering each node of* G *exactly* *ω*(G) *times;*

(ii) G *contains a set of* n *maximal independent sets* S $_1$,S ,$_2$...,S , $_n$*covering each node of* G *exactly* *α*(G) *times; and (iii)* K $_i$ ∩ S $_j$ = ∅ *exactly if* i = j.

Proof. Choose a maximal clique K of G and for each node x ∈ K choose a minimal clique cover Kx of G - x. By Remark 2.14, all cliques in Kx are cliques of size ω. Now let A be the n×n matrix whose first row is the characteristic vector of the clique K, and whose other rows are the characteristic vectors of all cliques in Kx for each x ∈ K. (Note: these are ω distinct clique covers Kx, each consisting of α cliques. So together with the row for K, this is a total of ωα + 1 = n rows).

Each node y /∈ K is covered by Kx exactly once for each x ∈ K. Each node z ∈ K is covered once by K and exactly once by each Kx with x =6 z. Thus, each node is covered exactly ω times. For each row a of $_i$ A, let K$_i$ denote the corresponding clique with characteristic vector a $_i$. We can write property (i) as 1A =.

ω1, where 1 denotes the row vector where all entries are 1. (Note: it remains to be shown that the K are $_i$ pairwise distinct).

We first proceed with the construction of S$_i$. For this purpose, for each i, choose a node v ∈ K $_i$ and denote by S an optimal coloring (minimal overlap with independent sets!) of G - v. By Remark 2.14, S consists of ω disjoint independent sets of size α. Obviously, each of these sets contains at most one node of K $_i$, and moreover K $_i$ -v has only ω-1 nodes, so there exists an S$_i$

$\in S$ with $K_i \cap S_i = \emptyset$. On the other hand, if j =6 i, then $K_j \cap S_i$ 6= $\emptyset$ (i.e. $|K_j \cap S_i| = 1$), since otherwise 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×n matrix whose

lines that are b_i (i = 1,...,n). According to the above observations, T

$$a_i b_j = \begin{matrix} 0 & i = j \\ 1 & \text{or else.} \end{matrix}$$

We now consider the matrix AB^T. According to the above equation $AB^T = J - I$, 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 follows directly property (i).

Furthermore

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

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 of 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.

$$A\text{-}1 = \omega\text{-}1J - BT .$$

Consider a solution t to the equation tA = c. Then it holds

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

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

In addition

$$t1^T = (1 - cB^T)1^T = n - cB^T 1^T = n - \alpha c1^T \alpha\omega = 1.$$

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 nodes 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 node of G *is contained in exactly* ω *maximal cliques and in exactly* α *maximal independent sets;*

(iv) each maximal clique cuts all but a maximal 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 (i)-(iv) is partitionable.

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

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

Now let x_i be a node of G and let h be ${}^{T}{}_{i}$ the corresponding column in A. Because $A^T B = J\text{-}I$ it follows that $h_i B = 1 - e_i$. That is, h_i selects ω rows of B (which are independent sets!) that together ${}_{i}$ cover G-x . Thus, it is $\chi(G\text{-}x_i) \leq \omega$. 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. 3The Strong Perfect Graph Conjecture

The odd circle C_{2k+1} is not a perfect graph for $k \geq 2$. Obviously, $\alpha(C_{2k+1}) = k$ and $k(C_{2k+1}) = k + 1$ (respectively, $\omega(C_{2k+1}) = 2$ and $\chi(C_{2k+1}) = 3$). On the other hand, every real subgraph of C is ${}_{2k+1}$ perfect. Thus, the odd circles (and hence their complements) are p-critical.

The strong "Perfect Graph Conjecture" states that the above graphs are in fact the only p-critical graphs, i.e., a graph is perfect if and only if it contains none of the above graphs as induced subgraphs. The conjecture can be formulated in the following equivalent ways:

$SPGC_1$ An undirected graph is perfect if and only if it has no induced subgraph which is isomorphic to C_{2k+1} or $\overline{C}_{2k+1}$ with $k \geq 2$.

SPGC$_2$ An undirected graph G is perfect exactly if every odd circle

of length at least 5 in G or $\overline{G}$ has a chord.

SPGC$_3$ The only p-critical graphs are C_{2k+1} and $\overline{C}_{2k+1}$ for $k \geq 2$.

In English, the graphs C_{2k+1} and $\overline{C}_{2k+1}$ are also called *odd hole* and *odd anti-hole*. We have already seen that p-critical graphs have an exceptionally high degree of symmetry. In particular, the previous section states that if G is a p-critical graph and $\alpha = \alpha(G)$ and $\omega = \omega(G)$, then the following statements must be satisfied.

1. $n = \alpha\omega + 1$
2. Each node lies in exactly ω maximal cliques.
3. Each node lies in exactly α maximal independent sets.
4. G has exactly n maximal cliques (of size ω).
5. G has exactly n maximal independent sets (of size α).
6. The maximal cliques and maximal independent sets can be $_n$ numbered $K_1,\ldots,K_n$ and $S_1,\ldots,S$ such that $|K_i \cap S_j| = 1 - \delta_{ij}$, where δ_{ij} denotes the Kronecker delta.

Obviously, every p-critical graph is connected. It is easy to see that C_n is the only connected graph with n nodes for which it holds that $\omega = 2$ and which has exactly n edges (maximal cliques) and for which each node is incident to exactly two edges (each node is contained in exactly two maximal cliques). This allows us to reformulate the conjecture again.

SPGC$_4$ There is no p-critical graph with $\alpha > 2$ and $\omega > 2$.

Following the families C_{2k+1} and $\overline{C}_{2k+1}$, another family of partitionable graphs can be constructed. The graph C^d_n has vertices $v_1,\ldots,v_n$ and v_i and v_j are connected by an edge exactly if i and j differ by at most d. Here, the indices modulo n are considered. It is easy to see that C is $^{\omega}{}_{\alpha\omega}{}^{-}{}_{+}{}^{1}{}_{1}$ an (α,ω)-partitionable graph. For $\omega = 2$, we thus obtain the family

C_{2k+1}, for $\alpha = 2$ the family $\overline{C}_{2k+1}$. However, these graphs are not p-critical for $\alpha > 2$ and $\omega > 2$.

Theorem 2.21 (without proof.). *For* $\alpha \geq 3$ *and* $\omega \geq 3$, *the partitionable graphs are.*

$C^{\omega}{}_{\alpha\omega}{}^{-}{}_{+}{}^{1}{}_{1}$ *not* p-critical.

Thus the conjecture can again be equivalently reformulated.

SPGC$_5$ If G is p-critical with $\alpha(G) = \alpha$ and $\omega(G) = \omega$, then G contains an induced Subgraph that is isomorphic to $C^{\omega-1}_{\alpha\omega+1}$.

In the meantime, the above conjecture could be shown. However, the proof is extremely extensive.

Theorem 2.22 ("Strong perfect graph theorem", 2006). *An undirected graph is exactly then perfect if it does not contain an induced subgraph isomorphic to* C_{2k+1} *or* $\overline{C}_{2k+1}$ *with* $k \geq 2$.

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 provides the foundation for the study of perfect graphs.

Definition 3.1 A graph G is *chordal if* every circle 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.

3.1Characterization

In the following we want to work out a characterization of chordal graphs, which will also be of central importance from an algorithmic point of view.

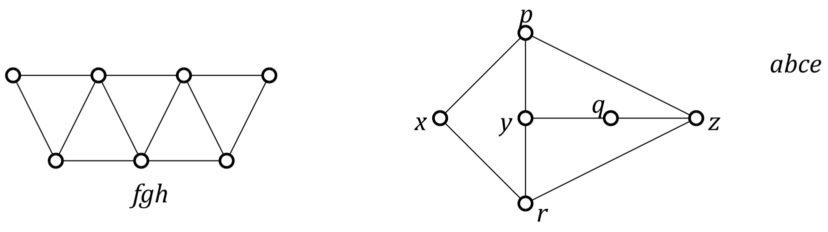


Figure 3.1: Two example graphs, the left one is chordal, the right one is not.

Definition 3.2. A node x of is called *simplicial* if its adjoint nodes Adj(x) form a complete subgraph of G, that is, Adj(x) is a (not necessarily maximal) clique.

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

G

Definition 3.3. Let G = (V,E) be an undirected graph and let σ = [v_1 ,...,v_n] be a node order. The order σ is a *perfect elimination scheme* if every node v_i is a simplicial node of the induced subgraph G $_{\{vi,...,v}$. $_{n\}}$ Equivalently, each of the sets X $_i$ = {v_j ∈ Adj(v $_i$) | j > i} is complete.

Obviously, the right graph in Figure 3.1 does not have a perfect elimination scheme because it does not have a simplicial node. 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.

A node set S ⊆ V is a *(node) separator* for two nonadjoint nodes a and b (also called an a-b separator) if a and b are in different context components of G - S .

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

Proof. Let S be an (inclusion) minimal a-b separator of G with connection components G and G $_B$ of G $_{V−S}$, containing a and b, $_A$ respectively. By virtue of the minimality of S, each node of S has a neighbor in A and a neighbor in B . For each pair of nodes x,y ∈ S, there is a path [x,$_1$a ,...,a_r ,y] and a path [y,b_1 ,...,b_t ,x] with a_i ∈ A and b_i ∈ B such that these paths have minimal length. Then the circle [x,a $_1$,...,a_r ,y,b_1 ,...,b $_t$,x] is simple and has length at least 4; hence it must have a chord. But since S is a separator, a $_i$b ∈/ $_j$ E. Moreover, it follows from the minimality of r and t that a $_i$a ∈/ $_j$ E and b $_i$b ∈/ $_j$ E for i < j - 1. Therefore, the only possible chord is xy ∈ E. □

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

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

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

Proof. If G is complete, the statement holds trivially. So suppose that G contains two nonadjoint nodes a and b and the statement holds for all graphs that have fewer nodes than . Let S be an (inclusion) minimal node separator for a and b, and let G and G be $_B$ the connection components of G $_{V−S}$, containing a and b, $_A$ respectively. By induction, either G contains $_{A+S}$ two nonadjoint simplicial nodes (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 hence every node in A is simplicial in G $_{A+S}$. Since Adj(A) ⊆ A+S, a simplicial node of G $_{A+S}$ in A is also simplicial in G. By the same argument, one shows that B also contains a simplicial node. □

G

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 node separator induces a complete subgraph of G*.*

Proof. (i) ⇒ (iii): is exactly the statement of Lemma 3.4.≥
⇒Each minimal∈ a-b separator contains x and y $_i$ for some i. But then (iii) (i): let [a,x,b,y $_1$,y$_2$,...,y $_k$,a] with k 1 be a simple circle of G = (V,E).
xy$_i$ E a chord of the circle.

⟹ (i)(ii): By Lemma 3.6, a chordal graph G has a simplicial node x. Since the graph GV-{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 circle of G and let x be a node of C with minimal index in a perfect elimination scheme. Since | Adj(x) ∩ C| ≥ 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
nodes of G, there is also a perfect elimination scheme of G starting with v.

3.2Detection of chordal graphs

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 node. If all nodes can be removed from the graph in this way, the deletion order yields a perfect elimination scheme. However, if the algorithm stops before this, we have found an induced subgraph that does not contain any simplicial node. Thus, the graph is not chordal. The main problem of this input : Undirected graph = (V,E).

Output : Node order σ.

```
1 Assign the label ∅ to each node;
2 for          in to 1 tue
    3choose an unnumbered node←v with largest label;
    4σ(i)    v;
```

G

```
5for each unnumbered node←w ∈ Adj(v) add i to Label(w);
6 End
```

Algorithm 1 : LexBFS

The advantage of this approach, which goes back to Fulkerson and Gross, lies in its efficiency. For example, finding a simplicial node 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 leaves us with a choice of at least two simplicial nodes at any point in time. Thus, we are free to choose a node v to $_n$ keep until the end. Similarly, however, we can now choose another node v $_{n-1}$, which is adjacent to $_n$ v , and cancel it for the n - 1th position. If we were to continue in this way, we would be creating 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 node to visit. Here, the queue used to implement breadth-first search contains a set of unordered subsets (nodes whose order is arbitrary are in the same set). Occasionally, these sets are refined, that is, 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 node has a *label*, which consists of a set of numbers listed in descending order. The nodes are then sorted lexicographically in the queue according to their labels.

Figure 3.3 shows an example of the flow of a lexicographic breadth-first search on the graph from Figure 3.1. It can be seen 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 the selection of the ite node (line 3 in the algorithm). The index i is decreased in each run. Thus, the order of nodes with different labels is always preserved. It

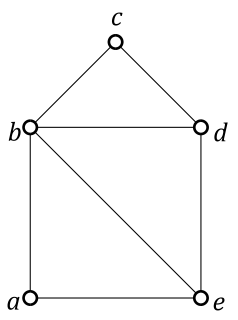


Figure 3.2: Chordal graph

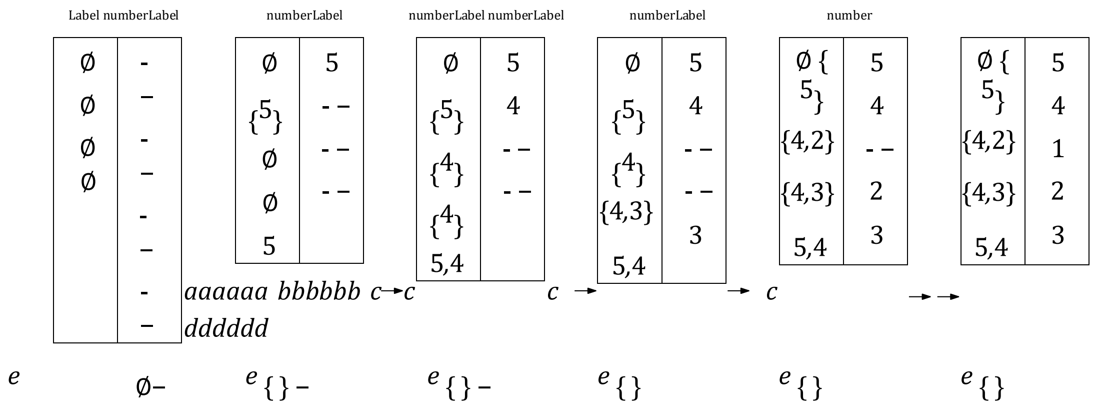


Figure 3.3: Chordal graph

the following properties apply:

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

(L2) $L_i (x) < L_i(y) \Rightarrow L_j(x) < L (_jy)$ $\forall j \leq i$ ∈∈

(L3) If σ^{-1} (a) < σ^{-1}b) < σ^{-1}(c) and c Adj(a) - Adj(b), then there exists a d Adj(b) - Adj(a) with σ (c) < σ ($^{-1-1}$d).

Property (L1) states that labels may only become larger, but not smaller. Property (L2) states that once a label of a node is smaller than the label of another node, it remains so in subsequent iterations. Property (L3) describes a condition under which a suitable node d must exist that was numbered before c. The reasoning here is as follows: If there were no such node, then 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 receiving a larger label than b and, since according to the other properties the order of 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 node order 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 nodes, and consider an order σ obtained by applying Algorithm 1 to a chordal graph G. By induction, it suffices to show that x = σ(1) is a simplicial node of G.

Thus, we assume that x is not simplicial. That is, there exist nodes x $_1$,x$_2$ ∈ Adj(x) with x$_1$ x ∈/$_2$ E. Here we choose these nodes such that x is $_2$ maximal with respect to σ. Now the nodes x = $_0$x ,x $_1$,x $_2$ just satisfy property (L3) such that there exists a node$_3$ x $_1$ adjacent to x 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 $_2$x cannot $_3$ exist either. We now proceed inductively. We have $_m$ constructed a sequence x $_0$,x ,x $_{12}$,x ,...$_3$,x satisfying the following properties.

(1) x0x^i ∈ E ⇔ i ≤ 2, ∈

⇔

(2) For i,j > 0, x $_i$x $_j$ E|i - j| = 2 holds,

(3) $\sigma^{-1}(x_0) < \sigma^{-1}(x\ ) <\ _1$--- $< \sigma^{-1-1}(x_{m2})$,

(4) For j ≥ 3, x is $_j$ the maximal node with respect to σ with x $_{j-2}$x$_j$ ∈ E but x $_{j-3}$x ∈/ $_j$ E.

We have just constructed the sequence for m = 3 and now show how to continue the construction. The nodes x $_{m-2}$,x $_{m-1}$ and x $_m$ satisfy property (L3), so there exists a node x $_{m+1}$ with σ ($^{-1}$x $_{m+1}$) > σ $^{-1}$(x $_m$) that is adjacent to $_{m-1}$ x 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 ,...$_1$,x $_{m+1}$ satisfies the above properties.

First, x is $_{m+1}$ 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 $_{m+1}$ apply property (L3) to x $_{m-3}$,x $_{m-2}$ and x and obtain a node larger than x $_{m+1}$ and $_{m-3}$ adjacent $_{m-2}$ to x but not to x . This would contradict the maximality of x$_m$ in (4). So it follows that x is $_{m+1}$ not $_{m-3}$ adjacent to x. If x$_{m+1}$ were adjacent to$_m$ one of the nodes x $_0$,...,x

$_{m-4}$ or x, then by (1) and (2) we would have a circle without chords. But this would contradict the fact that G is chordal.

So the above inductive procedure continues unrestricted. But since the graph is finite, this leads to a contradiction. The node x is therefore simplicial. It follows directly that a perfect elimination scheme is found for chordal graphs. The converse follows from Theorem 3.7. □

To obtain an efficient recognition algorithm for chordal graphs with linear runtime two things are still needed. 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 node order is a perfect elimination scheme.

3.2.1Implementation of LexBFS

Instead of computing the labels of the nodes, we simply keep the nodes ordered by their labels. To do this, we define for a label l the set S_l of nodes with label l that are not yet numbered. 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 node v, we store a pointer SET(v) pointing to the set that contains v. This pointer points to the set that contains 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 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 node with maximal label. We choose as v in line 3 of Algorithm 1 any node in the last set in Q and remove this node from SET(v). If SET(v) becomes empty as a result of this, we remove it from Q. We now set σ and σ^{-1} for the node v. We then iterate over the neighbors of v and move them into the new resulting sets. Here, a node $\in^{w}$ Adj(v) with label l is to be moved into the set $S_{l\cdot\sigma^{-1}(v)}$ that follows directly after 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 nodes is easily done in constant time. We then 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 flow of the procedure as pseudo-code. It is not hard to see that the whole update takes only O(| 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 perform a lexicographic breadth-first search on an undirected graph* G = (V,E) *with* O(|V| + |E|) *time and*
Space requirement calculated.

3.2.2Detection of perfect elimination schemes

We can now use lexicographic breadth-first search to determine a possible node order for a graph that is a perfect elimination scheme exactly when the graph is chordal. Thus, we must finally check whether the computed node order is a perfect elimination scheme. To prevent this step from becoming a bottleneck in chordal graph detection, we need an algorithm that decides, for a graph with a given node order, whether

```
for all unnumbered nodes w ∈ Adj(v)
tue
    if FLAG(SET(w)) = 0 then
        Create new set S and insert it directly after SET(w) in Q;
        FLAG(SET(w)) ← 1; FLAG(S') ← 0; Pointer to SET(w) in FIXLIST;
    Ende
    Let S' be the set immediately posterior to SET(w) in Q;
    Remove w from SET(w); Add w to S';
    SET(w) ← S'
End
for any amount of S in FIXLIST do
    if S is empty then
        FLAG(S) ← 0;
        Remove S from Q;
    End
    Remove S from FIXLIST;
End
```

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

it is a perfect elimination scheme. Of course, one can be naive about this and always check that for each node the 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 nodes in ascending order of numbering, check some neighborhoods, and then remove them. To check whether the neighborhood of the first node x forms a clique, we assign the neighbor y of x with the smallest number the task of checking whether it is connected to all of x's neighbors. This is not done until node y is processed. At this point, node y may have been given the task of checking neighborhoods by other nodes.

Let us now consider the time of processing y. If y is not adjoint to all neighbors of x, then the subsequent neighbors of x do not form a clique and the node order 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 needed to treat x.

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

Arrays are used to evaluate σ and σ^{-1} in constant time, for Adj(v) 1 Boolean procedure PERFECT(σ).

```
2 Start 3        for all nodes v do A(v)   ∅; 4      for i ← 1 to ∈ n - 1 tue ←
5          vσ   (i);
6          X←← {xAdj∅ (v) | σ ⁻¹( ∈v) < σ ⁻¹(x)};
7          If X =then go to line 10;
8       End uσ   (min{σ⁻¹ (x) | xX}        );
9          concatenate←X      - {u6 } to∅ A(u); 10        if A(v) - Adj(v) =then
11return false;
12           End
13
14       return true; 15 end
```

Algorithm 3 : Procedure to check if a given sequence of nodes σ is a perfect elimination scheme.

and A(v), doubly concatenated lists are used. The jump in line 7 is performed exactly j-1 times, where j denotes the number of context components. The set A(u) may contain duplicates. Algorithm 4 shows how the test in line 10 can be performed in O(| 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 for w ∈ A(v) tue← 3 if TEST(w) = 0 then
4return not empty;
5     End 6 End
7 for w ∈ Adj(v) do TEST(w)          0;

8 return empty;                  ←
```

Algorithm 4 : Adjacency test in line 10 of Algorithm 3.

Thus, the entire algorithm runs with runtime and memory proportional to

$$|V| + X \mid Adj(v)| + X \mid A(u)| .$$

v∈Vu∈V

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 hence 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 order σ is a perfect elimination scheme. The algorithm can be implemented with runtime and memory requirements proportional to* |V| + |E|.

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

The algorithm returns the statement "false" in the σ^{-1}(u)-th iteration exactly when there are three nodes u,v,w (σ^{-1-1} (v) < σ (u) < σ ($^{-1}$w)) where u is defined in line 8 by the σ^{-1}(v)-th iteration, and u,w ∈ Adj(v) but u is not adjacent to w.

Obviously, in the case of the answer "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 node with maximum index σ^{-1}(v) such that X = {w | w ∈ Adj(v) and σ^{-1}(v) < σ^{-1}(w)} is not complete; that is, the last node in the order that is not simplicial at the time of its removal. Let u be the node of X defined by the σ^{-1}(v)-th iteration in line 8. Then (in line 9) X - {u} is added to A(u). Since the algorithm does not terminate, each node x ∈ X - {u} is adjacent to u. Furthermore, due to the maximality of σ^{-1}(v), the node u is simplicial, so in particular every two nodes from X - {u} are adjoint. So X is complete, which contradicts the assumption. □

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

3.3Chordal 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 $J \subseteq I$ holds $T_i \cap T_j =6\ \emptyset$ for all $i,j \in J$ implies that $\cap_{j\in J} T_j =6\ \emptyset$.

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

Proof. We assume $T_i \cap T_j =6\ \emptyset$ for all $i,j \in J$. We first consider the case where there are three vertices a,b,c of T such that for each 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 $_{1,2}$P ,P be $_3$ the simple paths connecting a with b, b with

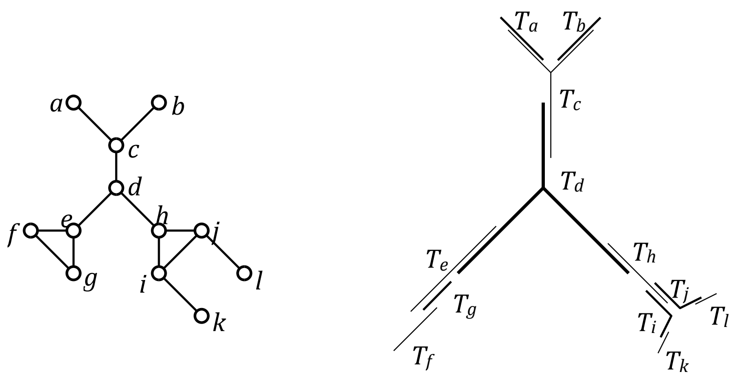


Figure 3.4: Chordal graph and a representation as a cut graph of subtrees of a tree.

c and connect c to a. Since T is a tree, it follows that $P_1 \cap P_{2\,3}\, 6= \emptyset$. However, each $T_s (s \in S)$ contains at least one of the paths P_i. Thus

$$\underset{s\in S}{\setminus}\; T_s \supseteq P_1 \cap P_2 \cap P_3 = 6\emptyset.$$

We now prove the proposition by induction. We assume that the proposition $T_i \cap T$

$$_j = 6\ \emptyset \text{ for all } i,j \in J \setminus T_j = 6\ \emptyset \Rightarrow {}_{j\in J}$$

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

Consider a family of subtrees $\{T_{i1}, \ldots, T_{ik+1}\}$. According to the induction hypothesis there exist nodes a,b,c of T such that

$$_{kk+1}\, a \in \setminus T_{ij},\ b \in \setminus T_{ij},\ c \in T_{i1} \cap T_{ik+1}.$$
$$_{j=1 j=2}$$

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

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

(i) G *is chordal.*

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

(iii) *There exists a tree* T = (K,E) *whose set of nodes* K are *the maximal cliques of* G *such that each of the induced subgraphs* T_{Kv} *(*v ∈ V*) is contiguous (i.e., a subtree), where* Kv *is the set of cliques in* K that *contain* v .

Proof. (iii) (ii) Suppose there exists a tree T = (K,E) satisfying the properties in (⇒iii). Let v,w ∈ V. The nodes∈ K v and w are adjacent in G exactly if there exists a maximal clique A containing both nodes. This is again the case exactly if Kv ∩K $_w$ =6 ∅, so exactly if T $_{Kv}$ ∩T $_{Kw}$ =6 ∅. So G is intersection graph of the family of subtrees {T $_{Kv}$ | v ∈ V}.

(ii) (i) Let {T_v } be $_{v∈V}$ a family of subtrees of a tree T such that vw ∈ E exactly if⇒ T $_v$ ∩ T $_w$ =6 ∅.

We assume that G contains a circle [v $_{0,1}$v ,...,v $_{k-1}$,v $_0$] with k > 3 that has no chord. Let T denote $_i$ the $_i$ tree corresponding to v. Then T $_i$ ∩ T $_j$ =6 ∅ 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 $_i$ ∩ T_{i+1} for i = 0,...,k - 1. Let b be $_i$ 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 and in T , $_i$ respectively $_{i+1}$; hence b_i lies in T $_i$ ∩T_{i+1} . Let P be $_{i+1}$ the simple path connecting b_i to b_{i+1} . Obviously, P $_i$ ⊆ T_i holds. So P $_i$ ∩P $_j$ = ∅ holds if i and j differ modulo k by more than 1. Moreover, P $_i$ ∩ P_{i+1} = {b_i } holds for i = 0,...,k - 1. Consequently, S_i P_i is a simple circle in T. This contradicts the definition of a tree. the theorem holds for all graphs that have fewer nodes than⇒G . (i) (iii) The proof is by induction on the size of G. We assume that.

If G is complete, T is a single node and the statement is trivial. If G is non-contiguous with connection components G $_1$,...,G_k , then by induction assumption for each of the graphs G there exists $_i$ a corresponding tree T $_i$satisfying the statements in (iii). We connect any node of T $_i$ to any node of T to $_{i+1}$ obtain the asserted tree satisfying property (iii) for G .

In the following, we consider the case where G is neither complete nor disjoint. Let a be a simplicial node of G and let A = {a} ∪Adj(a). Clearly, A is a maximal clique of G.

Let U = {u ∈ A | Adj(u) ⊂ A} and Y = A - U. Let a ∈ U; see Figure 3.5. Since G is not complete, there are nodes in V - A, and moreover since G is contiguous, and nodes in U are not adjacent to nodes 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 ^{0} = G $_{V-U}$, which is chordal and has fewer nodes than G. By induction assumption, let T ^{0} be a tree whose node set K is ^{0} the set of maximal cliques of G ^{0} such that for each node in V-U the node set Kv ^{0} = {X ∈ K ^{0} | v ∈ X} ^{0} induces a connected subgraph (subtree!) of T .

Remark. If Y is a maximal clique of G ^{0}, then K = K ^{0}+{A}-{Y}, otherwise K = K ^{0} + {A}.

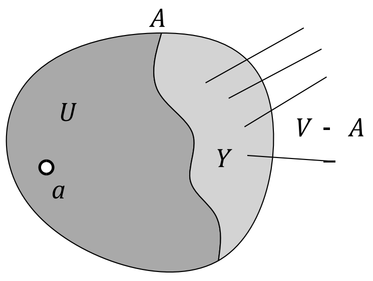


Figure 3.5: Illustration of clique A, its two subsets U and Y and its complement V - A

Let B be a maximal clique of G^0 containing Y . We distinguish two cases, depending on whether B = Y or not.

Case 1. If B = Y, we obtain T from T^0 , by renaming the node B to A .

Case 2. If B =6 Y , we obtain T from T^0 , by attaching the new node A as a leaf to the node B .

In both cases K_u = {A} for all u ∈ U and Kv = Kv ^{0} for all v ∈ V - A. This

Sets, by premise, each induce a subtree of T. Thus, we only need to check the sets K_y with y ∈ Y . In case 1 $\mathcal{K}_y = \mathcal{K}'_y + \{A\} - \{B\}$, which induces the same subtree as $\mathcal{K}'_y$, since we have merely renamed a node. In case 2 is$\mathcal{K}_y = \mathcal{K}'_y + \{A\}$, which obviously induces a subtree. Thus T is the tree we are looking for and the theorem is proved. □

3.4Perfectness

We will now show that chordal graphs are perfect. For this we use the fact that node 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_{A1} ,...,G_{At} *be the context components of* G $_{V–S}$. *If* S *is a clique, then*

$$\chi(G) = \max\chi(G_{S+Ai})\ _i$$

and

$$\omega(G) = \max\omega(G_{S+Ai})\ .\ _i$$

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

Similarly, surely $\omega(G) \geq \omega(G_{S+Ai})$ holds for all i. On the other hand, no clique X can be in G contain nodes from A_i and A_j with j =6 i, since S is a separator. Therefore, each clique X must be completely contained in one of the subgraphs G, $_{S+Ai}$ so equality follows. □

Corollary 3.16. *Let* S *be a separator of a connected graph* G = (V,E) *and let* G $_{A1}$,...,G $_{At}$ *be the coherence components of* G $_{V–S}$. *If* S *is a clique and every subgraph* G *is* $_{S+Ai}$ *perfect, then* G is *perfect.*

Proof. We assume that the statement holds for all graphs that have fewer nodes than G . Thus $\omega(G^0) = \chi(G^0)$ holds for all real induced subgraphs G^0 of G and it suffices to show that $\omega(G) = \chi(G)$.

For each of the subgraphs, $\chi(G_{S+Ai}) = \omega(G_{S+Ai})$ holds due to perfection, so in particular $\max_i \chi(G_{S+Ai}) = \max\ \omega(G_{i\ S+Ai})$. 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 nodes and assume that the statement holds for all graphs that have fewer nodes than G. We can assume without restriction that G is connected but not a clique. Then there is a separator S in G that $_{At}$ decomposes G into coherence components G $_{A1}$,...,G with t ≥ 2. Each of the graphs G $_{S+A1}$,...,G $_{S+At}$ is chordal and hence perfect by induction assumption. By Lemma 3.4, S forms a clique and hence, by Corollary 3.16, G is also perfect. □

3.5Algorithms 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 let σ be a perfect elimination scheme for G. The first observation, going back to Fulkersson and Gross, is that every maximal clique is of the form $\{v\} \cup X_v$, where

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

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

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

It is easy to modify Algorithm 3 to output the set $\{v\} \cup X_v$ at each step. 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 when all nodes in X_v were added to A(v) at once in a previous step. (Proof: Exercise!)

Since σ is a perfect elimination scheme, at each step the set of nodes added to A(v) is a subset of X_v. Thus, it suffices 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 whether $\{v\} \cup X_v$ is a maximal clique and therefore must be output when processing the node v . 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.

```
1 Procedure CLIQUES(σ) 2
Begin
3
4         for all nodes v do S(v)      0
```

```
5      for i ← 1 to ∈ n tue          ←
6          vσ   (i);
       χ ← 1;
7          X←← {xAdj∅ (v) ∅| $\sigma^{-1}$(v) < σ ($^{-1}$x)};
8          If Adj(v) =then print {v};
9          If X =then go to line 16;
10         uσ   (min{$\sigma^{-1}$ (x) | x ∈ X}); 11  S(←u) ← maxU{S(u), |X| - 1};
12             if S(v) < |X| then
13print {v}X;
14χ     End          max{χ,1 + |X|};
15         End←
16
17        print "The chromatic number is "χ;
18 End
```

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

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 α(G). Since G is perfect, α(G) = k(G), and we also want to specify an independent set and a clique cover of this quantity.

To this end, we inductively define a sequence of nodes y_1,y ,...$_2$,y_t by $y_1 = \sigma(1)$ and y_i = σ(min{σ^{-1} $^{-1}$(v) | σ (v) > σ^{-1}(y_i),v /∈ X_{y1} ∪ X_{y2} ∪ --- ∪ X_{yi-1}}) as the first successor of y in the $_i$

node order σ that does not exist in any of the sets X_{yj} with j < i

is included. In particular

$$V = \{y_1, \ldots, y_t\} \cup X_{y1} \cup \cdots \cup X_{yt}.$$

The following sentence applies.

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,...,t*) is a minimal clique cover of* G.

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 hence $\{Y_1, \ldots, Y_t\}$ 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 overlap. It is not hard to implement the algorithm with linear running time. □

Chapter 4

Comparability graphs

In this section we will look in more detail at comparability graphs, graphs that have a transitive orientation. We will see that these graphs are also perfect and give an algorithm to detect them.

4.1 Γ-chains and implication classes

An undirected graph G = (V,E) is a *comparability graph* if an orientation (V,F) of G exists with

$$F \cap F^{-1} = \emptyset, \qquad F + F^{-1} = E, \qquad F^2 \subseteq F,$$

where $F^2 = \{ac \mid ab, bc \in F \text{ for a node } b\}$. The relation F is a strike partial order of V whose comparability relation is E, and F is called the *transitive orientation of* G (or of E).

For example, consider a circle with nodes a,b,c,d (in that order along the circle); see Figure 4.1. The arbitrary choice ab ∈ F *forces* the orientation of the other edge incident to b towards b, i.e., cb ∈ F. Analogously, it also forces the orientation of the other edge incident to a away from a, i.e., ad ∈ F. Finally, cb ∈ F must also hold. Let us now make this notion of "forcing" more precise.

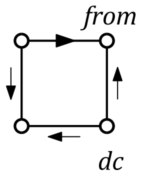


Figure 4.1: Example of forced edge directions.

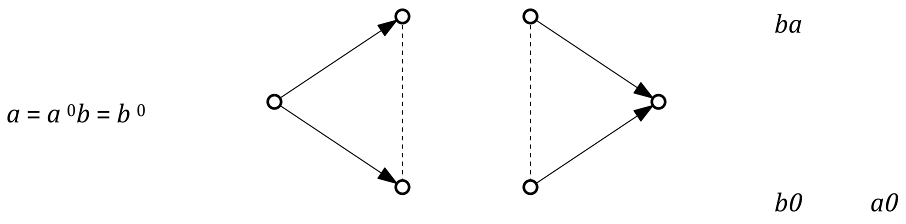


Figure 4.2: Illustration of the relation Γ.

We define a binary relation Γ on the edges of an undirected graph G = (V,E) as follows, see Figure 4.2:

$$ab\,\Gamma\,a^0b^0 \quad \text{if and only if} \quad \begin{cases} \text{either} & a = a^0 \text{ and } bb^0 \notin E \\ \text{or} & b = b^0 \text{ and } aa^0 \notin E \end{cases}$$

If $ab\,\Gamma\,a^0b^0$ holds, we say that ab^0 *directly enforces* the edge a^0b. Since E is irreflexive, ab Γ ab holds, but ab6 Γ ba. Clearly, the reflexive transitive closure Γ^* of Γ is an equivalence relation on R and thus partitions E into equivalence classes, which we call *implication classes of* G. Two edges ab and cd are exactly in the same

Implication class, if a sequence of edges exists

$$ab = a_0 b_0\ \Gamma\ a_1 b\ \Gamma_1\ a_k b_k = cd \qquad \text{with } k \geq 0.$$

Such a sequence is called a Γ-chain from ab to cd. We say that ab *forces* the edge cd if $ab\,\Gamma^*\,cd$.

The following properties apply (exercise!).

$$ab\,\Gamma\,a^0b^0 \Leftrightarrow ba\,\Gamma\,b^0a^0\ ab\,\Gamma^*\,a^0b^0$$

$$\Leftrightarrow ba\,\Gamma^*\,b^0a^0$$

Let I(G) be the set of implication classes of G. We define

$$\hat{I}(G) = \{\hat{A} \mid A \in I(G)\},$$

where $\hat{A} = A \cup A^{-1}$ denotes the symmetric closure of A. The elements of $\hat{I}(G)$ are called *color classes*.

Theorem 4.1 *Let* A *be an implication class of an undirected graph* G*. If* G *has a transitive orientation* F, then *either* $F \cap \hat{A} = A$ *or* $F \cap \hat{A} = A^{-1}$*. In both cases,* $A \cap A^{-1} = \emptyset$ *holds.*

Proof. The relation Γ is just defined such that for any transitive orientation F of G holds: if ab Γ a^0b^0 and ab ∈ F then a^0b^0 ∈ F.

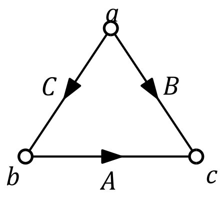


Figure 4.3: Precondition of the triangle lemma.

Repeated application of this property yields F ∩ A = ∅ or A ⊆ F. On the other hand, (i) $A \subseteq F + F^{-1}$ and (ii) $F \cap F^{-1} = \emptyset$ hold.

Now if F ∩ A = ∅, it follows that

$$F \cap A = \emptyset \Rightarrow A \subseteq F^{-1} \text{ [because of (i)]} \Rightarrow A^{-1} \subseteq F \Rightarrow F \cap A\hat{} = A^{-1}.$$

If, on the other hand, A ⊆ F, it follows that

$$A \subseteq F \Rightarrow A^{-1} \cap \subseteq F^{-1} \qquad \emptyset \Rightarrow F \cap A^{-1} = \emptyset \text{ [because of (ii)]} \Rightarrow F \cap A\hat{} = A.$$

In both cases $AA^{-1} =$. □

We will see later that the converse of Theorem 4.1 also holds; namely, if $A \cap A^{-1} = \emptyset$ for any implication class of A, then G has a transitive orientation.

Remark. One might now suppose that any union of implication classes $F = S_i A_i$ with $F \cap F^{-1} = \emptyset$ and $F + F^{-1} = \emptyset$ always forms a transitive ordering of G. However, this is not the case. For example, if we consider a triangle, we find that it has exactly six implication classes, each containing a single edge in one direction. To get an orientation of the above construction there are exactly $2^3 = 8$ possibilities. But two of them are not transitive.

We will use the next two lemmas repeatedly throughout the chapter. Let $ab = a_0b_0\ \Gamma\ a_1b\ \Gamma_1$ --- $\Gamma\ a_kb_k = cd$ be a Γ-chain. For i = 1,...,k holds.

$$a_{i-1}b_{i-1}\ \Gamma\ a_ib_{i-1}\ \Gamma\ a_ib_i,$$

since the added middle edge coincides with the first or the second edge. We obtain the following statement.

Lemma 4.2. *If* $ab\ \Gamma^*\ cd$ *holds, then there exists a Γ-chain from* ab *to* cd of *the form*

$$ab = a_0b_0\ \ \Gamma\ a_1b\ \Gamma_{0\,1}\ \ a_1b\ \Gamma\ \text{---}\ \Gamma_1\ \ a_{k2}b_k = cd.$$

Such a chain is called a *canonical Γ-chain*. The use of canonical chains occasionally simplifies proofs.
Lemma 4.3 (Triangular Lemma). Let A,B,C *be implication classes of an undirected Graph* $G = (V,E)$ *with* $A =6\ B$ *and* $A =6\ \ C^{-1}$*, so that* $ab \in C, ac \in B$ *and* $bc \in A$

(Figure 4.3).

(i) *If* $b^0c^0 \in A$*, then* $ab^0 \in C$ *and* $ac^0 \in B$.

(ii) *If* $b^0c^0 \in A$ *and* $a^0b^0 \in C$*, then* $a^0c^0 \in B$.

(iii) *No edge from* A is *incident to the node* a.

Proof. By Lemma 4.2 there exists a canonical Γ-chain

$$bc = b_0c_0\ \ \Gamma\ b_1c\ \Gamma_0\ \ b_1c\ \Gamma_1\ \dots\ \Gamma\ b_kc_k = b^0c^0.$$

We now inductively show the existence of $ab_i \in C$ and $ac_i \in B$ for $0 \le i \le k$. For $i = 0$, this is obviously satisfied. We now consider the case $i \ge 1$.

Since $ac_{i-1} \in B$, $b_i\,c_{i-1} \in A$, but $A =6\ B$, $ab_i \in E$. Since $b_i\,c\ \Gamma_{i-1}\ \ b_{i-1}c$, it $_{i-1}$ follows that $b_ib\ \in\!/_{i-1}\ E$. It follows that ab_{i-1} and ab_i are in the same implication class, namely C.
Since $b_i\,c_i \in A$ and $A =6\ C^{-1}$, it follows $b_i\,c_i6\ \Gamma\ b_ia$, hence $ac_i \in E$. Since $b_i\,c_i\ \ \Gamma\ b_i\,c_{i-1}$, c_i and c are $_{i-1}$ not adjacent. Consequently, $ac\ \Gamma_i\ \ ac_{i-1}$, so $ac_{i-1} \in B$. This proves property (i).

Furthermore, property (iii) follows directly from property (i). If there is an edge $b^0c^0 \in A$, then the existence of the edges ab^0 and ac^0 follows from (i). But if now $b^0 = a$ or $c^0 = a$ holds, this provides a contradiction to the irreflexivity of E.
For (ii), suppose first that $B =6\ C$. Consider the canonical Γ-chain $ab = a_0\,b\ \Gamma_{0\,0}\,a_{11}b\ \Gamma_1\ \text{---}\ \Gamma_1\,a_2b_k = a^0{}_kb^0$ in C. We show by induction that for $i = 0,\dots,k$ the triangle $a_i\,b_ic$ exists and is isomorphic to the triangle

abc (i.e., the edges are in the same implication classes). Obviously, this holds for the triangle a $_0$b $_0$c = abc. Now let i ≥ 1 and suppose that triangle a $_{i-1}$b $_{i-1}$c exists and is isomorphic to abc. Then b $_{i-1}$a$_i$ ∈ C $^{-1}$ and b $_{i-1}$c ∈ A. Since A =6 C $^{-1}$, it follows that a $_i$c ∈ E. Moreover, a $_i$c Γ a $_{i-1}$c ∈ B. Now consider the edge pair a $_i$b$_i$ ∈ C and a $_i$c ∈ B. By premise, B =6 C. This provides the existence of the edge b $_i$c ∈ E. It holds b $_i$c Γ b $_{i-1}$c ∈ A. So, in particular, the last triangle of the sequence, a ^{0}b ^{0}c, exists. Statement (i) then directly yields the existence of the edge a ^{0}c^0 ∈ B.

Let us now consider the remaining case B = C. Considering the pair of edges b^0 a^0 ∈ B $^{-1}$ and b^0 c^0 ∈ A $^{-1}$ directly yields the existence of the edge a ^{0}c^0 ∈ E because of A =6 C^{-1} = B. We now assume that a^0 c^0 ∈ D with C =6 D. By part (i), on the one hand, ac^0 ∈ B = C holds. On the other hand, we can also apply part (i) to the triangle a ^{0}b ^{0}c ^{0} with inverted edges as follows; namely, to the edges c ^{0}b^0 ∈ A $^{-1}$, b ^{0}a ∈^{0} C $^{-1}$ and c ^{0}a^0 ∈ D $^{-1}$. This works because B $^{-1}$ = C $^{-1}$ =6 D $^{-1}$ and (A $^{-1}$) $^{-1}$ = A =6 C $^{-1}$ = B $^{-1}$. Then, by part (i), the existence of the edge ba ∈ B $^{-1}$ yields the edge c ^{0}a ∈ D $^{-1}$, so ac^0 ∈ D =6 C . This is a contradiction. □

4.2. ALGORITHM FOR FINDING TRANSITIVE ORIENTATIONS

Theorem 4.4. *Let* A *be an implication class of an undirected graph* G = (V,E)*. Exactly one of the following statements holds:*

(i) A = A^ = A $^{-1}$

(ii) A ∩ A $^{-1}$ = *∅,* A *and* A^{-1} *are transitive and the only possible transitive orientations of* A^.

Proof. (i) Suppose A ∩ A $^{-1}$ =6 ∅. Let ab ∈ A ∩ A $^{-1}$, so ab Γ* ba. For any cd ∈ A, cd Γ * ab and dc Γ* ba hold. Since Γ* is an equivalence relation, then cd Γ * dc and dc ∈ A. Thus A = A^.

(ii) Suppose A ∩ A $^{-1}$ = ∅. Let ab,bc ∈ A. If ac /∈ E, then ab Γ cb, so cb ∈ A and hence bc ∈ A $^{-1}$; a contradiction. So it is ac ∈ E.

Let B ____

be the implication class of G containing ac. Suppose A =6 B. We apply statement (i) of the triangle lemma to the edge ab ∈ A and thus obtain ab ∈ B. So ac ∈ A and A is transitive. It also follows that A is $^{-1}$ transitive.

Moreover, A is an implication class of A^. By Theorem 4.1, A and A^{-1} are the only transitive orientations of A^. □

Corollary 4.5. *Every color class of an undirected graph* G *either has exactly two transitive orientations, one of which is the inverse of the other, or it has no transitive orientation at all. If* G *contains a color class that has no transitive orientation,* G is *not a comparability graph.*

4.2Algorithm for finding transitive orientations

In this section we will develop an algorithm to decide whether a given graph is a comparability graph. Let G = (V,E) be an undirected graph. A decomposition E = $B\hat{}_1$ + B^ $_2$ + --- + B^ of $_k$ the edge set E is called a G-decomposition of E if B is an implication class of B^ $_i$ $^+$ --- + $B\hat{}_k$ $_i$ for each i = 1,2,...,k. A sequence of edges [x $_1$y $_1$,x_2 y_2 ,...,x_k y $_k$] is called a *decomposition scheme of* G if there exists a G-decomposition E = B^ $_1$ + ---$B\hat{}_k$ with x_i y_i ∈ B_i for i = 1,...,k. Obviously, for any G-decomposition there can be many different decomposition schemes (any representative set of B^ $_i$). On the other hand, a G-decomposition is uniquely determined by specifying a decomposition scheme.

Obviously, a decomposition scheme and the corresponding G-decomposition can be easily computed using a Greedy procedure. Algorithm 6 shows the procedure.

An important observation is that removing edges causes implication classes to merge if necessary. In the following we will describe in more detail how and under what circumstances this happens. Before that, we need the notion of a multiplex.

```
Let G = (V,E) be undirected graph. Initial i = 1 and E $_1$
= E.
Step 1: Choose any edge e $_i$ = x $_i$$y_i$ ∈ E $_i$
Step 2: Enumerate the implication class B $_i$ of E $_i$ that $_i$ contains x $_i$y.
Step 3: Define E $_{i+1}$ = E $_i$ - B^ $_i$.
Step 4: If $E_{i+1}$ = ∅, set k = i and stop. Otherwise, increase i by 1 and go to step 1.
```

Algorithm 6 : Decomposition algorithm

Let G = (V,E) be an undirected graph. A complete subgraph (V $_S$,S) with r + 1 vertices is called a *simplex* with *rank* r if every undirected edge ab^ of S belongs to a different color class of G. The subgraph (V

$_M$,M) of G generated by a simplex S of rank r with M = {ab ∈ E | ab Γ^* xy for an xy ∈ S} is called a *multiplex of* rank r.

Theorem 4.6. *Let* A *be an implication class of an undirected graph* G = (V,E) *and let* D be *an implication class of* E - A^. *Exactly one of the following statements holds.*

(i) D *is an implication class of* E *and* A *is an implication class of* E - D^ .

(ii) D = B + C *where* B *and* C *are implication classes of* E and A^ + B^ + C^ *is a multiplex of* E *with rank 2.*

Proof. Removing A^ from E may cause some implication classes of E to merge. Let D be the union of k implication classes of E.

We first consider the case that k ≥ 2; then there exists a triangle of vertices a,b,c with bc ∈ A^ and either ac ∈ B and ab ∈ C, or else ca ∈ B and ba ∈ C, where B and C are different implication classes of E contained in D. Without restriction of generality, suppose that ac ∈ B and ab ∈ C. (The other case is identical for D $^{-1}$.) Suppose B = C $^{-1}$. Then ba,ac ∈ B, but bc /∈ B. By Theorem 4.4, then B = B^ = B , $^{-1}$which implies that B = C, a contradiction. So B^ ∩ C^ = ∅ holds, and G{a$_{,b,c}$} is indeed a three-colored triangle, A^ + B^ + C^ thus a multiplex of rank 2.

Any Γ-chain in E - A^ containing edges from B^ and C^ cannot contain edges from further implication classes, since all triangles in E with an edge in A^ and an edge in B^ (resp. C^) have the third side in C^ (resp. B^) according to the triangle lemma. So k = 2 and D = B + C.

We now consider the case k = 1 and show that then A is an implication class of E - D^. From what we have already proved, we know that if A is not an implication class of E - D^, then D^ + A^ + A^ is $_1$ a multiplex of rank 2 in E for a third implication class A $_1$ of E. But this contradicts the assumption that D is an implication class of E - A^; a contradiction of k = 1. So A is indeed an implication class of E - D^ . □

4.2. ALGORITHM FOR FINDING TRANSITIVE ORIENTATIONS

Theorem 4.7 (TRO Theorem). *Let* G = (V,E) *be an undirected graph with* G-decomposition E = B^ $_1$ + --- + B^$_k$. *The following statements are equivalent:*

(i) F = B $_1$ + B $_2$ + --- + B $_k$ *is a transitive orientation of* G

(ii) G = (V,E) *is a comparability graph.*

(iii) $A \cap A^{-1} = \emptyset$ *for each implication class* A *of* E.

(iv) $B_i \cap B_i^{-1} = \emptyset$ *for* i = 1,...,k.

(v) Every closed path of edges $_1$v v $_{32}$,v v ,...$_2$,v v_{q1} ∈ E *such that* v $_{q-1}$v $_1$,v *v* $_{q2}$,v v_{i-1} ∈/ $_{i+1}$ E *(for* i = 2,...,q - 1*) has even length.*

(ii) ⇒ (iii) ⇒is just the statement of Theorem 4.1. *Proof.* (i) (ii) obviously holds, F transitive orientation of G.

⇒ (iii) (iv). We use complete induction. Since B_1 is an implication class of E, B holds $_1$ $\cap B_1^{-1} = \emptyset$. If k = 1, we are done. We now assume that the implication holds for G all -decompositions of graphs of length less than k. In particular, it holds for E - B^ $_1$.

Let D be an implication class of E-B^$_1$. By Theorem 4.6, D is either an implication class of E (then $D \cap D^{-1} = \emptyset$) or D = B+C, where B and C are implication classes of E such that B^ ∩ C^ = ∅. The latter implies that

$$D \cap D^{-1} = (B + C) \cap (B^{-1} + C^{-1})$$

$$= (B \cap B^{-1}) + (C \cap C^{-1}) = \emptyset.$$

Thus, inductively, $B_i \cap B^{-}{}_i{}^1 = \emptyset$ holds for i = 2,...,k.

(iv) (i). Let E = B^ $_1$ + --- + B^$_k$ be a G-decomposition of E with $B_i \cap B^{-}{}_i{}^1 = \emptyset$. By Theorem 4.1,⇒ B is $_1$transitive. If k = 1, then the implication holds. We now assume that the implication holds for all G-decompositions of graphs of length less than k. By this assumption, F = B $_2$ + --- + B is $_k$a transitive orientation of E - B^ $_1$. It remains to show that B $_1$ + F is transitive.

Let ab,bc ∈ B $_1$ +F. If both edges are in B_1 or both are in F then it follows from the transitivity of B and F $_1$ respectively that ac ∈ B $_1$ +F as well. So we assume that ab ∈ B $_1$ and bc ∈ F. (The case that ab ∈ F and bc ∈ B $_1$ goes analogously.) By premise, ab 6 Γ $*$ cb, so ac ∈ E. Suppose ac /∈ B $_1$ +F. Then ca ∈ B $_1$ +F holds. From ca ∈ B $_1$ and ab ∈ B it $_1$ follows by means of the transitivity of B $_1$, that cb ∈ B $_1$; a contradiction. Similarly, from ca ∈ F and bc ∈ F , it follows that ba ∈ F ; also a contradiction. Consequently, ac ∈ B $_1$ +F and hence F = B $_1$ + --- + B is $_k$ a transitive orientation of E.

cation class⇔ A of E with $A \cap A^{-1}$ =6 ∅. Suppose $v_{12} \in A \cap A^{-1}$. By Lemma 4.2 (ii) (v). If G is not a comparability graph, then (by statement (iii)) there exists an impliexist a Γ-chain

$$v1v2 \ulcorner v3v2 \ulcorner v3v4 \ulcorner --- \Gamma\ vqvq\text{-}1 \ulcorner vqvq+1 = v2v1 .$$

We then consider the closed path $v_1\ {}_2v_3\ {}_2$,v v ,...,v v $_{q-1q}$,v $v_q\ {}_1$. By construction, q is odd because in the chain the index of the first node in each case is odd. Thus, the sequence of edges forms exactly such a closed path.

Conversely, if E contains such a closed path of odd length q, then

$$v1v2 \ulcorner v3v2 \ulcorner v3v4 \ulcorner --- \Gamma\ vqvq\text{-}1 \ulcorner vqv1 \ulcorner v2v1 .$$

It follows that for the implication class A_2 containing v $_1$v, $A \cap A^{-1}$ =6 ∅. So G (by statement (iii)) is not

a comparability graph. □

By combining the TRO theorem with the decomposition algorithm, we obtain a comparability graph detection algorithm. Moreover, this also allows us to compute a transitive orientation, if it exists; see Algorithm 7.

```
1 Start
        2   Initialize: i1; E i  E; F     ∅
        3   Choose x_i y_i ∈ E←i arbitrarily; ← ←
        4   Enumerate implication class B i of E i containing x iy;i
        5   if B i ∩ B ⁻i¹ = ∅ then
        6add B i to F;
        7   otherwise
        8gib from "G is not a comparability graph";
        9STOP;
        10  End
        11  Egg+1     Ei - B^i;
        12  if←Ei+1 = ∅ then
13ki;
14STOP; ←
15otherwise
16ii        + 1;
17go to line 3;←
```

```
18
    End
19 End
```

Algorithm 7 : TRO algorithm

One can show that the algorithm can be implemented with linear space and at most quadratic time requirements.

Theorem 4.8. *recognizing comparability graphs and computing a transitive orientation can be solved in* $O(\Delta \cdot |E|)$ *time and with* $O(|V| + |E|)$ *space; here* Δ *denotes the maximum node degree.*

In fact, both problems can be solved even in linear runtime. Interestingly, the problem of testing whether a given orientation is transitive is equivalent to the problem of multiplying two matrices. According to the current state of knowledge, this requires a significantly larger runtime.

4.3Coloring and other problems on comparability graphs

For any *acyclic* (not necessarily transitive) orientation F of an undirected graph $G = (V,E)$, an associated strict partial order can be defined by $x < y$ exactly if there exists a nontrivial path from x to y. We now define a *height function* h as follows: $h(v) = 0$ if v is a sink; otherwise $h(v) = 1 + \max\{h(w) \mid vw \in F\}$. The function h can be computed in linear time using recursive depth-first search. (Practice!) The function h is a true nodal coloring, though not necessarily a minimal one. The number of colors used is equal to the length of a longest path in F. An unfavorable choice of F may lead to the use of many colors; however, if F is transitive, the situation is more favorable.

Suppose G is a comparability graph and F is a transitive orientation of G. In this case, due to the transitivity of F , each path in F corresponds to a clique of G. Thus, the height function in this case yields a coloring with exactly $\omega(G)$ colors. Moreover, comparability graphs are hereditary, so it holds that $\omega(G_A) = \chi(G_A)$ for any induced subgraph G_A of G. This shows the following theorem.

Theorem 4.9. *Comparability graphs are perfect.*

The following famous theorem is a direct consequence of this and the theorem about perfect graphs (Theorem 2.8).

Theorem 4.10 (Dilworth's Theorem). Let $(X,\leq)$ *be a partially ordered set. The minimum number of linearly ordered subsets (*chains*) to cover* X *is equal to the maximum size of a subset of* X of *which no two elements are comparable (*antichain*).*

Using the algorithm from Section 4.2, we can compute in a transitive orientation in $O(\Delta|E|+|V|)$ steps. Using the height function, we can then gain a minimal coloring in $O(|V|+|E|)$ time. By finding a longest path, we can also compute a maximal clique.

Finally, we show the computation of α(G). Let (V,F) be a transitive orientation of G. We transform this into a flow network with two additional nodes s and t and edges sx and yt for each source x and each sink y of F. Moreover, we assign each node a lower bound of 1 for capacity. The edges emanating from s are also assigned a cost of 1 for each unit of flow transported.

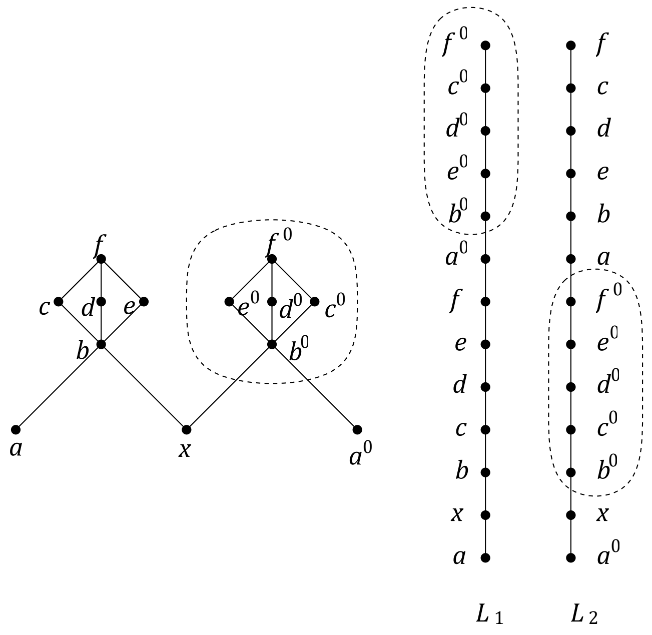


Figure 4.4: Example partial order P of dimension 2 and an associated minimal realizer of size 2.

The resulting flow problem can be easily described as a minimum cost flow problem and solved in polynomial time. Such a flow can always be decomposed into a set of st paths. Each of these paths then corresponds to a clique, so that we obtain a total overlap of cliques. Cost minimization ensures that a minimal clique overlap is chosen. Since comparability graphs are perfect, α(G) = k(G) holds, and we have thus computed the value α(G).

4.4The dimension of partial orders

Let (X,P) be a partial order. We follow the English notation and denote (X,P) also as *poset* (partially ordered set). Each partial order can be extended to a linear order L of X using topological sorting. Let L(P) be the set of all linear extensions of P. Any subset L ⊆ L(P) satisfying $\top_{L\in L}$ L = P is called a *realizer* of P; its size is |L|. Here ab ∈ $\top_{L\in L}$ L is exactly if ab ∈ L for every L ∈ L.

Obviously, L(P) is itself a realizer of P. We define the dimension of P, dimP to be the smallest possible size of a realizer for P. Such a realizer is also called a *minimal realizer* for P .

Lemma 4.11. *Let* (X,P) *be a poset. For every* Y ⊆ X, dimP$_Y$ ≤ dimP *holds.*

Proof. Obviously, restricting a realizer $\mathcal{L}$ of P to the elements of Y yields a realizer of P_Y. The statement follows by choosing $\mathcal{L}$ as the minimal realizer of P . □

Theorem 4.12. *Let* G *be the comparability graph of a poset* P*. Then* $\dim P \leq 2$ *holds exactly if the complement graph* $\overline{G}$ *is transitive orientable.*

Proof. Let F be a transitive orientation of $\overline{G}$. It is not hard to see that $\{P + F, P + F^{-1}\}$ is a realizer of P . Conversely, let $\{L_1, L_2\}$ be a realizer of P.

Then $F = L_1 - P = (L_2 - P)^{-1}$ is a transitive orientation of $\overline{G}$. Indeed, if $ab, bc \in F$ but $ac \notin F$, then the transitivity of L_1 implies that $ac \in P$ and the transitivity of L_2 implies that $ca \in P$; a contradiction. □

It follows directly from the previous theorem that two partial orderings having the same comparability graph either both have dimension at most 2 or both have dimension genuinely greater than 2. In fact, a stronger statement holds.

Theorem 4.13 (without proof). *If* P *and* Q *are two orders with the same comparability graph* G*, then* $\dim P = \dim Q$ *holds.*

Chapter 5

Split graphs

An undirected graph G may have one or more of the following properties:

Property V: G is a comparability graph.

Property $\overline{V}$: $\overline{G}$ is a comparability graph (G is a *co-comparability graph*).

Property C: G is chordal.

Property $\overline{C}$: $\overline{G}$ is chordal (G is *co-chordal*).

In fact, these four properties are independent of each other. There are example graphs for each of the 16 possible combinations. (Exercise!?) In the following three chapters, we will look at three of these combinations in more detail.

5.1Characterization of Split Graphs

An undirected graph G = (V,E) is a split *graph* if there exists a decomposition V = S + K such that S is an independent set and K is a complete set; see Figure 5.1. The edges between S and K are not subject to any constraints. In general, the decomposition V = S + K of a split graph is neither unique, nor is S or K necessarily a maximal independent set or maximal clique.

Since the complement of an independent set is complete and vice versa, the following theorem follows directly.

Theorem 5.1 An *undirected graph* G = (V,E) *is a split graph if and only if its*

Complement $\overline{G}$ *is a split graph.*

Theorem 5.2 *Let* G *be a split graph whose nodes have been decomposed into an independent set* S *and a complete set* K. *Then exactly one of the following statements holds:*

(i) |S| = α(G) *and* |K| = ω(G)

(in this case the parition S + K is *unique)*

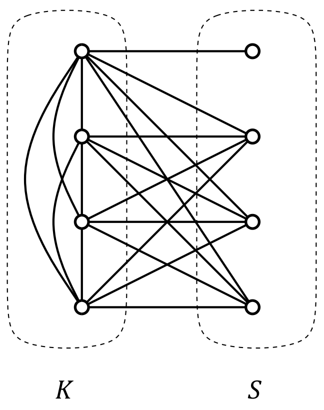


Figure 5.1: A split graph with one of its four possible partitions in S and K. Moving one of the degree-4 nodes from S to K again yields a valid decomposition.

(ii) $|S| = \alpha(G)$ *and* $|K| = \omega(G) - 1$

(in this case there exists an $x \in S$ *such that* $K + \{x\}$ *is complete).*

(iii) $|S| = \alpha(G) - 1$ *and* $|K| = \omega(G)$

(in this case there exists a $y \in K$ *such that* $S + \{y\}$ is *independent).*

Proof. Since an independent set and a complete set share at most one node, the sum $\alpha(G) + \omega(G)$ is either equal to $|V|$ or equal to $|V| + 1$.

If $\alpha(G) + \omega(G) = |V|$, then case (i) exists. Suppose there were another partition $V = S^0 + K^0$. Let $\{x\} = S \cap K^0$ and $\{y\} = S^0 \cap K$; the cuts are nonempty, otherwise $S = S^0$ and $K = K^0$ would follow. Moreover, such a cut can of course contain at most one element. Moreover, by definition it holds that $x \in S$ and $y \in K$, so x =6 y. Consequently, $K^0 = K - \{y\} + \{x\}$ and $S^0 = S - \{x\} + \{y\}$ holds. If x and y are adjoint, then $\{x\} + K$ is a larger clique, otherwise $\{y\} + S$ is a larger independent set; a contradiction. So the decomposition $V = S + K$ is unique.

If $\alpha(G) + \omega(G) = |V| + 1$, then case (ii) or (iii) exists. We prove only case (ii), case (iii) goes analogously. Let $|S| = \alpha(G)$, $|K| = \omega(G) - 1$ and let K be ^{0} a clique of size $\omega(G)$. Since $S + K$ is a partition and K^0 is larger than K, $S \cap K$ must be ^{0} nonempty and therefore of size 1. Let $\{x\} = S \cap K^0$. Consequently, $K^0 = K + \{x\}$ is complete. □

Theorem 5.3 *Let* G *be an undirected graph. The following statements are equivalent:*

(i) G *is a split graph,*

(ii) G *and* $\overline{G}$ *are chordal,*

(iii) G *does not contain an induced subgraph isomorphic to* $2K_2$, C_4 *or* C_5.

Suppose⇒G contains an induced chordless circle C of length at least 4. *Proof.* (i) (ii). Let G = (V,E) be a split graph with decomposition V = S + K as above.
Then at least one, but at most two, of the nodes of C are contained in K. In the latter case, the two nodes on C must be adjacent. But then in both cases S contains a pair of nodes connected by an edge. Consequently

G is chordal. According to Theorem 5.1, $\overline{G}$ is also a split graph and must therefore also be chordal.

(ii) $\overline{\text{(iii)}}$. Obviously, C_4 and C_5 are not chordal, and 2K is ${}_{2\,4}$ not chordal because $2K_2$ = C.

co-chordal. ⇒
Cliques chosen so that⇒ G has as few edges ${}_{V-K}$ as possible. We need (iii) (i). Let K be a (cardinality) maximal clique of G that is among all maximal
show that S = V - K is independent.

Suppose G_S contains an edge xy. Due to the maximality of K, no node of S can be adjacent to every node from K. If x and y were both adjacent to all nodes from K except the same node z, then K-{z}+{x}+{y} would be a larger complete set than K. Consequently, there exist two distinct nodes u,v ∈ K with xu /∈ E and yv /∈ E.

Since G contains neither an induced copy of $2K_2$ nor an induced copy of C_4, it follows that exactly one of the edges xv and yu in G is. We assume without restriction of generality that xv /∈ E and yu ∈ E. Now let w be any vertex of

K - {u,v}. If w were not adjacent to any of the nodes x,y, then G{$x_{,y,v,w\}=\tilde{}2K2}$. If w were not adjacent to y but were adjacent to x, then G{$x_{,y,u,w\}=\tilde{}C\ 4}$. Consequently, y is adjacent to every node of K - {v} and K^0 = K - {v} + {y} is a cardinality-maximal clique.

Since G_{V-K0} cannot have fewer edges than G_{V-K}, it follows from the fact that x is adjacent to y but not to v that there must be a vertex t =6 y in V - K adjacent to v but not to y. The edge tx must be in E, otherwise {t,x,y,v} would ${}_2$ induce a copy of 2K. Similarly, tu /∈ E, otherwise {t,x,y,u} would induce a copy of C. ${}_4$ But then {t,x,y,u,v} induces a copy of C_5; a contradiction. Foglich is S = V - K independent and G a split graph. □

5.2Grad sequences and split graphs

A sequence $\Delta = [d_1, d_2, ..., d_n]$ of integers with $n - 1 \geq d \geq_{1\,2}$ --- $\geq d \geq_n$

0 is called *graphical* if there exists an undirected graph that has Δ as a degree sequence. For example, [2,2,2,2] is graphical because it is the degree sequence of C_4. In fact, C_4 is the only graph with this degree sequence. The sequence [2,2,2,2,2], on the other hand, corresponds to both $2K_3$ and C_6. It is easy to specify sequences that are not graphs; for example, [1,1,1] and [4,4,2,1,1]. A simple necessary condition is that Pd_i must be even. But the previous example shows that this is not sufficient.

We now give two classical characterizations of graphical sequences without proofs

Version from 19 October 2021, 14:25

on.

Theorem 5.4 (Havel '65, Hakimi '62). *A sequence Δ of integers with* $n - 1 \geq d \geq_1$ --- $\geq_2 d \geq_n 0$ *is graphical if and only if the modified (non-ascending sorted) sequence*

$$\Delta^0 = [d_2 - 1, d_3 - 1, ..., d_{d1+1} - 1, d_{d1+2}, ..., d_n]$$

is graphic.

Theorem 5.5 (Erdős, Gallai '60). *A sequence of integers* $n - 1 \geq d \geq_{1\,2}$ --- $\geq d \geq_n 0$ *is graphical if and only if.*

(i) $P^n_{i=1} d_i$ *straight, and*

(ii) $P^r_{i=1} d_i \leq r(r - 1) + P^n_{i=r+1} \min\{r, d_i\}$ *for* $r = 1,2,...,n - 1$.

Inequality (ii) is called the r-th Erdős-Gallai inequality (EGU). Both theorems imply efficient algorithms for testing whether a given sequence is graphical or not. (Exercise!)

As we have seen before, a degree sequence does not uniquely describe a graph in general. Therefore, the question arises which graph properties are completely determined by the degree sequence. For example, it is relatively easy to show that transitive oriented complete graphs are completely characterized by the sequence of input degrees. There is also a characterization of the degree sequences of trees. (Exercise!) In the following we will study this problem for split graphs.

Let Δ = [d_1 ,...,d_n] be a sequence of integers with n - 1 ≥ d ≥$_{1\,2}$ --- ≥ d ≥$_n$ 0, and let ζ = [0,1,2,...,n - 1]. We now consider the position i of the sequences at which Δ overtakes the sequence ζ. Let m be the largest index i such that d_i ≥ i - 1. It is either m = n, in which case Δ is the degree sequence of K $_n$, or d_m ≥ m - 1 and d $_{m+1}$ < m.

Split graphs can be characterized as those graphs for which the mth EGU is satisfied with equality, where m is as defined before. Intuitively, the position m just represents the transition in the list from the degrees of the clique nodes to the degrees of the nodes from the independent set.

Theorem 5.6. *Let* G = (V,E) *be an undirected graph with degree sequence* d ≥$_1$ d ≥$_2$ --- ≥ d $_n$*, and let* m = max{i | d ≥$_i$ i - 1}*. Then* G is *a split graph if and only if.*

mn

Xd $_i$ = m(m - 1) + X d_i .

i=1i=m+1

Furthermore, then ω(G) = m*.*

Proof. The theorem holds securely if G is complete. Thus, we can assume that d_m ≥ m - 1 and d $_{m+1}$ < m. Since Δ is non-increasingly sorted, min{m,d $_i$} = d $_i$ for i ≥ m + 1. Therefore, the m-th EGU simplifies to.

mn

s = Xd_i ≤ m(m - 1) + X d_i . (5.1)

i=1i=m+1

Let K be the set of the first m nodes with maximum degree. The left summand of equation (5.1) can be $_2$ decomposed into two contributions s = s $_1$ + s with

$s_1 = Xx_K |\{z \in K \mid xz \in E\}| \leq m(m - 1)$ (5.2)

$\times$ $\in$

$s_2 = |y \notin K \mid xy \in E\}|$

$y \notin K i{=}m{+}1$

$x \in K$

n

(5.3)

In inequality (5.2) equality holds exactly if K is complete. On the other hand, in inequality (5.3) equality holds if and only if V - K is an independent set. Hence G is a split graph if inequality (5.1) is satisfied with equality.

$= X |\{x \in K \mid xy \in E\}| \leq X d_i$.

Conversely, let G = (V,E) be a split graph. By Theorem 5.2, V can be decomposed into an independent set S and a complete set K such that |K| = ω(G). Each node in K has degree at least |K| - 1, and, since K is cardinality-maximal, all nodes S have degree at most |K| - 1. Thus, we can assume that the nodes are ordered such that $K = \{v_1, ..., v_{|K|}\}$ and $S = \{v_{|K|+1}, ..., v_n\}$, where $\deg(v_i) = d_i$. Moreover, $d_{|K|} \geq |K| - 1$ and $d|K|+1_{\leq |K|} - 1 < |K|$ holds, so ω(G) = |K| = m. Thus, since K is complete and S = V -K independent, inequalities (5.2) and (5.3), and hence inequality (5.1), are satisfied with equality. □

Corollary 5.7. *If* G *is a split graph, then any graph with the same degree sequence is also a split graph.*

Chapter 6

Permutation graphs

In this chapter we consider a very basic class of perfect graphs with many applications. Let π be a permutation of the numbers 1,2,...,n. We consider π as a sequence $[\pi_1, \pi_2, ..., \pi_n]$; where π denotes $_i$ the image π(i). Analogously, $(\pi^{-1})_i$, which we $^{-1}_i$ write as π for short, gives the position of π in the $_i$ sequence.

Given such a permutation π, we can define an undirected graph G[π] as follows. The nodes of G[π] are numbered from 1 to n, and two nodes are connected if the node with the larger number in π is to the left of the node with the smaller number (i.e., they have been permuted by the permutation). The graph G[π] is also called an *inversion graph*.

Formally, G[π] can be defined as follows. If π is a permutation of the numbers 1,2,...,n, then the inversion graph G[π] = (V,E) is defined as follows:

$$V = \{1,2,...,n\}$$

and

$$ij \in E \Leftrightarrow (i - j)(\pi^{-1}_i - \pi^{-1}_j) < 0.$$

An undirected graph G is a permutation *graph* if there exists a permutation π with $G \cong G[\pi]$. Figure 6.1 shows an example of a permutation graph.

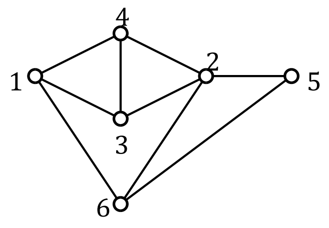


Figure 6.1: The permutation graph G[4,3,6,1,5,2]

6.1Characterization

Permutation graphs have a number of interesting properties. For example, consider what happens when we flip the permutation π. Then every pair of numbers that was previously in π in the correct order is now in the wrong order, and vice versa. So the permutation graph obtained is the complement graph of G[π]. Thus, if π denotes $^{\rho}$ the permutation obtained by inverting the sequence π, then

$$G[\pi^{\rho}\,]=\tilde{}\,\overline{G[\pi]}\,.$$

So the complement of a permutation graph is also a permutation graph.

Another property of graphs G[π] is that they are transitive orientable. We obtain a transitive orientation F by directing each edge towards the endpoint with the larger number. Now, if ij ∈ F and jk ∈ F, then i < j < k and $\pi^{-1}_{i} > \pi^{-1}_{j} > \pi^{-1}_{k}$, so ik ∈ F. In fact, even the following stronger statement holds.

Theorem 6.1 An *undirected graph* G *is a permutation graph if and only if* G *and* $\overline{G}$ *are comparability graphs.*

Proof. Suppose G=˜G[π]; then G is comparability graph since G is transitive orientable. Moreover, $\overline{G}$=˜G[π$^{\rho}$] is also a permutation graph and hence also a comparability graph.

Conversely, let (V,F_1) and (V,F_2) be transitive orientations of G = (V,E) and $\overline{G}$ = (V,$\overline{E}$). We claim that (V,$F_1 + F_2$) is an acyclic orientation of the complete graph (V,E + $\overline{E}$). Suppose $F_1 + F_2$ contained a circle [v_0,v ,$_{21}$v ,...,v $_{l-1}$,v $_0$] of smallest length l. If l > 3, then the circle can be $_0$ shortened by v_0 v $_2$ or v v , $_2$which contradicts minimality. On the other hand, if l = 3, then at least two of the edges of the circle are in the same F $_i$, which implies that F $_i$ is not transitive. So (V,$F_1 + F_2$) is acyclic. Analogously, it can be shown that (V,$F^{-1}_{1} + F_2$) is acyclic.

We now construct a permutation π such that G=˜G[π]. An acyclic orientation of a complete graph is always transitive and determines a unique linear ordering of the nodes (using topological sorting). We now proceed in three steps:

Step I: Label the nodes using the $_2$ order determined by F $_1$ + F; the node x with input degree i - 1 is given the label L(x) = i.

Step II: Label the nodes using the order specified by F $_1^{-1} + F_2$; the node x with input degree i - 1 is given the label L ^{0}(x) = i.

Note that

$$xy \in E \Leftrightarrow [L(x) - L(y)][L^0(x) - L(^0y)] < 0, \qquad (6.1)$$

since exactly the edges in E are flipped between step I and step II.

Step III: Define π as follows: for node x, let L(x) = i, then set π^{-1}_i = L^0 (x). Thus, the permutation π transforms the label L into the label L ^{0}.

According to equation (6.1), π is the desired permutation and L is the desired graph isomorphism. ☐

Remark. Using the notions from Section 4.4, G is a permutation graph exactly when the transitive orientations of G, considered as partial orderings, have dimension at most 2. The two labels L and L constructed above. ^{0}

—

form a realizer of a transitive orientation of G.

Theorem 6.1 also describes an algorithmic procedure for detecting permutati-

—

onsgraph by computing a transitive orientation of G and G. By the procedure described in the proof, one then obtains a permutation π. This method requires O(n ^{3}) time and O(n ^{2}) space for graphs with n nodes.

Moreover, from transitive orientability follows another important relation between the permutation π and the cliques or independent sets of the graph
G[π].

Remark. The descending subsequences of π just correspond to the cliques of G[π]. The ascending subsequences of π just correspond to the independent sets of G[π].

A related but somewhat easier problem is whether, for a given numbering L of the nodes of a graph G with numbers 1,...,n, there exists a permutation π such that G = G[π]. In this case, the given numbering L is called a permutation *labeling*. A characterization of permutation labels will be presented in the exercise.

6.2Applications

Permutation graphs can be thought of as a class of cut graphs in the following way. Write the numbers 1,2,...,n horizontally from left to right in a line. Below that, write the sequence of numbers $\pi_1,\pi,\ldots_2,\pi_n$ also from left to right. Finally, draw straight lines connecting the two 1s, the two 2s, and so on. We also call this the *matching representation of* π; see Figure . Note that the i-th and j-th segments intersect exactly when the order of i and j in π is reversed; this is the same criterion as the existence of an edge in G[π]. Therefore, the intersection graphs of segments of a matching graph are exactly the permutation graphs.

Application 1

Suppose we have two sets X and Y of cities lying on two parallel straight lines. Suppose further that flight routes connect some of the cities in X with some

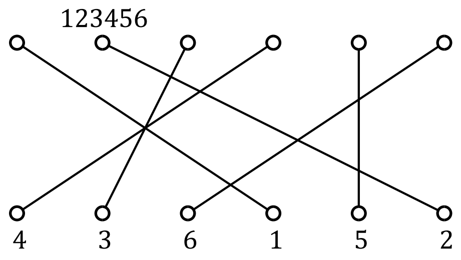


Figure 6.2: Matching representation of G[4,3,6,1,5,2].

of cities in Y to be served all at the same time. Our task is to determine a flight altitude for each of the flight routes so that overlapping routes are given different altitudes. This will ensure that there are no collisions during the flights. Obviously, this is a coloring problem.

The data are available as a bipartite graph embedded in the plane as in Figure 6.3a. We first number the flight paths by traversing the cities on the northern straight line from west to east. This gives us a matching representation and hence a corresponding permutation graph G[π]; see Figure 6.3b. Since assigning the flying ears is now equivalent to coloring the nodes of G[π], so that adjacent nodes get different colors; Figure 6.3c. For example, since G[π] is a comparability graph, this can be done using the algorithm from Section 4.3. In the next section, we will present a more efficient coloring algorithm for permutation graphs.

Application 2

Let I = {I_1 | i = 1,...,n} be a set of intervals of a straight line, where I_i = (x_i ,y_i) and |I_i | = y_i - $x_{i\ i}$ denotes the length of I . We further assume that the intervals that may overlap are sorted such that $x_1 \leq x \leq_2 --- \leq x_n$.

Let w_i be the cost of shifting interval I_i (we assume that the shift cost is independent of the shift distance). Find the cheapest shift of intervals such that (1) the order is maintained and (2) no intervals overlap after the shift. (The intervals could be, for example, the memory requirements of programs at certain points in time).

A solution might look like the following. We consider the oriented graph (I,F) with

$$(I_i,I_j) \in F \Leftrightarrow X\ |I_k| \leq y_j - x_i \qquad (i < j)\ .$$

$$i \leq k \leq j$$

Then two intervals are connected by F if the intervals between them can be shifted in such a way, that none of these j - i + 1 intervals

6.3. SORTING PERMUTATIONS WITH QUEUES

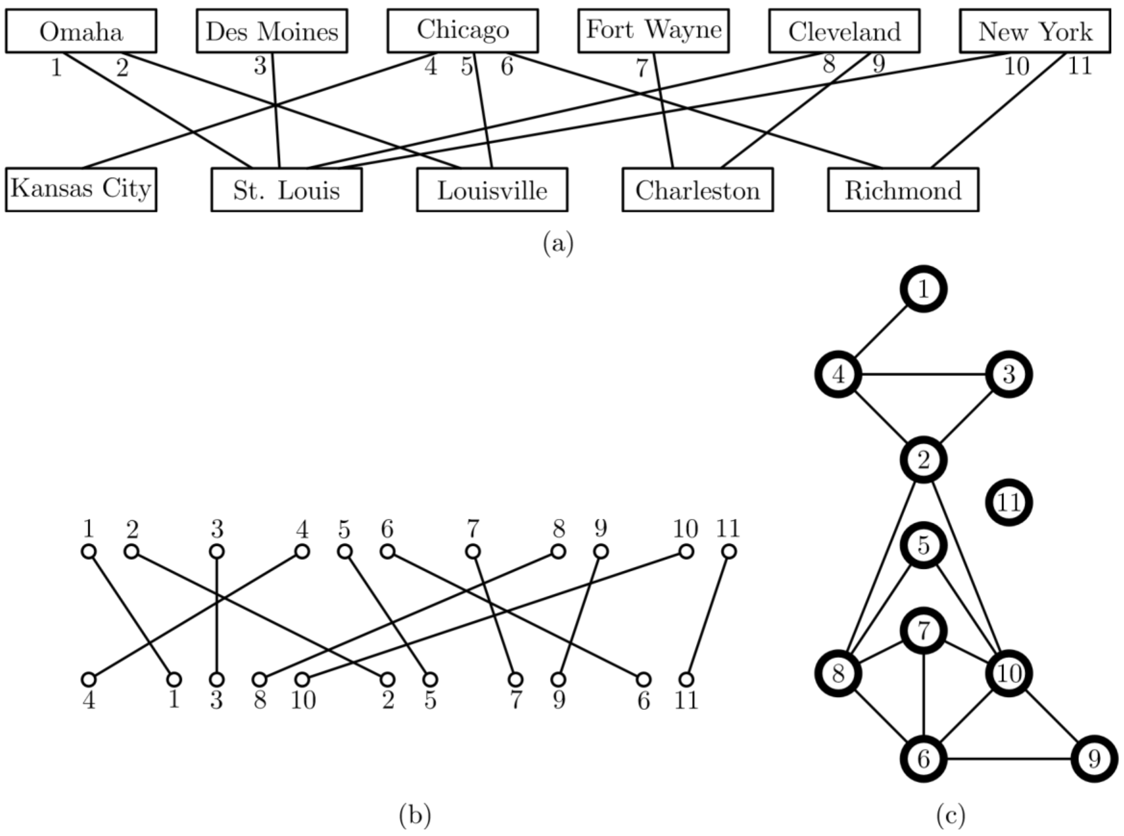


Figure 6.3: (a) A bipartite graph B representing flight routes between cities. (b) The matching representation of the permutation π constructed from the bipartite graph B. (c) The graph G[π]. Coloring the nodes of G[π] solves the height assignment problem for B.

(including the held I_i and I_j) overlap. It is not difficult to show that F is transitive. The solution to the problem is now to find a clique in F with maximum weight (to hold it, all other intervals are shifted). So, in other words, a clique of maximal weight must be found in the graph $(I,F+F^{-1}$ $)$. This graph is not only a comparability graph but even a permutation graph.

6.3Sorting Permutations with Queues

A queue is a simple data structure into which data is input at one end and extracted at the other end according to the FIFO (first-in-first-out) principle. We consider the problem of a permutation π of the numbers 1,2,...,n using a network of k parallel queues as in

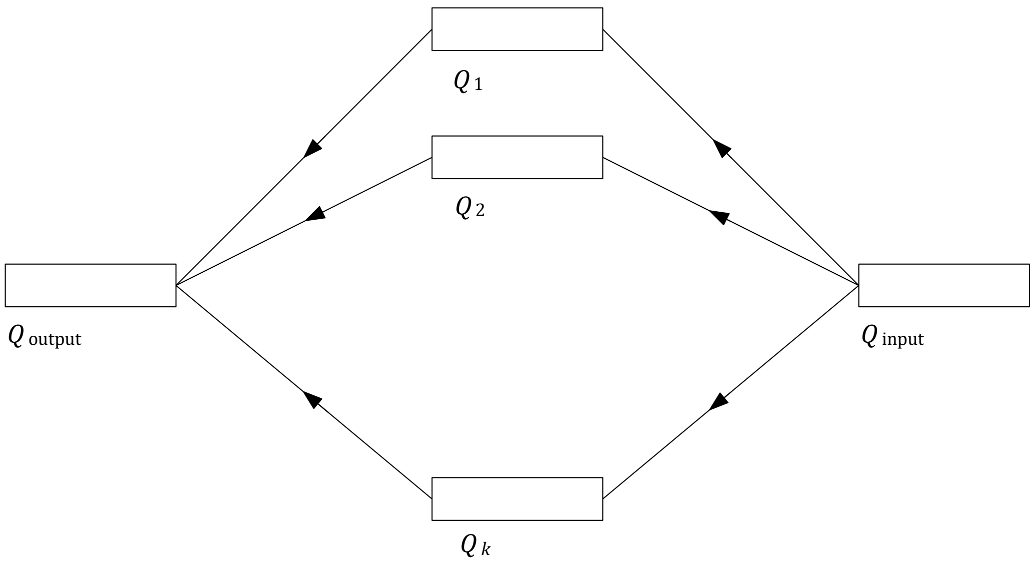


Figure 6.4: A network of k parallel queues. (Attention: Data flow from right to left!)

Figure 6.4. Initially, the entire permutation is in the input queue. Gradually, the numbers from the input move to the k inner queues, where they are temporarily stored until they are moved to the output queue. We assume that each of the queues has unbounded capacity and data may not be moved against the directions of the arrows. You can think of this as being like a marshalling yard where wagons are to be shuffled. Here, the number of tracks that can be used in parallel (i.e., the number of queues) is often limited. This leads to the following problem. Given a network of k parallel queues, characterize the permutations that can be sorted with it. Or, given a permutation π, how many queues are needed to sort it? Also, find an optimal sort in this case.

The central question is: When do two numbers have to pass through different queues? Exactly when they are interchanged in π, that is, they are in the wrong order. So if i and j are adjacent in G[π], they must pass through different queues.

Proposition 6.2. *Let π =* [π_1 ,...,π_n] *be a permutation of the numbers* {1,2,...,n}*. There is a bijection between the true* k-colourings *of* G[π] *and the successful sorting strategies for π in a network with* k *queues.*

Proof. We assign a different color to each queue Q_i . Now we sort the permutation with the k-network and color each number as it enters a queue with the corresponding color. Since nodes i and j connected in G[π] shift-queues, they also receive different colors.

Conversely, consider a real coloring of G[π] with colors 1,2,...,k. If the element x of the input has color c, then we send c to the queue Q_c. Suppose this strategy is unsuccessful. Then one of the queues, say Q contains $_m$ a pair of numbers x and y in the wrong order. But then x and y are adjacent in G[π] and yet colored with the same color, a contradiction. Obviously, this correspondence is a bijection. □

Corollary 6.3. *Let π be a permutation. The following numbers are equal.*

(i) The chromatic number of G[π]*,*

(ii) the minimum number of queues to sort π, (iii) the length of the longest

descending subsequence in π.

Proof. The equivalence of (i) and (ii) follows directly from Proposition 6.2; the proof also shows how the corresponding solutions can be transformed into each other.

The equality of (i) and (iii) holds, since such a longest subsequence of π corresponds to a maximal clique of G[π] whose size is just χ(G[π]), the permutation graph is perfect. □

The *canonical sorting strategy for* π inserts each number into the first available queue. This gives us a *canonical coloring* of G[π]. Algorithm 8 simulates this procedure. It computes a minimal coloring. Using a binary search for the implementation of line 4, we obtain a total running time of O(nlogn).

Theorem 6.4. *Let π be a permutation of the numbers* {1,2,...,n}*. The canonical coloring of* G[π] *as computed by Algorithm 8 is a minimal coloring.*

Proof. Clearly, Algorithm 8 computes a true χ-coloring of G[π]. We need to show that χ = χ(G[π]). It suffices to show that π contains a descending subsequence of length χ.

We consider the prior function v defined as follows: If COLOR(π_j) = i ≥ 2, let π denote $_{v(j)}$ the entry LAST(i - 1) during the jth iteration. Obviously, $\pi_{v(j)} > \pi_j$ and v(j) < j holds, since the entry π was $_{v(j)}$ at the end of Q_{i-1} , thus $_i$ forcing π_j into Q.

Now let π be$_{j\chi}$ a node with color χ. Then the sequence

$$\pi j1,\pi j2,...,\pi j\chi$$

with $\pi_{ji-1} = \pi_{v(ji)}$ for i = χ,χ - 1,...,2 the desired descending sequence. □

Input : Permutation π = [π_1,...,π_n] of numbers {1,...,n}.

Output : Coloring of the nodes of G[π] and chromatic number χ of G[π].

Procedure : In the jth iteration, π is$_j$ moved to the queue Q_i with the smallest index i for which it holds that π is $_j$ larger than the last entry in Q_i. Instead of storing the entire queue Q_i, only one entry LAST(i) is stored, which stores the last (largest) number in Q_i . The counter k stores the number of queues actually used.

```
1 Start
2 k0;
3for←j←←← 1 to←n←tue
4i                    Index of the first allowed queue;
5COLOR(π j)               i;
6LAST(i)π j;
7k              max{k,i};
8     End 10 End←
9     χk;
```

Algorithm 8 : Algorithm for canonical coloring of permutations.

To find a minimal clique overlap of G[π], we can simply apply Algorithm 8 to the inverse π^{ρ} π.

The running time of the algorithm is known for both. Interestingly, this is independent of the number of edges in→ G. Are problems in O(nlogn), provided the permutation π and the isomorphism G G[π] be-

the mappings are not known, however, we can either compute them or use the coloring algorithm for comparability graphs from Section 4.3.

Chapter 7

Interval graphs

Interval graphs were historically one of the first types of cut graphs. The question of how to recognize intersection graphs was raised by G. Hajös as early as 1957. We have already learned about interval graphs in Chapter 1.3, where we proved some basic properties. In this chapter, using the techniques from the previous chapters, we want to gain a more detailed understanding of interval graphs and related graph classes.

7.1Characterization

Theorem 7.1. *For an undirected graph* G, the *following statements are equivalent.*

(i) G *is an interval graph.*

(ii) G *does not contain a chordless* 4-circle *and its complement* $\overline{G}$ *is a comparability graph.*

(iii) The (inclusion) maximal cliques of G *can be linearly ordered such that for each node* x *of* G*, the maximal cliques of* G *containing* x *are consecutive in order.*

From Section 1.3.⇒ *Proof.* (i) (ii): This is just the statement of Proposition 1.6 and Proposition 1.8.

consider a transitive orientation⇒F of G. (ii) $\_\_$ (iii): we assume that G = (V,E) does not contain a chordless 4-circle and.

Assertion A. Let A_1 and A_2 be two distinct maximal cliques of G.

(a) There is an edge in F with an endpoint in A_1 and an endpoint in A_2.

(b) All edges of $\overline{E}$ connecting A_1 to A_2 have the same orientation in F.

Proof of Assertion A. (a) If there were no such edge in F, then $A_1 \cup A$ would be $_2$ a clique of G, a contradiction of maximality.

(b) Let ab ∈ F and dc ∈ F with a,c ∈ A_1 and b,d ∈ A_2. We carry this to the

Contradiction. If a = c or b = d, we get a contradiction directly from the transitivity of F . Thus, the four nodes must be different in pairs. Moreover one of the edges ad or bc must be contained in E, otherwise G would contain a chordless 4-

Circle. We assume without restriction of generality that ad ∈ E and consider the orientation in F. By means of the transitivity of F, it follows directly from ad ∈ F ac ∈ F and from da ∈ F it follows db ∈ F. This is impossible since these edges are inside cliques A and $_1$ A, $_2$ respectively. This concludes the proof of assertion A.

Now consider the following relation on the set C of maximal cliques of G:

A $_1$< A $_2$ exactly if an edge in F connecting A $_1$ and A$_2$ is directed towards A. $_2$ By assertion A, this defines an oriented complete graph on C. We claim that (C,<) is a transitive oriented clique, so it orders C linearly. Suppose that A $_1$< A $_2$ and A $_2$< A $_3$. Then there exist edges wx,yz ∈ F with w ∈ A $_1$, x,y ∈ A $_2$ and z ∈ A $_3$. If xz /∈ E or wy /∈ E, it follows that wz ∈ F and therefore A $_1$< A $_3$. So we can assume that wy,yx and xz all lie in E . Since G does not contain a chordless 4-circle, wz /∈ E follows and the transitivity of F implies wz ∈ F. Hence A $_1$< A $_3$. This proves the assertion with the transitive oriented clique.

We now assume that C $_m$ has been linearly arranged as A $_1$,A $_{,2}$...,A according to the above relation. Suppose there were cliques A $_i$< $_j$A < A $_k$ with ∈ ^{x} A $_i$, ∈ ^{x}/A $_j$ and x ∈ A $_k$. Since x /∈ A $_j$, there exists a node y ∈ A $_j$such that xy /∈ E. But A $_i$< A $_j$ implies xy ∈ F, whereas A $_j$< A $_k$ implies that yx ∈ F; a contradiction. Thus statement
(iii) proven.
(iii) (i) For each node x ∈ V, denote I(x) the set of all maximal cliques of G⇒ containing x. The sets

I(x) are intervals of the linear ordered set

(C,<). It remains to show that

xy ∈ E ⇔ I(x) ∩ I(y) =6∅(x,y ∈ V) .

This is obviously the case, since two nodes are adjacent if and only if they are contained together in a maximal clique. □

From Theorem 7.1 and Propositions 1.6, we directly obtain the following corollary.

Corollary 7.2 An *undirected graph* G *is an interval graph if and only if it is*

is chordal and its complement $\overline{G}$ *is a comparability graph.*

The coloring problem, as well as the computation of a maximum clique, a maximum independent set, or a minimum clique cover, can be solved in polynomial time using the algorithms from Chapters 3 and 4. Furthermore

construct a polynomial detection algorithm by combining Algorithm 1 (chordal graph detection) and Algorithm 7 (comparability graph detection).

Statement (iii) from Theorem 7.1 has an interesting formulation using matrices. A matrix whose entries are 0 or 1 has the *consecutive-ones property for columns* if the rows can be permuted so that the 1s in each column are consecutive. The clique matrix of an undirected graph G = (V,E) is a matrix M that has a column for each vertex in V and a row for each maximal clique C of G. The entry for column v and row C is 1 if and only if v ∈ C. Statement (iii) can then be formulated as follows.

Theorem 7.3. *an undirected graph is an interval graph if and only if its clique matrix has the consecutive-one property for columns.*

Proof. An order of maximal cliques of G corresponds exactly to a permutation of the rows of M. The theorem thus follows directly from Theorem 7.1. □

This also allows to obtain a more efficient detection algorithm, namely one with linear running time. For this, one first computes all maximal cliques using a perfect elimination scheme. Then, one uses the PQ-trees invented by Booth and Lueker to obtain a representation of all clique orders in which for each node all cliques in which it is contained are consecutive. For the data structure we refer to the lecture "Algorithms for planar graphs".

The oldest characterization goes back to Lekkerkerker and Boland and dates from 1962.

Theorem 7.4 (Leccerkerker and Boland '62, no proof). *An undirected graph* G *is an interval graph if and only if it satisfies the following conditions:*

(i) G *is chordal, and*

(ii) each three nodes of G *can be arranged such that each path from the first to the third node traverses the neighborhood of the second.*

A triple of nodes that do not satisfy property (ii) is called an *asteroidal triple*. Thus a graph is an interval graph if and only if it is chordal and does not contain an asteroidal triple.

7.2Applications

Interval graphs are one of the most useful modeling tools for real-world problems. The straight line on which the intervals lie can represent any one-dimensional unit. The linearity often follows from physical constraints, time constraints or a mapping by a cost function.

For many problems there are dedicated algorithms for interval graphs with often much better running times. In addition to the algorithms for coloring, clique covering, independent set, and maximal clique discussed in the lecture, the existence of a Hamiltonian path/circle, for example, can also be decided efficiently. For some problems, such as the problem of finding a cut with maximum weight, the complexity is still open.

We have already learned about a first application for distributing courses to rooms in the introduction. Now we want to look at two more examples.

Application 1

Let c_1 ,...,c_n be chemicals that must be cooled for storage under precisely defined conditions. Each of the chemicals c $_i$ must be stored at a constant temperature between t $_i$ and t $_i^0$ degrees. How many refrigerators with different temperatures are needed to store the chemicals?

For the solution, we consider the interval graph with nodes c_1 ,...,c_n , where we connect two nodes exactly when their temperature intervals overlap. According to the Helly property (Section 3.3), it follows that if {c ,...$_{i1}$,c $_{ik}$} is a clique of G, the intervals {[t $_{ij}$,t $_i^0{}_j$] | j = 1,...,k} have a common intersection t . A single refrigerator with temperature t is then sufficient to store them all together. Thus, a solution to the minimization problem is obtained by computing a minimal clique cover.

Application 2

Another application arises from the following problem. Let X be a set of data entries and let A be a set of subsets of X needed to answer individual queries. Is it possible to arrange the data in X in linear memory in such a way that for each A ∈ A the required data entries are in a contiguous memory area?

This problem can be easily written as follows. We consider a 0/1 matrix M with one row per data entry and one column for each query. Here, the entry in row i and column j is 1 if and only if the i-th data entry in the j-th
Query Needed. Obviously, the initial question is equivalent to whether M has the consecutive-one property for columns. By Theorem 7.3, this is the case exactly if the graph defined by M is an interval graph. It should be noted that this question can of course also be solved directly by means of PQ-trees.
7.3. PREFERENCE AND INDIFFERENCE

7.3Preference and indifference

Let V be a set. Suppose that a decision maker decides for each pair of elements in V that he strictly prefers one of them over the other or that the two elements are indifferent for him, i.e. almost completely equivalent.

With the help of these two relations (preference and indifference) two graphs H = (V,P) and G = (V,E) can be defined as follows. For each two different elements x,y ∈ V holds

xy ∈ Px is preferred over y, xy ∈ E ⇔⇔ x and y are

equivalent.

By definition, H is an oriented graph, G is an undirected graph, and $(V,P+P^{-1}+E)$ is complete. In fact, we can reasonably expect H to have some more structure. Given a rational decision maker, we can expect H to be at least acyclic. In fact, it is even reasonable to expect H to be transitive; if someone prefers y to x and z to y, it is not reasonable to expect x and z to be considered equivalent. So in what follows we require that H is transitive, that is, P is a partial order.

One way to quantify preferences is to use a so-called utility function u that assigns a real number u(x) to each element x ∈ V such that x is preferred over y if and only if u(x) is *sufficiently larger* than u(y). Let formally δ > 0 and u: V R. Then u is called *semi-order-.*

Utility function for a binary relation (V,P)→if the following condition is satisfied:

$$xy \in P \Leftrightarrow u(x) \geq u(y) + \delta \qquad (x,y \in V).$$

The following theorem answers the question which binary relations can be described by a half-order utility function.

Theorem 7.5 (Scott and Suppes '58, without proof). *A binary relation* (V,P) *has a half-order utility function if and only if the following conditions hold for all* x,y,z,w ∈ V :

(S1) P *is irreflexive;*

(S2) xy ∈ P *and* zw ∈ P *implies* xw ∈ P *or* zy ∈ P

(S3) xy ∈ P *and* yz ∈ P *implies* xw ∈ P *or* wz ∈ P.

A relation satisfying the three conditions (S1), (S2) and (S3) is also called a *semi-order*. Now consider the indifference relation G = (V,E) of a semiorder (V,P) admitting a semiorder utility function u . Two distinct nodes x and y are adjacent in G if and only if $|u(x) - u(y)| < \delta$. Thus G can be represented as an interval graph where each node x is represented by an open interval of length δ whose center is u(x). Thus indifference graphs are unit interval graphs, i.e., interval graphs that have a representation in which all intervals have the same length, without restriction length 1.

Theorem 7.6 (Roberts '69). *Let* G = (V,E) *be an undirected graph. The following statements are equivalent.*

(i) *There exists a real-valued function* u: V R *such that for every two different*

Node x,y ∈ V *holds* →

$$xy \in E \Leftrightarrow |u(x) - u(y)| < 1.$$

(ii) *There exists a semi-order* (V,P) *such that* $E = P + \overline{P}^{-1}$.

(iii) $\overline{G}$ *is a comparability graph and any transitive orientation of* $\overline{G}$ *is a semiorder.*

(iv) G *is an interval graph that does not contain an induced copy of* $K_{.1,3}$

(v) G *is a true interval graph.*

(vi) G *is a unit interval graph.*

Interval I_x = (⇒ u(x)- $\frac{1}{2}, u(x) + \frac{1}{2})$. Obviously, $I_x \cap I_y$ =6 ∅ holds exactly if |u(x)- *Proof.* (i) (vi): Let u be the function from (i). We associate with each node x the u(y)| < 1. So $\{I_x\}_{x \in V}$ is an interval representation of G with unit intervals. is an interval representation with unit intervals always real. ⇒ (vi) (v): since no unit interval can genuinely contain another unit interval,

Let⇒ {z_1,z_2 ,z_3}1 be an independent set and2 $_3$y be $_i$ adjoint to each of z . If $_{1\,2}$I $_{3zj}$ (v) (iv): Suppose G contained a subgraph G $_{y,z\,,z\,,z\,1,3}$ isomorphic to K .
the interval of I_z ,$_z$I ,I_z , which lies completely between the other two, then it follows that I_y must contain the interval I $_{zj}$ completely. This is a contradiction, so G cannot contain an induced subgraph isomorphic to $K_{.1,3}$

Let F⇒be a transitive orientation of $\overline{G}$. __ Clearly, ∈ $\overline{F}$ is irreflexive and thus satisfies (iv) (iii): since G is an interval graph, $\overline{G}$ = (V,$\overline{E}$) is a comparability graph.
Property (S1). Suppose it were xyF and zwF. Since G according to Theorem 7.1 has no

contains an induced circle of length 4, there exists another edge in E on the set of nodes {x,y,z,w}. By transitivity, it then follows that xw ∈ F or zy ∈ F ; thus, F satisfies property (S2). We now show property (S3). Let xy ∈ F and yz ∈ F, but xw /∈ F and wz /∈ F. By transitivity of F it follows that wx /∈ F and zw /∈ F and that wy /∈ F and yw /∈ F, but xz ∈ F. Accordingly, $Gx_{,y,z,w}$ is isomorphic to $K_{1,3}$. Contradiction.

(iii) ⇒ (ii): Follows directly.

7.3. PREFERENCE AND INDIFFERENCE

(ii) (i): If (V,P) is a semiorder, then there exists a real-valued function u^0 : V R and a number⇒ δ > 0 such that xy ∈ P exactly if $u^0(x) - u^0(y) \geq \delta$. We define→

$u(x) = u^0(x)/\delta$. Since $P + P^{-1} = E$ holds

$$xy \in E \Leftrightarrow |u(x) - u(y)| < 1.$$

□