CHAPTER 29

Tree-Structured Graphs

Andreas Brandstädt

Feodor F. Dragan

CONTENTS

29.1 Graphs with Tree Structure, Related Graph Classes, and Algorithmic Implications 752
29.2 Chordal Graphs and Variants 753
29.2.1 Chordal Graphs 753
29.2.2 Some Subclasses of Chordal Graphs 756
29.3 α-Acyclic Hypergraphs and Their Duals 757
29.3.1 Motivation from Relational Database Theory 757
29.3.2 Some Basic Hypergraph Notions 759
29.3.3 Hypergraph 2-Coloring 763
29.3.4 Kőnig Property 764
29.3.5 α-Acyclic Hypergraphs and Tree Structure 764
29.3.6 Graham's Algorithm, Running Intersection Property, and Other Desirable Properties Equivalent to α-Acyclicity 767
29.3.7 Dually Chordal Graphs, Maximum Neighborhood Orderings, and Hypertrees 770
29.3.8 Bipartite Graphs, Hypertrees, and Maximum Neighborhood Orderings 773
29.3.9 Further Matrix Notions 775
29.4 Totally Balanced Hypergraphs and Matrices 776
29.4.1 Totally Balanced Hypergraphs versus β-Acyclic Hypergraphs 776
29.4.2 Totally Balanced Matrices 778
29.5 Strongly Chordal and Chordal Bipartite Graphs 779
29.5.1 Strongly Chordal Graphs 779
29.5.1.1 Elimination Orderings of Strongly Chordal Graphs 779
29.5.1.2 Γ-Free Matrices and Strongly Chordal Graphs 782
29.5.1.3 Strongly Chordal Graphs as Sun-Free Chordal Graphs 783
29.5.2 Chordal Bipartite Graphs 786
29.6 Tree Structure Decomposition of Graphs 788
29.6.1 Cographs 788
29.6.2 Optimization on Cographs 790
29.6.3 Basic Module Properties 791
29.6.4 Modular Decomposition of Graphs 793
29.6.5 Clique Separator Decomposition of Graphs 794
29.7 Distance-Hereditary Graphs, Subclasses, and γ-Acyclicity 794
29.7.1 Distance-Hereditary Graphs 794
29.7.2 Minimum Cardinality Steiner Tree Problem in Distance-Hereditary Graphs 799

29.7.3 Important Subclasses of Distance-Hereditary Graphs 801
29.7.3.1 Ptolemaic Graphs and Bipartite Distance-Hereditary Graphs 801
29.7.3.2 Block Graphs 802
29.7.3.3 γ-Acyclic Hypergraphs 802
29.8 Treewidth and Clique-Width of Graphs 803
29.8.1 Treewidth of Graphs 803
29.8.2 Clique-Width of Graphs 805
29.9 Complexity of Some Problems on Tree-Structured Graph Classes 807
29.10 Metric Tree-Like Structures in Graphs 808
29.10.1 Tree-Breadth, Tree-Length, and Tree-Stretch of Graphs 808
29.10.2 Hyperbolicity of Graphs and Embedding Into Trees 810

29.1 GRAPHS WITH TREE STRUCTURE, RELATED GRAPH CLASSES, AND ALGORITHMIC IMPLICATIONS

The aim of this chapter is to present various aspects of tree structure in graphs and hypergraphs and its algorithmic implications together with some important graph classes having nice and useful tree structure. In particular, we describe the hypergraph background and the tree structure of chordal graphs (introduced in Chapter 28) and some graph classes which are closely related to chordal graphs such as chordal bipartite graphs, dually chordal graphs, and strongly chordal graphs as well as important subclasses.

As already defined in Chapter 28, a graph is *chordal* if each of its induced cycles has only three vertices (i.e., each cycle with at least four vertices has a so-called *chord*). The study of chordal graphs goes back to [1], and the many aspects of chordal graphs are described in surveys and monographs such as [2–5] and others. The interest in chordal graphs and related classes comes from applications in computer science, in particular, relational database schemes [6,7], matrix analysis, models in biology, statistics, and others. Chordal graphs are closely related to the famous concept of *treewidth* introduced by Robertson and Seymour [8] but appears also under the name of *partial k-trees* in [9, 10] (see, e.g., [11]). The notion of treewidth plays a central role in algorithmic and complexity aspects on graphs.

Chordal graphs appear in the literature under different names such as *triangulated graphs* (Chapter 4 of [4]), *rigid-circuit graphs*, *perfect elimination graphs* and others. Most of the applications are due to the tree structure of chordal graphs which can be described in terms of so-called *clique trees* (arranging the maximal cliques of the graph in a tree).

The hypergraph-theoretical background of chordal graphs is given by α-acyclic hypergraphs which play an important role in the theory of relational database schemes. Various desirable properties of such schemes can be expressed in terms of various levels of acyclicity of hypergraphs [6,7]: Chordal graphs correspond to α-acyclic hypergraphs, dually chordal graphs correspond to the dual hypergraphs of α-acyclic hypergraphs, strongly chordal graphs correspond to β-acyclic hypergraphs (which are equivalent to totally balanced hypergraphs), ptolemaic graphs correspond to γ-acyclic hypergraphs, and block graphs correspond to Berge-acyclic hypergraphs. Actually, tree structure of hypergraphs was captured as *arboreal hypergraphs* by Berge [12,13]; a hypergraph is α-acyclic if and only its dual is arboreal.

We discuss also another width parameter of graphs, namely clique-width, and its relationship to treewidth as well as its algorithmic applications. Very similar to treewidth, it is known that whenever a problem is expressible in a certain kind of Monadic Second-Order Logic, and one deals with a class of graph whose clique-width is bounded by a constant then the problem is efficiently solvable on this class. This is one of the main reasons for the

great interest in treewidth and clique-width of (special) graphs. In general, it is NP-hard to determine the clique-width of a graph, and for many important graph classes, the clique-width is unbounded. For some interesting classes, however, clique-width is bounded.

Finally, we discuss some other graph parameters, namely, the tree-length and the tree-breadth of a graph, the tree-distortion and the tree-stretch of a graph, the Gromov's hyperbolicity of a graph. All these parameters try to capture and measure tree likeness of a graph from a metric point of view. The smaller such a parameter is for a graph, the closer graph is to a tree metrically. Graphs for which such parameters are bounded by small constants have many algorithmic advantages; they allow efficient approximate solutions for a number of optimization problems. Note also that recent empirical and theoretical work has suggested that many real-life complex networks and graphs arising in Internet applications, in biological and social sciences, in chemistry and physics have tree-like structures from a metric point of view.

29.2 CHORDAL GRAPHS AND VARIANTS

In this section, we collect some notions and well-known facts on chordal graphs which are described in Chapter 28 (see also the monograph [4] and the survey [3] as well as [5] for details). In order to make this section self-contained, we briefly repeat some of the basic definitions and properties. Throughout this section, let $G = (V, E)$ be a finite undirected graph which is simple (i.e., loop-free and without multiple edges).

29.2.1 Chordal Graphs

Definition 29.1 *A graph is* chordal *if it does not contain any chordless cycle with at least four vertices.*

Obviously, trees and forests are chordal since they are cycle-free for any cycle length. Chordal graphs have a nice separator property which was found by Dirac [14].

Definition 29.2

i. *The vertex set $S \subseteq V$ is a* separator (*or* cutset) *for nonadjacent vertices $a, b \in V$* (a–b-separator) *if a and b are in different connected components in $G[V \setminus S]$.*

ii. *S is a* minimal *a-b-*separator *if S is an a-b-separator and no proper subset of S is an a–b-separator.*

iii. *S is a* (minimal) separator *if there are vertices a, b such that S is a* (*minimal*) *a–b-separator.*

Theorem 29.1 [14] *A graph G is chordal if and only if every minimal separator in G induces a clique.* ■

Definition 29.3

i. *A vertex $v \in V$ is* simplicial in *G if $N(v)$ induces a clique in G.*

ii. *An ordering $(v_1, \ldots, v_n)$ of the vertices of V is a* perfect elimination ordering (p.e.o.) of *G if for all $i \in \{1, \ldots, n\}$, the vertex v_i is simplicial in the remaining subgraph $G_i := G[\{v_i, \ldots, v_n\}]$.*

Obviously, the notion of a simplicial vertex generalizes leaves in trees.

Lemma 29.1 [14] *Every chordal graph with at least one vertex contains a simplicial vertex. If G is not a clique then G contains at least two nonadjacent simplicial vertices.* ■

Corollary 29.1 [14,15] *G is chordal if and only if G has a perfect elimination ordering. Moreover, every simplicial vertex of a chordal graph G can be the first vertex of a perfect elimination ordering of G.*

For a collection $\mathcal{T}$ of subtrees of a tree T, let the *vertex intersection graph* $G_{\mathcal{T}}$ of $\mathcal{T}$ be the graph having the elements of $\mathcal{T}$ as its vertices, and two subtrees t and t' from $\mathcal{T}$ are adjacent in $G_{\mathcal{T}}$ if they share a vertex in T.

Proposition 29.1 *The vertex intersection graph of a collection of subtrees in a tree is chordal.*

Proof. Let $G = (V, E)$ be the vertex intersection graph of a collection of subtrees in a tree T. Suppose G contains a chordless cycle $(v_0, v_1, \ldots, v_{k-1}, v_0)$ with $k > 3$ corresponding to the sequence of subtrees $T_0, T_1, \ldots, T_{k-1}, T_0$ of the tree T; that is, $T_i \cap T_j \neq \emptyset$ if and only if i and j differ by at most one modulo k. All arithmetic will be done modulo k.

Choose a point a_i from $T_i \cap T_{i+1}$ $(i = 0, \ldots, k-1)$. Let b_i be the last common point on the (unique) simple paths from a_i to a_{i-1} and a_i to a_{i+1}. These paths lie in T_i and T_{i+1}, respectively, so that b_i also lies in T_i and T_{i+1}. Let P_{i+1} be the simple path connecting b_i to b_{i+1} in T. Clearly $P_i \subseteq T_i$, so $P_i \cap P_j = \emptyset$ for i and j differing by more than 1 mod k. Moreover, $P_i \cap P_{i+1} = \{b_i\}$ for $i = 0, \ldots, k-1$. Thus, $\bigcup_i P_i$ is a simple cycle in T, contradicting the definition of a tree. ■

The tree structure of chordal graphs is described in terms of so-called *clique trees* of the maximal cliques of the graph; see Theorem 29.2. Let $\mathcal{C}(G)$ denote the family of $\subseteq$-maximal cliques of G. A *clique tree* T of G has the maximal cliques of G as its nodes, and for every vertex v of G, the maximal cliques containing v form a subtree of T. This property will be generalized in the hypergraph chapter; it can be taken for defining α-acyclicity of a hypergraph (see Definition 29.17). The existence of a clique tree characterizes chordal graphs:

Theorem 29.2 [16–18] *A graph is chordal if and only if it has a clique tree.*

Proof. "$\Longleftarrow$": Assume that G has a clique tree T. If T has only one node then G is a clique and thus chordal. Now let T have $k > 1$ nodes and assume as induction hypothesis that the assertion is true for clique trees with less than k nodes. Let C be a leaf node in T, let C' be its neighbor in T, let V_C be the subset of G vertices occuring only in C, and let T' be the clique tree restricted to $V \setminus V_C$. ■

V_C must be nonempty since otherwise, $C \subset C'$ which is impossible by maximality of the cliques. Now start a p.e.o. of G with the vertices of V_C and then continue with a p.e.o. for $G - V_C$ which must exist since T' has less nodes than T.

"$\Longrightarrow$": For this direction, we use a version described by Spinrad in [19]: Assume that G is chordal and let $\sigma = (v_1, \ldots, v_n)$ be a p.e.o. of G. We construct a clique tree for the subgraph $G_i = G[v_i, \ldots, v_n]$ for all vertices, starting with $i = n$ and ending with $i = 1$. Let C_i be the clique consisting of v_i and all neighbors v_j of v_i, $j > i$. After each vertex v_i is processed, v_i is given a pointer to the clique C_i in the tree. We note that vertices may be added to this clique later in the algorithm, but v_i will always point to a clique which contains C_i.

Let v_i be the next vertex considered, and assume we know the clique tree on the graph induced by vertices $v_{i+1}, \ldots, v_n$. We need to add C_i to the clique tree. Let v_j be the first (i.e., leftmost) vertex of C_i on the right of v_i in σ. If $|C_i| = |C_j| + 1$, and the clique pointed to

by v_j is equal to C_j then we add v_i to this clique; in other words, C_i replaces C_j in the tree. Otherwise, add C_i as a new node of the tree. Connect C_i to the tree by adding an edge from C_i to the clique pointed to by v_j.

To see that the algorithm is correct, it is sufficient to look at two cases. Either C_j is a maximal clique in $G_{i+1} = G[\{v_{i+1}, \ldots, v_n\}]$ or it is not. If C_j is a maximal clique, it clearly must be replaced by C_i if C_j is contained in C_i, which occurs if $C_i = C_j \cup \{v_i\}$, and the algorithm does this correctly. If C_j is not a maximal clique in G_{i+1} or C_i does not contain C_j, then C_i cannot contain any maximal clique of G_{i+1}, and must be added as a new node. All elements of $C_i - v_i$ are in the clique pointed to by v_j, so the subtrees generated by the occurrences of all vertices remain connected. ■

A consequence of Theorem 29.2 and Proposition 29.1 is:

Corollary 29.2 [16–18] *A graph is chordal if and only if it is the intersection graph of certain subtrees of a tree.*

Since a p.e.o. of a chordal graph can be determined in linear time (see, e.g., [4,20]), the proof of Theorem 29.2 implies the following.

Theorem 29.3 *Given a chordal graph $G = (V, E)$, a clique tree of G can be constructed in linear time $\mathcal{O}(|V| + |E|)$.* ■

Interestingly, a clique tree of a chordal graph G gives also the minimal separators of G.

Lemma 29.2 [21,22] *Let $G = (V_G, E_G)$ be a chordal graph with clique tree $T = (\mathcal{C}(\mathcal{G}), E_T)$. Then $S \subseteq V_G$ is a minimal separator in G if and only if there are maximal cliques Q_i, Q_j of G with $Q_iQ_j \in E_T$ such that $S = Q_i \cap Q_j$.* ■

The specific structure of chordal graphs allows to solve various problems efficiently which is well described in [4]; as another example we give here a linear-time algorithm by Andras Frank [23] for maximum weight independent set (MWIS) on chordal graphs.

Let $G = (V, E)$ be a chordal graph with perfect elimination ordering $(v_1, \ldots, v_n)$ of G and $\omega : V \longrightarrow R^+$ a nonnegative weight function on V. The algorithm of Frank efficiently constructs a maximum weight stable set $\mathcal{I}$ of G in the following way:

(0) $\mathcal{I} := \emptyset$; all vertices in V are *unmarked*

(1) **for** $i := 1$ **to** n **do**

  if $\omega(v_i) > 0$ **then** mark v_i and let $\omega(u) := \max(\omega(u) - \omega(v_i), 0)$ for all vertices $u \in N_i(v_i)$.

(2) **for** $i := n$ **downto** 1 **do**

  if v_i is marked **then** let $\mathcal{I} := \mathcal{I} \cup \{v_i\}$ and unmark all vertices $u \in N(v_i)$.

Theorem 29.4 [23] *The algorithm described above is correct and runs in linear time.* ■

It is clear that the algorithm runs in linear time. For the correctness, we need the following (inductive) argument: As in the algorithm, let $(v_1, \ldots, v_n)$ be a p.e.o. of G and ω a weight function on V. Now let ω' be the weight function resulting from step (1) of the algorithm for the simplicial vertex v_1. We claim the following proposition.

Proposition 29.2 $\alpha_\omega(G) = \alpha_{\omega'}(G - v_1) + \omega(v_1)$.

This is clear by the following argument: If v_1 is in a maximum weight stable set S in G then none of its neighbors are in S, and the claim holds. Otherwise, if $v_1 \notin S$ then exactly one of its neighbors, say v_i, $i > 1$, is in S (otherwise S would not be a maximal stable set), and now $\omega'(v_i) = \omega(v_i) - \omega(v_1)$ holds.

29.2.2 Some Subclasses of Chordal Graphs

As mentioned in Chapter 28, interval graphs are a very important subclass of chordal graphs. Here is another subclass of chordal graphs which plays an important role in various contexts:

Definition 29.4 *A graph is a* split graph *if its vertex set can be partitioned into a clique and a stable set. Such a partition is called a* split partition.

It is easy to see that the complement of a split graph is a split graph as well, and split graphs are chordal. In what follows, we say a vertex x *sees* a vertex y if x is adjacent to y; otherwise we say x *misses* y.

Theorem 29.5 [24] *The following conditions are equivalent:*

i. *G is a split graph.*

ii. *G and $\overline{G}$ are chordal.*

iii. *G contains no induced $2K_2 = \overline{C_4}$, C_4, C_5 (i.e., G is $(2K_2, C_4, C_5)$-free).*

Proof. "(ii) $\Longleftrightarrow$ (iii)": If G and $\overline{G}$ are chordal then obviously G contains no induced $2K_2$, C_4 and C_5. In the other direction, note that for every $k \geq 6$, C_k contains a $2K_2$, and $\overline{C_5} = C_5$. Thus, if G contains no induced $2K_2$, C_4 and C_5 then G and $\overline{G}$ are chordal.

"(i) $\Longrightarrow$ (ii)": If the vertex set V of G has a partition into a clique Q and a stable set S then obviously, every vertex in S is simplicial in G. Thus, a p.e.o. of G can start with all vertices of S and finish with all vertices of Q. Similar arguments hold for $\overline{G}$, and thus G and $\overline{G}$ are chordal.

"(i) $\Longleftarrow$ (ii)": Suppose that G and $\overline{G}$ are chordal (or, equivalently, G contains no induced $2K_2$, C_4, and C_5).

If there is a vertex $v \in V$ which is simplicial in G and $\overline{G}$ then $N[v]$ is a clique and $\overline{N[v]}$ is a stable set giving the desired split partition.

If there is a vertex $v \in V$ which is neither simplicial in G nor simplicial in $\overline{G}$ then let $a, b \in N(v)$ be vertices with $ab \notin E$ and let $c, d \in \overline{N[v]}$ with $cd \in E$. Since G is $2K_2$-free, a sees c or d, and similarly, b sees c or d but since G is C_4-free, a and b do not have a common neighbor in c, d. Thus, say, a sees c but not d and vice versa for b but now v, a, b, c, d induce a C_5 in G which is a contradiction.

Thus, every vertex $v \in V$ is either simplicial in G or simplicial in $\overline{G}$. Let $V_1 := \{v \in V \mid v$ is simplicial in $G\}$ and $V_2 := \{v \in V \mid v$ is simplicial in $\overline{G}\}$. Note that $V = V_1 \cup V_2$ is a partition of V. Now, if V_1 is a stable set and V_2 is a clique then this gives the desired split partition. Suppose to the contrary that V_1 contains an edge $xy \in E$. Then since G is $2K_2$-free, the set of nonneighbors of x and y form a stable set, and since x and y are simplicial, the set of neighbors of x and y form a clique which gives the desired split partition. ■

Theorem 29.5 does not immediately give a linear-time recognition of split graphs. The following nice characterization of split graphs in terms of their degree sequence leads to linear-time recognition of split graphs:

Theorem 29.6 [25,26] *Let G have the degree sequence $d_1 \geq d_2 \geq \ldots \geq d_n$ and $\omega := \max\{i \mid d_i \geq i - 1\}$. Then G is a split graph if and only if $\Sigma_{i=1}^{\omega} d_i = \omega(\omega - 1) + \Sigma_{i=\omega+1}^{n} d_i$.* ■

See [25,26] for more details.

Finally, another interesting subclass of chordal graphs should be mentioned which will be discussed in more detail in the section on strongly chordal graphs and on β-acyclicity. Assume that G is a chordal graph. A chord x_ix_j in a cycle $C = (x_1, x_2, \ldots, x_{2k}, x_1)$ of even length $2k$ is an *odd chord* if the distance in C between x_i and x_j is odd.

Farber [27] defined strongly chordal graphs in terms of strong elimination orderings rather than odd chords in even cycles (see Definition 29.33), but he showed that chordal graphs having odd chords in even cycles are exactly the strongly chordal graphs (see Theorem 29.34).

Chordal graphs can be generalized in a natural way by placing a variety of restrictions on the number and type of chords with respect to a cycle. A fairly general scheme is given in the following definition (which was motivated by relational database schemes).

Definition 29.5 [28] *For $k \geq 4$ and $\ell \geq 1$, a graph G is* (k, ℓ)-chordal *if each cycle in G of length at least k contains at least ℓ chords.*

Thus chordal graphs are the (4,1)-chordal graphs. Further conditions can be placed on the parity of the cycles (chords in odd cycles), the parity of the cycle distance of the end vertices of chords (odd chords), requiring crossing and/or parallel chords, requiring all these conditions for G and $\overline{G}$, and requiring these conditions in bipartite graphs (where all cycles are of even length). Thus, for example, the (5,2)-odd-crossing-chordal graphs are the graphs such that every odd cycle of length at least five has at least two crossing chords.

See [3] for more details and Theorem 29.45 for a characterization of (5,2)-chordal graphs.

29.3 α-ACYCLIC HYPERGRAPHS AND THEIR DUALS

29.3.1 Motivation from Relational Database Theory

Fagin [7] gives a very nice introduction into acyclic database schemes (of various degrees, namely α-, β-, and γ-acyclicity) and their equivalence to desirable properties of relational databases. Since Fagin's introduction is mostly informal and we need some definitions, we follow the presentation in papers such as [29] for this subsection.

A (relational) *database scheme* as introduced by Codd [30] can be thought of as a collection of table skeletons, or, alternatively, as a set of subsets of *attributes*, or column names in the tables. These attribute subsets form the hyperedges of a finite hypergraph. A *relational database* corresponds to a family of relations over the attributes.

Let $V = \{v_1, \ldots, v_n\}$ be a finite set of distinct symbols called *attributes* or *column names* (*name*, *first name*, *age*, *birthday*, *citizenship*, *married*, *home address*, *telephone number*, etc).

Let $Y \subseteq V$. A *Y-tuple* is a mapping that associates a value (from a certain universe U) with each attribute in Y. For instance, if $Y = \{name, age, citizenship, married\}$ then a Y-tuple is a 4-tuple such as $(Higgins, 48, Canada, no)$.

If $X \subseteq Y$ and t is a Y-tuple, then *the projection* $t[X]$ denotes the X-tuple obtained by restricting t to X. For instance, if $X = \{name, citizenship\}$ and $t = (Higgins, 48, Canada, no)$ then $t[X] = (Higgins, Canada)$.

A *Y-relation* is a finite set of Y-tuples. If r is a Y-relation and $X \subseteq Y$ then by the *projection* $r[X]$ of r onto X, we mean the set of all tuples $t[X]$, where $t \in r$.

If V is a set of attributes, then we define a *relational database scheme* (*database scheme* for short) $\mathcal{E} = \{E_1, \ldots, E_m\}$ to be a set of subsets of V, that is, $(V, \mathcal{E})$ is a hypergraph over vertex set V.

Intuitively, for each i, the set E_i of attributes is considered to be the set of column names for a relation; the E_i's are called *relation schemes*. If $r_1, \ldots, r_m$ are relations, where r_i is a relation over E_i, $i \in \{1, \ldots, m\}$, then we call $\{r_1, \ldots, r_m\}$ a *database over* $\mathcal{E}$.

The *join* $r_1 \bowtie r_2$ of two relations r_1 and r_2 with attribute sets E_1 and E_2, respectively, is the set of all tuples t with attribute set $E_1 \cup E_2$ for which the projection $t[E_i]$ is in r_i, $i = 1, 2$.

Example 29.1

r_1 :	A	B
	0	0
	1	1

r_2 :	B	C
	0	0
	1	1

r_3 :	A	C
	0	1
	1	0

$r_1 \bowtie r_2$:	A	B	C
	0	0	0
	1	1	1

$r_1 \bowtie r_3$:	A	B	C
	0	0	1
	1	1	0

$r_2 \bowtie r_3$:	A	B	C
	1	0	0
	0	1	1

More generally, the *join* $r_1 \bowtie \ldots \bowtie r_m$ of the relations $r_1, \ldots, r_m$, $m \geq 2$, with attribute sets $E_1, \ldots, E_m$, respectively, is the set of all tuples t with attribute set $E_1 \cup \ldots \cup E_m$, such that for each $i \in \{1, \ldots, m\}$, the projection $t[E_i]$ of tuple t onto attributes E_i fulfills $t[E_i] \in r_i$.

The join of all three relations in Example 29.1 is empty: $r_1 \bowtie r_2 \bowtie r_3 = \emptyset$.

We say that a relation r with attributes $E_1 \cup \ldots \cup E_m$ obeys the *join dependency* $\bowtie \{E_1, \ldots, E_m\}$ if $r = r_1 \bowtie \ldots \bowtie r_m$, where $r_i = r[E_i]$ for each $i \in \{1, \ldots, m\}$.

A highly desirable property of a relational database $r_1, \ldots, r_m$, $m \geq 2$, is that the entries in it are conflict-free. In general, the attribute sets are not pairwise disjoint, and it easily might happen that an entry in one of the relations is updated while the same entry in another relation is not. Pairwise consistency captures conflict-freeness for every two of the relations, and global consistency, roughly saying, means that all of them together are conflict-free. If the relations are globally consistent then they are pairwise consistent but not vice versa as Example 29.1 shows; surprisingly, it turns out that the equivalence of pairwise and of global consistency corresponds to a hypergraph acyclicity property of the underlying attribute sets.

More formally, let r and s be relations with attributes R and S, respectively, and let $Q = R \cap S$, that is, Q is precisely the set of attributes that r and s have in common. We say that r *and* s *are consistent* if $r[Q] = s[Q]$, that is, the projections of r and s onto their common attributes are the same.

Example 29.2

r_1 :	A	B	C
	0	1	2
	1	2	3
	2	3	4

r_2 :	A	D	E
	0	3	4
	0	5	6
	3	4	5

$r_1 \bowtie r_2$:	A	B	C	D	E
	0	1	2	3	4
	0	1	2	5	6

In Example 29.2, r_1 and r_2 have only A as common attribute, and the projection $r_1[A]$ is $\{0, 1, 2\}$ while the projection $r_2[A]$ is $\{0, 3\}$; thus, r_1 and r_2 are not consistent.

In Example 29.1, each pair r_i, r_j of relations, $i, j \in \{1, 2, 3\}$, is consistent.

Definition 29.6 *Let* $\{r_1, \ldots, r_m\}$ *be an arbitrary database over* $\mathcal{E} = \{E_1, \ldots, E_m\}$.

i. $\{r_1, \ldots, r_m\}$ *is* pairwise consistent *if for all* $i, j \in \{1, \ldots, m\}$, r_i *and* r_j *are consistent.*

ii. $\{r_1, \ldots, r_m\}$ *is* globally consistent *if there is a relation* r *over the attribute set* $E_1 \cup \ldots \cup E_m$ *such that for each* $i \in \{1, \ldots, m\}$, $r_i = r[E_i]$. *Then* r *is called* universal *for* $\{r_1, \ldots, r_m\}$.

Thus, $\{r_1, \ldots, r_m\}$ is globally consistent if and only if there is a (universal) relation r such that each r_i is the projection of r onto the corresponding attribute set of r_i. Such a universal relation need not be unique, but it is known that if there is such a universal relation r, then also $r_1 \bowtie \ldots \bowtie r_m$ is such a universal relation:

Lemma 29.3 *If r is a universal relation for $r_1, \ldots, r_m$ with attribute sets $E_1, \ldots, E_m$ then $r \subseteq r[E_1] \bowtie \ldots \bowtie r[E_m] = r_1 \bowtie \ldots \bowtie r_m$.*

It is clear that if $\{r_1, \ldots, r_m\}$ is globally consistent then it is pairwise consistent but in general, the converse is false as the relations r_1, r_2, r_3 in Example 29.1 show which are pairwise consistent but not globally consistent.

Honeyman et al. [31] have shown the following theorem.

Theorem 29.7 [31] *The global consistency of a relational database is an NP-complete problem.* ■

In [29], it is shown that for a relational database scheme, pairwise consistency implies global consistency if and only if it is α-acyclic (see Theorem 29.17).

29.3.2 Some Basic Hypergraph Notions

A pair $H = (V, \mathcal{E})$ is a (finite) *hypergraph* if V is a finite vertex set and $\mathcal{E}$ is a collection of subsets of V (the *edges* or *hyperedges* of H). Hypergraphs are a natural generalization of undirected graphs; unlike edges, hyperedges are not necessarily two-elementary. In many cases, hyperedges containing exactly one vertex (so-called *loops*) are excluded. Equivalently, a hypergraph $H = (V, \mathcal{E})$ with $V = \{v_1, \ldots, v_n\}$ and $\mathcal{E} = \{e_1, \ldots, e_m\}$ can be described by its $n \times m$ vertex-hyperedge incidence matrix $M(H)$ with entries $m_{ij} \in \{0, 1\}$ and $m_{ij} = 1 \iff v_i \in e_j$ for $i \in \{1, \ldots, n\}$ and $j \in \{1, \ldots, m\}$.

Subsequently, we collect some basic notions and properties—see, for example, [13].

Definition 29.7 *A hypergraph $H = (V, \mathcal{E})$ is* simple *if it has no repeated edges. Moreover, if no hyperedge $e \in \mathcal{E}$ is properly contained in another hyperedge $e' \in \mathcal{E}$ then H is called a* Sperner family *or* clutter.

In the database community (see, e.g., [29]), clutters are called *reduced hypergraphs*.

Definition 29.8 *Let $H = (V, \mathcal{E})$ be a finite hypergraph.*

i. *The* subhypergraph *induced by the subset $A \subseteq V$ is the hypergraph $H[A] = (A, \mathcal{E}_A)$ with edge set $\mathcal{E}_A = \{e \cap A \mid e \in \mathcal{E}\}$.*

ii. *The* partial hypergraph *given by the edge subset $\mathcal{E}' \subseteq \mathcal{E}$ is the hypergraph with the vertex set $\bigcup \mathcal{E}'$ and the edge set $\mathcal{E}'$.*

Note that both restrictions $A \subset V$ and $\mathcal{E}' \subset \mathcal{E}$ can be combined in a subhypergraph $H'[A] = (A, \mathcal{E}'_A)$ with edge set $\mathcal{E}'_A = \{e \cap A \mid e \in \mathcal{E}' \subset \mathcal{E}\}$ called *partial subhypergraph* in [13].

The partial hypergraphs [13] are called *subhypergraphs* in [6]. Since this may cause confusion, we also use the name *edge-subhypergraphs* for partial hypergraphs and *vertex-subhypergraphs* in case (i).

Dualization is a classical concept which is well-known from geometry; there, points and hyperplanes exchange their role. Here, dualization means that vertices and hyperedges exchange their role:

Definition 29.9 *Let $H = (V, \mathcal{E})$ be a finite hypergraph. For $v \in V$, let $\mathcal{E}_v = \{e \in \mathcal{E} \mid v \in e\}$. The* dual hypergraph *$H^* = (\mathcal{E}, \mathcal{E}^*)$ of H has vertex set $\mathcal{E}$ and hyperedge set $\{\mathcal{E}_v \mid v \in V\}$.*

If the hypergraph H is given in terms of its incidence matrix $M(H)$ then the incidence matrix of the dual of H is the transposal of $M(H)$: $M(H^*) = (M(H))^T$.

Evidently, the dual of the dual of H is isomorphic to H itself since the twofold transposal of a matrix is the matrix itself:

Proposition 29.3 $(H^*)^* \sim H$.

Graphs and hypergraphs are closely related to each other. The next definition represents two examples.

Definition 29.10 *Let $H = (V, \mathcal{E})$ be a finite hypergraph.*

i. *The* 2-section graph *$2SEC(H)$ of H has the vertex set V, and two vertices u, v are adjacent if u and v are contained in a common hyperedge: $\exists e \in \mathcal{E}$ such that $u, v \in e$.*

ii. *The* line graph *$L(H) = (\mathcal{E}, F)$ is the intersection graph of $\mathcal{E}$, that is, for any $e, e' \in \mathcal{E}$ with $e \neq e'$, $ee' \in F \Longleftrightarrow e \cap e' \neq \emptyset$.*

The 2-section graph of H is denoted by $[H]_2$ in [13]; the line graph is also called *representative graph* in [13]. Again, these notions have different names in different communities; the 2-section graph is also called *adjacency graph* in [32], *primal graph* [33], or *Gaifman graph* [34] and has no name but is denoted by $G(H)$ in [29]. The line graph is also called *dual graph* in [35].

The following isomorphism is easy to see.

Proposition 29.4 $2SEC(H) \sim L(H^*)$.

A subfamily $\mathcal{E}' \subseteq \mathcal{E}$ is called *pairwise intersecting* if for all $e, e' \in \mathcal{E}'$, $e \cap e' \neq \emptyset$.

Definition 29.11 *Let $H = (V, \mathcal{E})$ be a hypergraph.*

i. *H is* conformal *if every clique C in $2SEC(H)$ is contained in a hyperedge $e \in \mathcal{E}$.*

ii. *H has the* Helly property *if every pairwise intersecting subfamily $\mathcal{E}' \subseteq \mathcal{E}$ has nonempty total intersection: $\bigcap \mathcal{E}' \neq \emptyset$.*

The following is easy to see.

Proposition 29.5 *H has the Helly property if and only if H^* is conformal.*

The next theorem gives a polynomial time criterion for testing the Helly property of a hypergraph. It is closely related to an earlier criterion for conformality given by Gilmore which will be mentioned in Theorem 29.9.

For a hypergraph $H = (V, \mathcal{E})$ and for any 3-elementary set $A = \{a_1, a_2, a_3\} \subseteq V$, let $\mathcal{E}_A$ denote the set of all hyperedges $e \in \mathcal{E}$ such that $|e \cap A| \geq 2$.

Theorem 29.8 [13,36] *A hypergraph $H = (V, \mathcal{E})$ has the Helly property if and only if for all 3-elementary sets $A = \{a_1, a_2, a_3\} \subseteq V$, the total intersection of all hyperedges containing at least two vertices of A is nonempty: $\bigcap \mathcal{E}_A \neq \emptyset$.*

Proof. "$\Longrightarrow$": Let H be a hypergraph with the Helly property, and let $\{e_1, \ldots, e_k\} \subseteq \mathcal{E}$ be the hyperedges for which $|e_i \cap A| \geq 2$, $i \in \{1, \ldots, k\}$. Then for all $i \neq j$, $i, j \in \{1, \ldots, k\}$, $e_i \cap e_j$ is nonempty and thus, their total intersection is nonempty since H has the Helly property.

"$\Longleftarrow$": Now assume that $\{e_1, \ldots, e_\ell\} \subseteq \mathcal{E}$ is a collection of pairwise intersecting hyperedges. If $\ell = 2$ then obviously their total intersection is nonempty; thus let $\ell > 2$. We assume inductively that the assertion of nonempty total intersection is true for less than ℓ hyperedges with pairwise nonempty intersection.

Then by the induction hypothesis, $e_1 \cap \ldots \cap e_{\ell-1} \neq \emptyset$; let $a_1 \in e_1 \cap \ldots \cap e_{\ell-1}$. Moreover, $e_2 \cap \ldots \cap e_\ell \neq \emptyset$; let $a_2 \in e_2 \cap \ldots \cap e_\ell$. Finally $e_1 \cap e_\ell \neq \emptyset$; let $a_3 \in e_1 \cap e_\ell$.

Let $A := \{a_1, a_2, a_3\}$. It is easy to see that in the case $|A| < 3$ we are done. Now let $|A| = 3$. Thus every e_i, $i = 1, \ldots, \ell$, contains at least two elements from the 3-elementary set A, and by the assumption, their total intersection is nonempty. ■

An obvious consequence of Theorem 29.8 is as follows:

Corollary 29.3 *Testing the Helly property for a given hypergraph can be done in polynomial time.*

Corollary 29.4 *Every collection of subtrees of a tree has the Helly property.*

Proof. Let T be a tree with at least three vertices (otherwise the assertion is obviously fulfilled), and let a, b, c be any three vertices in T. We consider the set of all subtrees of T containing at least two of the vertices a, b, c. Let $P(x, y)$ denote the uniquely determined path in the tree T between x and y. Let x_0 denote the last vertex in $P(a, b) \cap P(b, c)$ (this intersection contains at least vertex b). Then $P(a, c)$ consists of $P(a, x_0)$ followed by $P(x_0, c)$. Thus the three paths $P(a, b)$, $P(b, c)$ and $P(a, c)$ have vertex x_0 in common, that is, x_0 is contained in every subtree of T which contains at least two of the vertices a, b, c. Thus, by Theorem 29.8, every system of subtrees has the Helly property. ■

A nice inductive proof of Corollary 29.4 is given in a script by Alexander Schrijver: The induction is on $|V(T)|$. If $|V(T)| = 1$ then the assertion is trivial. Now assume $|V(T)| \geq 2$, and let $\mathcal{S}$ be a collection of pairwise intersecting subtrees of T. Let t be a leaf of T. If there exists a subtree of T consisting only of t, the assertion is trivial. Hence we may assume that each subtree in $\mathcal{S}$ containing t also contains the neighbor of t in T. So, after deleting t from T and from all subtrees in $\mathcal{S}$, this collection is still pairwise intersecting, and the assertion follows by induction.

Actually, Theorem 29.8 is formulated in a more general way in [13]; there are various interesting generalizations of the Helly property.

According to Proposition 29.5, Theorem 29.8 can be dualized as follows.

Theorem 29.9 (Gilmore, see [13]) *Let $H = (V, \mathcal{E})$ be a hypergraph. H is conformal if and only if for all 3-elementary edge sets $A = \{e_1, e_2, e_3\} \subseteq \mathcal{E}$ of hyperedges, there is a hyperedge $e \in \mathcal{E}$ with $(e_1 \cap e_2) \cup (e_1 \cap e_3) \cup (e_2 \cap e_3) \subseteq e$.*

Proof. "$\Longrightarrow$": Obviously, $(e_1 \cap e_2) \cup (e_1 \cap e_3) \cup (e_2 \cap e_3)$ is a clique in the 2-section graph $2SEC(H)$ of H. By conformality, there is a hyperedge e with $(e_1 \cap e_2) \cup (e_1 \cap e_3) \cup (e_2 \cap e_3) \subseteq e$.

"$\Longleftarrow$": Let $A = \{e_1, e_2, e_3\} \subseteq \mathcal{E}$ and let $\mathcal{E}_u$ be a hyperedge in H^* containing at least two of e_1, e_2, e_3. Then $u \in (e_1 \cap e_2) \cup (e_1 \cap e_3) \cup (e_2 \cap e_3)$ and thus also $u \in e$. Thus, e is in the total intersection of all hyperedges $\mathcal{E}_u$ which contain at least two of e_1, e_2, e_3. Then by Theorem 29.8, H^* has the Helly property and thus, by Proposition 29.5, H is conformal. ■

There is a third type of graphs derived from a hypergraph $H = (V, \mathcal{E})$, namely the bipartite vertex-edge incidence graph $\mathcal{I}(H)$ (which is a reformulation of the incidence matrix of H in terms of a bipartite graph). The two color classes of $\mathcal{I}(H)$ are the sets V and $\mathcal{E}$, respectively, and a vertex v and an edge e are adjacent if and only if $v \in e$. More formally:

Definition 29.12 *Let $H = (V, \mathcal{E})$ be a finite hypergraph. In the* bipartite incidence graph *$\mathcal{I}(H) = (V, \mathcal{E}, I)$ of H, $v \in V$ and $e \in \mathcal{E}$ are adjacent if and only if $v \in e$.*

In the other direction, namely from graphs to hypergraphs, the most basic constructions are the following:

Definition 29.13 *Let $G = (V, E)$ be a graph.*

i. *The clique hypergraph $\mathcal{C}(G)$ consists of the $\subseteq$-maximal cliques of G.*

ii. *The neighborhood hypergraph $\mathcal{N}(G)$ consists of the closed neighborhoods $N[v]$ of all vertices v in G.*

iii. *The disk hypergraph $\mathcal{D}(G)$ consists of the iterated closed neighborhoods $N^i[v]$, $i \geq 1$, of all vertices v in G, where $N^1[v] := N[v]$ and $N^{i+1}[v] := N[N^i[v]]$.*

Note that in general, the neighborhood hypergraph $\mathcal{N}(G)$ is not simple since different vertices can have the same closed neighborhood in G. The following is easy to see.

Proposition 29.6 *$\mathcal{N}(G)$ is self-dual, that is, $(\mathcal{N}(G))^* \sim \mathcal{N}(G)$.*

Moreover, the 2-section graph of $\mathcal{C}(G)$ is isomorphic to G and thus, $\mathcal{C}(G)$ is conformal. Note that a hypergraph uniquely determines its 2-section graph but not vice versa.

Lemma 29.4 *Every conformal Sperner hypergraph $H = (V, \mathcal{E})$ is the clique hypergraph of its 2-section graph $2SEC(H)$: $H = \mathcal{C}(2SEC(H))$.*

Proof. Let H be conformal and Sperner. We show:

1. For every $e \in \mathcal{E}$, e is a maximal clique in $2SEC(H)$:

   Obviously, e is a clique in $2SEC(H)$ and thus, there is a maximal clique C' in $2SEC(H)$ with $e \subseteq C'$. Since H is conformal, there is an $e' \in \mathcal{E}$ with $C' \subseteq e'$, that is, $e \subseteq C' \subseteq e'$ and since H is Sperner, $e = C' = e'$ follows.

2. For every maximal clique C in $2SEC(H)$, $C \in \mathcal{E}$ holds:

   By conformality of H, there is $e \in \mathcal{E}$ with $C \subseteq e$, and since e is a clique in $2SEC(H)$, there is a maximal clique C' in $2SEC(H)$ with $e \subseteq C'$, that is, $C \subseteq e \subseteq C'$. By maximality of C, $C = e = C'$ follows. ■

For a graph $G = (V, E)$, let $G^2 = (V, E^2)$ with $xy \in E^2$ for $x \neq y$ if and only if $d_G(x, y) \leq 2$, that is, either $xy \in E$ or there is a common neighbor z of x and y.

The following is easy to see.

Proposition 29.7 *$G^2 \sim L(\mathcal{N}(G))$.*

For graph $G = (V, E)$, let $B(G) = (V', V'', F)$ denote the bipartite graph with two disjoint copies V' and V'' of V, and for $v' \in V'$ and $w'' \in V''$, $v'w'' \in F$ if and only if either $v = w$ or $vw \in E$.

The following is easy to see.

Proposition 29.8 $B(G) \sim \mathcal{I}(\mathcal{N}(G))$.

The line graph of $\mathcal{C}(G)$ is the classical *clique graph operator* in graph theory.

Definition 29.14 *Let G be a graph.*

i. *The* clique graph $K(G)$ *of G is defined as $K(G) = L(\mathcal{C}(G))$.*

ii. *G is a* clique graph *if there is a graph G' such that G is the clique graph of G', that is, $G = K(G')$.*

Theorem 29.10 [37] *A graph G is a clique graph if and only if some class of complete subgraphs of G covers all edges of G and has the Helly property.* ■

See [3,5] and in particular the survey [38] by Szwarcfiter for more details on clique graphs. Recognizing whether a graph is a clique graph is NP-complete [39].

29.3.3 Hypergraph 2-Coloring

A hypergraph $H = (V, \mathcal{E})$ is 2-*colorable* if its vertex set V has a partition $V = V_1 \cup V_2$ such that every hyperedge $e \in \mathcal{E}$ has at least one vertex from each of the sets V_1 and V_2. See [13] for the more general notion of hypergraph coloring. The *Hypergraph 2-Coloring Problem* (also called *Bicoloring Problem*, *Set Splitting Problem* [SP4] in [40]) is the question whether a given hypergraph is 2-colorable.

Lovász [41] has shown that the Hypergraph 2-Coloring Problem is NP-complete even for hypergraphs whose hyperedges have size at most 3 (see [40]); the original reduction in [41] is from the graph coloring problem (which has been shown to be NP-complete in [42]) to hypergraph 2-coloring.

The following nice reduction from the satisfiability problem SAT to the hypergraph 2-coloring problem was given in [43].

Let $F = C_1 \wedge \ldots \wedge C_m$ be a Boolean expression in conjunctive normal form (CNF for short) with clauses $C_1, \ldots, C_m$ and variables $x_1, \ldots, x_n$. Each clause consists of a disjunction of literals, that is, unnegated or negated variables.

Let $H_F = (V_F, \mathcal{E}_F)$ be the following hypergraph for F:

The vertex set $V_F = \{x_1, \ldots, x_n\} \cup \{\neg x_1, \ldots, \neg x_n\} \cup \{f\}$ where f is a new symbol different from the variable symbols.

The edge set $\mathcal{E}_F$ of H_F consists of the following edges:

i. For all $i \in \{1, \ldots, n\}$, let $X_i = \{x_i, \neg x_i\}$,

ii. For all $j \in \{1, \ldots, m\}$, let Y_j be the set of all literals in C_j plus, additionally, the element f.

We show that F is satisfiable if and only if H_F is 2-colorable:

Given a truth assignment which satisfies F, we associate with it the following 2-coloring $V_1 \cup V_2$. If x_i has truth value 1 then $x_i \in V_1$ and $\neg x_i \in V_2$ and vice versa if x_i has truth value 0. The element f belongs to V_2. Now, for each $i \in \{1, \ldots, n\}$, the edge $\{x_i, \neg x_i\}$ intersects both V_1 and V_2. An edge Y_i intersects V_2 on f and intersects V_1 since it has a true literal.

On the other hand, given a 2-coloring $V_1 \cup V_2$ of H_F, with, say $f \in V_2$ we assign true to each x_i in V_1 and false to those in V_2. This gives a truth assignment since the edges $\{x_i, \neg x_i\}$ meet both V_1 and V_2. The edge Y_j of every clause C_j meets V_1 on an element other than f which ensures that every clause is satisfied. This shows the following theorems.

Theorem 29.11 [41] *The* 2*-coloring problem for hypergraphs is NP-complete.* ■

Based on Theorem 29.11, in [41], Lovász has shown the following theorem.

Theorem 29.12 [41] *The* 3*-coloring problem for graphs is NP-complete.* ■

See [44] for another proof of the NP-completeness of the 3-coloring problem.

29.3.4 Kőnig Property

The following definition generalizes the fundamental notions of matching and vertex cover in graphs to the corresponding notions in hypergraphs.

Definition 29.15 *Let $H = (V, \mathcal{E})$ be a hypergraph.*

i. *An edge set $\mathcal{E}' \subseteq \mathcal{E}$ is called* matching *if the edges of $\mathcal{E}'$ are pairwise disjoint. The* matching number *$\nu(H)$ is the maximum number of pairwise disjoint hyperedges of H. This parameter $\nu(H)$ is also frequently called* packing number *of H.*

ii. *A* transversal of *$\mathcal{E}$ is a subset $U \subseteq V$ such that U contains at least one vertex of every $e \in \mathcal{E}$. The* transversal number *$\tau(H)$ is the minimum number of vertices in a transversal of H.*

iii. *H has the* Kőnig Property *if $\nu(H) = \tau(H)$.*

Note that for every hypergraph, $\nu(H) \leq \tau(H)$ holds. A well-known theorem of Kőnig states that for bipartite graphs G, $\nu(G) = \tau(G)$ holds. This justifies the name *Kőnig property* and is closely related to the celebrated max-flow min-cut theorem by Ford and Fulkerson.

29.3.5 α-Acyclic Hypergraphs and Tree Structure

Unlike the case of graphs, there is a bewildering diversity of cycle notions in hypergraphs, and some of them play an important role in connection with desirable properties of relational database schemes [6,7,29,32,45]. Thus, for example, the desirable property of a relational database scheme that pairwise consistency should imply global consistency turns out to be equivalent to α-acyclicity of the scheme [6,29]; as shown in Theorems 29.16 and 29.17, a relational database scheme has this property if and only if it is α-acyclic. Moreover, α-acyclicity is equivalent to many other desirable properties of such schemes. The most important property of an α-acyclic hypergraph for applications in databases and other fields seems to be the existence of a join tree for α-acyclic hypergraphs:

Definition 29.16 *Let $H = (V, \mathcal{E})$ be a hypergraph.*

i. *Tree T is a* join tree *of H if the node set of T is the set of hyperedges $\mathcal{E}$ and for every vertex $v \in V$, the set $\mathcal{E}_v$ of hyperedges containing v forms a subtree in T.*

ii. *H is* α-acyclic *if H has a join tree.*

Note that in this way, α-acyclicity of a hypergraph is defined without referring to any cycle notion in hypergraphs.

Tarjan and Yannakakis [20] gave a linear-time algorithm for testing α-acyclicity of a given hypergraph.

Tree structure in hypergraphs has been captured in the hypergraph community as *arboreal hypergraphs* [13] (as well as its dual version, the *co-arboreal hypergraphs*) and *tree-hypergraphs* in [46]. We call arboreal hypergraphs *hypertrees*.

Definition 29.17 *A hypergraph $H = (V, \mathcal{E})$ is a* hypertree *if there is a tree T whose set of nodes is V and such that every hyperedge $e \in \mathcal{E}$ induces a subtree in T.*

Note that in [33], Gottlob et al. define the notion of hypertrees in a completely different way.

The following properties are easy to see:

Proposition 29.9 *Let $H = (V, \mathcal{E})$ be a hypergraph.*

i. *H is a hypertree if and only if its dual H^* is α-acyclic.*

ii. *If H is a hypertree then every edge-subhypergraph of H is a hypertree as well but not necessarily every vertex-subhypergraph of H.*

iii. *If H is α-acyclic then every vertex-subhypergraph of H is α-acyclic as well but not necessarily every edge-subhypergraph of H.*

The fact that α-acyclic hypergraphs may contain hyperedge cycles of a certain kind (there are various cycle definitions in hypergraphs), and the fact that edge-subhypergraphs of α-acyclic hypergraphs are not necessarily α-acyclic are somewhat counterintuitive in comparison with cycles in graphs and led Goodman and Shmueli [32] to the name *tree schema* for α-acyclic hypergraphs (see also [47] for a discussion).

The following theorem gives an important characterization of hypertrees (α-acyclic hypergraphs, respectively).

Theorem 29.13 [48–50] *A hypergraph H is a hypertree if and only if H has the Helly property and its line graph $L(H)$ is chordal.*

Proof. "$\Longrightarrow$": Let $H = (V, \mathcal{E})$ be a hypertree and let T be a tree with vertex set V such that for all $e \in \mathcal{E}$, $T[e]$ induces a subtree in T. By Corollary 29.4, every hypertree H has the Helly property. By Proposition 29.1, $L(H)$ is chordal.

"$\Longleftarrow$": A dual variant of the assertion is the following: If H is conformal and $2SEC(H)$ is chordal then H is α-acyclic. Without loss of generality we may assume that no hyperedge of H is contained in another one. By Lemma 29.4, H is the clique hypergraph of its 2-section graph, and by Theorem 29.2, the chordal graph $2SEC(H)$ has a clique tree. Thus, H is α-acyclic. ■

By Propositions 29.4 and 29.5, Theorem 29.13 can also be formulated in the following equivalent way.

Corollary 29.5 *H is α-acyclic if and only if H is conformal and $2SEC(H)$ is chordal.*

See Definition 29.15 for the Kőnig property. As a consequence of Theorem 29.13, we obtain.

Corollary 29.6 *Hypertrees have the Kőnig property.*

Proof. Let $H = (V, \mathcal{E})$ be a hypertree. Then by Theorem 29.13, H has the Helly property and there is a p.e.o. $(e_1, \ldots, e_m)$ of the edge set $\mathcal{E}$ of $L(H)$. Since e_1 is simplicial in $L(H)$, the set $\mathcal{E}_1$ of hyperedges intersecting e_1 is pairwise intersecting. By the Helly property, there is a vertex v in the intersection of $\mathcal{E}_1$. Now assume inductively that the hypergraph $H' = (V, \mathcal{E} \setminus \mathcal{E}_1)$ fulfills already the condition $\tau(H') = \nu(H')$. A maximum packing of H consists of a packing of H' and one additional hyperedge from $\mathcal{E}_1$, and a minimum transversal of H consists of a minimum transversal of H' and additionally the vertex v. Thus, also $\tau(H) = \nu(H)$ holds. ■

The α-acyclicity of a hypergraph H can also be characterized in terms of an inequality concerning the weighted line graph of H. This was shown by Acharya and Las Vergnas in the hypergraph community (see Theorem 29.14) but was also discovered by Bernstein and Goodman [51] in the database community.

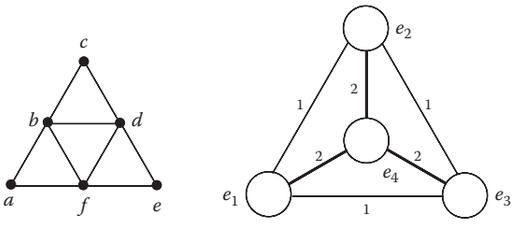

Figure 29.1 A 3-sun with its four maximal cliques $e_1 = \{a, b, f\}$, $e_2 = \{b, c, d\}$, $e_3 = \{d, e, f\}$ and $e_4 = \{b, d, f\}$ and the weighted line graph of them.

Definition 29.18 *Given a hypergraph $H = (V, \mathcal{E})$ with $\mathcal{E} = \{e_1, \ldots, e_m\}$, let $L_w(H)$ denote the* weighted line graph *of H whose nodes are the hyperedges of H which are pairwise connected and the edges are weighted by $w(e_ie_j) = |e_i \cap e_j|$. For any edge set F of $L(H)$, let $w(F)$ denote the sum of all edge weights in F. Let w_H denote the maximum weight of a spanning tree in $L_w(H)$. For a spanning tree T of $L_w(H)$, let T_v denote the subgraph of T induced by the hyperedges containing v, let $N(T_v)$ denote its node set and $E(T_v)$ its edge set.*

If T is a spanning tree of $L_w(H)$ then obviously for every vertex $v \in V$, the following inequality holds (Figure 29.1):

$$1 \leq |N(T_v)| - |E(T_v)|. \tag{29.1}$$

Since any tree with $k \geq 2$ nodes has $k - 1$ edges, equality holds in (29.1) exactly when T_v is a subtree of T. The following lemma summarizes what is implicitly contained in Theorem 29.14.

Lemma 29.5 [52] *Let $H = (V, \mathcal{E})$ be a hypergraph with $\mathcal{E} = \{e_1, \ldots, e_m\}$ and let $L_w(H)$ be as in Definition* 29.18. *Then a spanning tree T of $L_w(H)$ is a join tree of H if and only if*

$$|V| = \sum_{j=1}^{m} |e_j| - \sum_{ij \in E(T)} |e_i \cap e_j|. \tag{29.2}$$

Proof. Suppose T is a spanning tree of $L_w(H)$. For each $v \in V$, the subgraph T_v consisting of all hyperedges containing v satisfies $1 \leq |N(T_v)| - |E(T_v)|$ as described in (29.1), with equality if and only if, for all $v \in V$, T_v is connected. Summing over all $v \in V$ in (29.1) proves that inequality

$$|V| \leq \sum_{j=1}^{m} |e_j| - \sum_{ij \in E(T)} |e_i \cap e_j| \tag{29.3}$$

holds, and equality holds in (29.3) if and only if the spanning tree T is a join tree. ■

Note that the result of summing the right hand side of (29.1) is $\Sigma_{j=1}^{m}|e_j| - w(T)$ for the spanning tree T of $L_w(H)$. Thus also

$$|V| \leq \sum_{j=1}^{m} |e_j| - \max\{w(T) \mid T \text{ spanning tree of } L_w(H)\} = \sum_{j=1}^{m} |e_j| - w_H \tag{29.4}$$

with equality in (29.4) if and only if H has a join tree.

Inequality (29.4) led to the following parameter (see [53–55]):

Definition 29.19 *Let $H = (V, \mathcal{E})$ be a hypergraph and w_H as in Definition* 29.18. *The* cyclomatic number $\mu(H)$ of H *is defined as*

$$\mu(H) = \sum_{j=1}^{m} |e_j| - |V| - w_H.$$

Note that the cyclomatic number of a hypergraph can be efficiently determined by any maximum spanning tree algorithm. Now, the following theorem is a simple corollary of Lemma 29.5.

Theorem 29.14 [53] *A hypergraph H satisfies $\mu(H) = 0$ if and only if H is α-acyclic.* ■

Note that Lemma 29.5 respectively Theorem 29.14 suggests a way how to find a join tree of an α-acyclic hypergraph, namely, taking any maximum spanning tree (determined, e.g., by Kruskal's greedy algorithm) of the weighted line graph $L_w(H)$. Independently, this has been discovered in the database community by Bernstein and Goodman [51] and rediscovered several times; see Chapter 2 of the monograph by McKee and McMorris [5].

However, this is not the most efficient way to construct a clique tree of a given chordal graph; Theorem 29.3 gives a linear-time algorithm for constructing a clique tree.

29.3.6 Graham's Algorithm, Running Intersection Property, and Other Desirable Properties Equivalent to α-Acyclicity

In this subsection, we collect some properties which are equivalent to α-acyclicity of a hypergraph. Some of these conditions are desirable properties of relational database schemes as mentioned in the introduction. Beeri et al. [29] give a long list of such equivalences; we mention here only some of them and give a few proofs which might be suitable for a first glance at this field of research.

In Corollary 29.1 we have seen: A graph G is chordal if and only if G has a p.e.o.

A generalization of this for α-acyclic hypergraphs is known under the name *Graham's Algorithm* (or *Graham Reduction*):

Definition 29.20 [56,57] *Let $H = (V, \mathcal{E})$ be a hypergraph.*

i. Graham's Algorithm *on H applies the following two operations to H repeatedly as long as possible:*

   1. *If a vertex $v \in V$ is contained in exactly one hyperedge $e \in \mathcal{E}$ then delete v from e.*
   2. *If a hyperedge e is contained in another hyperedge e' then delete e.*

ii. Graham's Algorithm succeeds on H *if repeatedly applying the two operations leads to empty hypergraph, that is, to $\mathcal{E} = \{\emptyset\}$.*

Graham's algorithm is also called *GYO algorithm* since Yu and Ozsoyoglu [57] came to exactly the same algorithm. Vertices which occur in only one edge are frequently called *ear vertices* (*isolated vertices* in [29]) and edges containing such a vertex are frequently called *ears* (*knobs* in [29]). Note that any ear node in H is simplicial in the 2-section graph of H.

Theorem 29.15 [29,32] *H is α-acyclic if and only if Graham's algorithm succeeds on H.*

Proof. "$\Longrightarrow$": Let H be α-acyclic, that is, by Corollary 29.5, H is conformal and $2SEC(H)$ is chordal. If H is not Sperner then the (possibly repeated) application of rule (2) leads to a Sperner hypergraph H' which is conformal and for which $2SEC(H')$ is chordal. By Lemma 29.4, H' is isomorphic to the maximal clique hypergraph $\mathcal{C}(2SEC(H'))$.

Let $(v_1, \ldots, v_n)$ be a p.e.o. of $2SEC(H')$. Then v_1 is simplicial and thus contained in only one hyperedge of H', that is, v_1 can be deleted by rule (1). Now the same argument can be repeated and shows the assertion.

"$\Longleftarrow$": Assume that Graham's Algorithm succeeds on H. Then, the repeated application of rules (1) and (2) defines a vertex ordering $\sigma = (v_1, \ldots, v_n)$ of V (i.e., the ordering in which by rule (1), the vertices get deleted). We claim that σ is a p.e.o. Indeed, for each $i \in \{1, \ldots, n\}$, when (1) is applicable to v_i, this vertex is contained in only one hyperedge and thus is simplicial in the remaining 2-section graph.

We finally show that H is conformal: Let C be a clique in $2SEC(H)$ and let v_i be its leftmost element in σ. Then, when eliminating this vertex by rule (1), v_i is contained in only one hyperedge, say e, and all its neighbors in the 2-section graph are in e including C, that is, $C \subseteq e$ which means that H is conformal. ■

Note that Graham's algorithm produces a perfect elimination ordering of the 2-section graph of H if H is α-acyclic.

The *Running Intersection Property* is another notion from the database community which turns out to be equivalent to α-acyclicity of a hypergraph:

Definition 29.21 [29] *Let $H = (V, \mathcal{E})$ be a hypergraph. H has the* running intersection property *if there is an ordering $(e_1, e_2, \ldots, e_m)$ of $\mathcal{E}$ such that for all $i \in \{2, \ldots, m\}$, there is a $j < i$ such that $e_i \cap (e_1 \cup \ldots \cup e_{i-1}) \subseteq e_j$.*

Theorem 29.16 [29,32] *A hypergraph is α-acyclic if and only if it has the running intersection property.*

Proof. "$\Longrightarrow$": If the hypergraph $H = (V, \mathcal{E})$ is α-acyclic then it has a join tree T. Select a root for T. Let $(e_1, \ldots, e_m)$ be an ordering of $\mathcal{E}$ by increasing depth. Thus, if e_j is the parent of e_i, then $j < i$. Clearly, each path from e_i to any of $e_1, \ldots, e_{i-1}$ must pass through e_i's parent e_j. Now if $v \in V$ is a vertex in $e_i \cap e_k$ for some $k < i$, then all hyperedges along the T-path between e_i and e_k contain v. Since this path passes through e_j, it follows that $v \in e_j$ which implies $e_i \cap (e_1 \cup \ldots \cup e_{i-1}) \subseteq e_j$. Thus, H has the running intersection property.

"$\Longleftarrow$": Let $H = (V, \mathcal{E})$ be a hypergraph and let $(e_1, \ldots, e_m)$ be an ordering of $\mathcal{E}$ fulfilling the running intersection property. The proof is by induction on the number m of hyperedges. The basis $m = 2$ is trivial. $(e_1, \ldots, e_{m-1})$ also has the running intersection property, and by induction hypothesis, there is a join tree T' for $e_1, \ldots, e_{m-1}$. Let T be obtained from T' by adding node e_m and edge $e_m e_j$ for a j such that $e_m \cap (e_1 \cup \ldots \cup e_{m-1}) \subseteq e_j$. Obviously, T is a join tree for $e_1, \ldots, e_m$. ■

For the next theorem, we need a few more definitions.

A *path* between two vertices $u, v \in V$ in hypergraph $H = (V, \mathcal{E})$ is a sequence of $k \geq 1$ edges $e_1, \ldots, e_k \in \mathcal{E}$ such that $u \in e_1$, $v \in e_k$ and for all $i = 1, \ldots, k-1$, $e_i \cap e_{i+1} \neq \emptyset$.

H is *connected* if for all pairs $u, v \in V$, there is a path between u and v in H.

The *connected components* of H are the maximal connected vertex-subhypergraphs of H.

For a reduced hypergraph $H = (V, \mathcal{E})$ and two edges $e, e' \in \mathcal{E}$, $e \cap e'$ is an *edge-intersection-separator*, *e.i.-separator* for short (called an *articulation set* in [29]) if the reduced vertex-subhypergraph $H[V \setminus (e \cap e')]$ has more connected components than H.

A hypergraph $H = (V, \mathcal{E})$ is *edge-intersection-separable*, *e.i.-separable* for short (called *acyclic* in [29]) if for each $U \subseteq V$, if the reduction of $H[U]$ is connected and has more than one edge (i.e., is nontrivial) then it has an edge-intersection-separator.

A hyperedge subset $\mathcal{F} \subseteq \mathcal{E}$ is *closed* if for each $e \in \mathcal{E}$, there is an edge $f \in \mathcal{F}$ such that $e \cap \bigcup \mathcal{F} \subseteq f$.

A reduced hypergraph $H = (V, \mathcal{E})$ is *closed-e.i.-separable* (called *closed-acyclic* in [29]) if for each $U \subseteq V$, if $H[U]$ is connected and has more than one edge and its set of edges is closed then it has an e.-i.-separator. A hypergraph is *closed-e.i.-separable* if its reduction is.

Note that in this definition, separators are always intersections of edges.

In [58], it is shown that a hypergraph is acyclic if and only if it is closed-acyclic, that is, e.i.-separable and closed-e.i.-separable are equivalent notions. This has the advantage that it is not necessary to deal with partial edges that are not edges.

Recall that in Section 29.3.1, pairwise and global consistency, semijoins and full reducers, monotone join expressions, and monotone sequential join expressions are defined.

Apparently, there is a close connection between the Helly property of a hypergraph and the equivalence between pairwise and global consistency of a relational database scheme (see Definition 29.6): A relational database $r_1, \ldots, r_m$ over scheme $\mathcal{E} = \{e_1, \ldots, e_m\}$ is pairwise consistent if for every pair $i, j \in \{1, \ldots, m\}$, r_i, r_j is consistent. Let R_i, $i \in \{1, \ldots, m\}$, denote the set of relations over at least the attributes e_i such that the projection to e_i is r_i. In other words, pairwise consistency means that for all $i, j \in \{1, \ldots, m\}$, the intersection $R_i \cap R_j$ is nonempty. Global consistency means that the intersection $\bigcap_{i=1}^{m} R_i$ is nonempty.

The next theorem is part of the main theorem in [29] which contains various other conditions. See the same paper for a detailed discussion of other papers where parts of these equivalences were shown.

Theorem 29.17 [29] *Let $\mathcal{E}$ be a hypergraph. The following conditions are equivalent:*

i. *$\mathcal{E}$ has the running intersection property.*

ii. *$\mathcal{E}$ has a monotone sequential join expression.*

iii. *$\mathcal{E}$ has a monotone join expression.*

iv. *every pairwise consistent database over $\mathcal{E}$ is globally consistent.*

v. *$\mathcal{E}$ is closed-e.i.-separable.*

vi. *every database over $\mathcal{E}$ has a full reducer.*

vii. *the GYO reduction algorithm succeeds on $\mathcal{E}$.*

viii. *$\mathcal{E}$ has a join tree (i.e., $\mathcal{E}$ is α-acyclic).*

Proof. In Theorems 29.16 and 29.17, it is already shown that conditions (i), (vii), and (viii) are equivalent.

(i) $\Longrightarrow$ (ii): Assume that $\mathcal{E}$ has the running intersection property. Let $(e_1, e_2, \ldots, e_m)$ be an ordering of $\mathcal{E}$ such that for all $i \in \{2, \ldots, m\}$, there is a $j_i < i$ such that $e_i \cap (e_1 \cup \ldots \cup e_{i-1}) \subseteq e_{j_i}$. Now we show that $(\ldots((e_1 \bowtie e_2) \bowtie e_3) \ldots \bowtie e_m)$ is a monotone, sequential join expression: If $r = \{r_1, \ldots, r_m\}$ is a pairwise consistent database over $\mathcal{E} = \{e_1, e_2, \ldots, e_m\}$, then the join $r_1 \bowtie \ldots \bowtie r_i$ (which we abbreviate as q_i) is consistent with r_{i+1} $(1 \leq i < n)$.

An easy inductive argument shows that $r_k = q_i[e_k]$ whenever $k \leq i$. In particular, let $k = j_{i+1}$, and let $V := e_{i+1} \cap (e_1 \cup \ldots \cup e_i)$. Since $V \subseteq e_m$, it follows that $r_k[V] = q_i[V]$. But also $r_{i+1}[V] = r_k[V]$ since r_{i+1} and r_k are consistent. Hence $r_{i+1}[V] = q_i[V]$. So r_{i+1} is consistent with q_i which was to be shown.

(ii) $\Longrightarrow$ (iii): This is immediate since every monotone sequential join expression is a monotone join expression.

(iii) $\Longrightarrow$ (iv): Assume that $\mathcal{E}$ has a monotone join expression. We must show that every pairwise consistent database over $\mathcal{E}$ is globally consistent. Let r be a pairwise consistent database over $\mathcal{E}$. It is not hard to see that since no tuples are lost in joining together the relations in r as dictated by the monotone join expression, it follows that every member of r is a projection of the final result $\bowtie r$. Hence r is globally consistent, which was to be shown.

(iv) $\Longrightarrow$ (v): Details are described in [29].

(v) $\Longrightarrow$ (i): Details are described in [29]—GYO reduction succeeds.

(iv) $\Longrightarrow$ (vi): Assume that every pairwise consistent database over $\mathcal{E}$ is globally consistent. Let $r_1, \ldots, r_m$ be a database over $\mathcal{E}$. We have to show that $r_1, \ldots, r_m$ has a full reducer, that is, after finitely many semijoins $r_i \ltimes r_j$, we obtain a globally consistent database. Note that when further semijoin operations do not change anything, the resulting relations are pairwise consistent. By assumption, these are also globally consistent which means that we have a full reducer.

(vi) $\Longrightarrow$ (iv): Assume that every database over $\mathcal{E}$ has a full reducer. Let $r_1, \ldots, r_m$ be a pairwise consistent database over $\mathcal{E}$. By assumption, it has a full reducer but in the case of pairwise consistent relations, the input and output of the full reducer is the same, that is, the result of the full reducer is the database $r_1, \ldots, r_m$ itself, and the result of a full reducer is guaranteed to be globally consistent. Thus, $r_1, \ldots, r_m$ is globally consistent. ■

29.3.7 Dually Chordal Graphs, Maximum Neighborhood Orderings, and Hypertrees

Theorem 29.2 says that a graph is chordal if and only if it has a clique tree, that is, a graph G is chordal if and only if its hypergraph $\mathcal{C}(G)$ of maximal cliques is α-acyclic (or co-arboreal). The dual variant of this means that $\mathcal{C}(G)$ is a hypertree; the corresponding graph class called *dually chordal graphs* was studied in [59–61] and has remarkable properties. In particular, the notion of *maximum neighbor* and *maximum neighborhood ordering* (used in [60,62,63]) has many consequences for algorithmic applications and is somehow dual to the notion of a simplicial vertex. For the next definition, we need the notation of neighborhood in the remaining subgraph.

Let $G = (V, E)$ be a graph and $(v_1, \ldots, v_n)$ be a vertex ordering of G. For all $i \in \{1, \ldots, n\}$, let $G_i := G[\{v_i, \ldots, v_n\}]$ and $N_i[v]$ be the neighborhood of v in G_i: $N_i[v] := N[v] \cap \{v_i, \ldots, v_n\}$.

Definition 29.22 *Let $G = (V, E)$ be a graph.*

i. *A vertex $u \in N[v]$ is a* maximum neighbor of v *if for all $w \in N[v]$, $N[w] \subseteq N[u]$, that is, $N^2[v] = N[u]$. (Note that possibly $u = v$ in which case v sees all vertices of G.)*

ii. *A vertex ordering $(v_1, v_2, \ldots, v_n)$ of V is a* maximum neighborhood ordering *of G if for all $i \in \{1, \ldots, n\}$, v_i has a maximum neighbor in G_i, that is, there is a vertex $u_i \in N_i[v_i]$ such that for all $w \in N_i[v_i]$, $N_i[w] \subseteq N_i[u_i]$ holds.*

iii. *A graph is* dually chordal *if it has a maximum neighborhood ordering.*

Note that dually chordal graphs are not a hereditary class; adding a universal vertex makes every graph dually chordal. The following characterization of dually chordal graphs shows that these graphs are indeed dual (in the hypergraph sense) with respect to chordal graphs:

Theorem 29.18 [59,61] *For a graph G, the following conditions are equivalent:*

i. *G has a maximum neighborhood ordering.*

ii. *There is a spanning tree T of G such that every maximal clique of G induces a subtree of T.*

iii. *There is a spanning tree T of G such that every disk of G induces a subtree of T.*

iv. *$\mathcal{N}(G)$ is a hypertree.*

v. *$\mathcal{N}(G)$ is α-acyclic.*

Proof. Let $G = (V, E)$ be a graph.

(i) $\Longrightarrow$ (ii): By induction on $|V|$. Let x be the leftmost vertex in a maximum neighborhood ordering of G and let y be a maximum neighbor of x, that is, $N^2[x] = N[y]$. If $x = y$, that is, $N^2[x] = N[x]$, then x sees all other vertices of G; let T be a star with central vertex x which fulfills (ii). Now assume that $x \neq y$; by induction hypothesis, there is a spanning tree of the graph $G - x$ fulfilling (ii) for $G - x$. Among all such spanning trees, choose a tree T in which y is adjacent to a maximum number of vertices from $N(x)$.

Claim 29.1 *In T, y sees all vertices of $N(x) \setminus \{y\}$.*

Proof of Claim 29.1. Assume to the contrary that there is a vertex $z \in N(x) \setminus \{y\}$ which is nonadjacent to y in T. Consider the T-path $y - \ldots - v - z$ connecting y and z. Let T_v (T_z, respectively) be the connected component of T obtained by deleting the T-edge vz such that T_v contains v (T_z contains z, respectively). Adding to these subtrees T_v, T_z a new edge yz, we obtain the tree T'. Since y and z are adjacent in $G - x$, T' is a spanning tree of $G - x$. Now we show that T' fulfills condition (ii) as well.

Let Q be a maximal clique of $G - x$. If $z \notin Q$ then Q is completely contained in one of the subtrees T_v, T_z, that is, Q induces one and the same subtree in both T and T'. Now suppose that $z \in Q$. Since $N[z] \subseteq N[y] = N^2[x]$, we have $y \in Q$ by maximality of Q. Let u_1, u_2 be any two vertices of Q. If both belong to the same subtree T_v or T_z then u_1 and u_2 are connected by the same path in T and T', and we are done. Now let u_1 be in T_v and u_2 be in T_z. In T_v, the vertices y and u_1 are connected by a T-path P_1 consisting of vertices from Q. In a similar way, the vertices z and u_2 are connected by a T-path P_2 in T_z. Gluing together these paths P_1 and P_2 with the edge yz, we obtain a T'-path connecting u_1 and u_2 in T'. Hence any maximal clique Q of $G - x$ induces a subtree in T', that is, T' satisfies condition (ii) as well. This, however, contradicts to the choice of T; thus, in T, y sees all vertices of $N(x) \setminus \{y\}$ which shows Claim 29.1.

Now let T be a spanning tree fulfilling the claim for $G - x$. Let T^* be the tree obtained from T by adding a leaf x adjacent to y. Obviously, T^* fulfills condition (ii).

(ii) $\Longrightarrow$ (iii): Let T be a spanning tree of G such that every clique of G induces a subtree in T. We claim that every disk $N^r[z]$ induces a subtree in T as well. In order to prove this, it is sufficient to show that the vertex z and every vertex $v \in N^r[z]$ are connected by a T-path consisting of vertices from $N^r[z]$. Let $v = v_1 - v_2 - \ldots - v_k - v_{k+1} = z$ be a shortest G-path between v and z. By Q_i we denote a maximal clique of G containing the edge $v_i v_{i+1}$, $i \in \{1, \ldots, k\}$. From the choice of T, it follows that v_i and v_{i+1} are connected by a T-path $P_i \subseteq Q_i$. The vertices of $P = \bigcup_{i=1}^{k} P_i$ induce a subtree $T[P]$ of T. Thus, v and z are connected by a T-path p. Since for all vertices $w \in Q_i$, for the G-distances $d(z, w) \leq d(z, v_i) \leq r$ holds,

every clique Q_i is contained in the disk $N^r[z]$. Thus, the claim follows from the obvious inclusion

$$p \subseteq P \subseteq \bigcup_{i=1}^{k} Q_i \subseteq N^r[z]$$

(iii) $\Longrightarrow$ (iv) is obvious.

(iv) $\Longleftrightarrow$ (v) is obvious by the self-duality of the neighborhood hypergraph $\mathcal{N}(G)$ and the duality between hypertree and α-acyclicity.

(iv) $\Longrightarrow$ (i): Let $\mathcal{N}(G)$ be a hypertree. Then by Theorem 29.13, $\mathcal{N}(G)$ has the Helly property and $L(\mathcal{N}(G))$ is chordal. Let $\sigma = (e_1, \ldots, e_m)$ be a perfect elimination ordering of $L(\mathcal{N}(G))$. Since the hyperedges e_i of $\mathcal{N}(G)$ are the closed neighborhoods, $\sigma = (N[v_1], \ldots, N[v_n])$. Suppose inductively that there is a maximum neighborhood ordering for $G - v_1$. It suffices to show that v_1 has a maximum neighbor u_1. Since $N[v_1]$ is simplicial in $L(\mathcal{N}(G))$, the closed neighborhoods intersecting $N[v_1]$ are pairwise intersecting. By the Helly property of $\mathcal{N}(G)$, there is a vertex u_1 in the intersection of all such closed neighborhoods including $N[v_1]$ itself, that is, there is a vertex u_1 with $N^2[v_1] = N[u_1]$. Thus, u_1 is a maximum neighbor of v_1.

The equivalence of (i) and (iv) can be shown in an easy direct way as follows: We know already $(iv) \Longrightarrow (i)$. By Theorem 29.13, we can also show the other direction

$(i) \Longrightarrow (iv)$: Let G have the maximum neighborhood ordering $\sigma = (v_1, \ldots, v_n)$. We have to show that $\mathcal{N}(G)$ has the Helly property and $L(\mathcal{N}(G))$ is chordal.

Let $N[x_1], \ldots, N[x_k]$ be a collection of pairwise intersecting closed neighborhoods in G. Without loss of generality, let x_1 be the leftmost vertex of $x_1, \ldots, x_k$ in σ. Then x_1 has a maximum neighbor u_1, that is, there is a vertex u_1 for which $N^2[x_1] = N[u_1]$. Then $u_1 \in \bigcap_{i=1}^{k} N[x_i]$, and thus, $\mathcal{N}(G)$ has the Helly property.

Now we show that $N[v_1]$ is simplicial in $L(\mathcal{N}(G))$: Let $N[x]$ and $N[y]$ be closed neighborhoods intersecting $N[v_1]$. Let v_1 have a maximum neighbor u_1, that is, $N^2[v_1] = N[u_1]$. Since $x, y \in N^2[v_1]$, it follows that $u_1 \in N[x] \cap N[y]$ and thus, $N[v_1]$ is simplicial in $L(\mathcal{N}(G))$. Inductively, it follows that $\sigma = (N[v_1], \ldots, N[v_n])$ is a perfect elimination ordering of $L(\mathcal{N}(G))$. ■

Since $L(\mathcal{N}(G))$ is isomorphic to G^2 (recall Proposition 29.7), Theorem 29.18 implies the following corollary.

Corollary 29.7 *Graph G is dually chordal if and only if G^2 is chordal and $\mathcal{N}(G)$ has the Helly property.*

Another characterization which follows from the basic properties is the following.

Corollary 29.8 *Graph G is dually chordal if and only if $G = L(H)$ for some α-acyclic hypergraph H.*

As a corollary of Theorem 29.18, dually chordal graphs can be recognized in linear time since α-acyclicity of $\mathcal{N}(G)$ can be tested in linear time [20]. Parts of Theorem 29.18 were found also by Szwarcfiter and Bornstein [64] and later again by Gutierrez and Oubiña [65]; in particular, it was shown in [64] that dually chordal graphs are the clique graphs of intersection graphs of paths in a tree. This implies that dually chordal graphs are the clique graphs of chordal graphs (in the sense of Definition 29.14). See [63,66] for algorithmic applications of maximum neighborhood orderings and [3] for more structural details. In [19], a linear-time algorithm for constructing a special (canonical) maximum neighborhood ordering for a dually chordal graph is described.

New characterizations of dually chordal graphs in terms of separator properties are given by De Caria and Gutierrez in [67–70]. Another new characterization was found by Leitert in [71].

In [72], Moscarini introduced the concept of *doubly chordal graphs*, that is, the graphs which are chordal and dually chordal. This class was introduced for efficiently solving the Steiner problem (motivated by database theory); this can be done, however, also for the larger class of dually chordal graphs (see [63]) and also for the class of homogeneously orderable graphs which contain the dually chordal graphs [73]:

$$\text{doubly chordal} \subset \text{dually chordal} \subset \text{homogeneously orderable}$$

29.3.8 Bipartite Graphs, Hypertrees, and Maximum Neighborhood Orderings

For bipartite graphs $B = (X, Y, E)$, the one-sided neighborhood hypergraphs are of fundamental importance. Let

$$\mathcal{N}^X(B) = \{N(y) \mid y \in Y\} \text{ as well as}$$

$$\mathcal{N}^X(B) = \{N(x) \mid x \in X\}.$$

Note that $(\mathcal{N}^X(B))^* = \mathcal{N}^Y(B)$ and vice versa.

Motivated by database schemes, the following concepts were introduced.

Definition 29.23 [28] *Let $B = (X, Y, E)$ be a bipartite graph.*

i. *B is* X-conformal *if for all $S \subseteq Y$ with the property that all vertices of S have pairwise distance* 2*, there is an $x \in X$ with $S \subseteq N(x)$.*

ii. *B is* X-chordal *if for every cycle C in B of length at least* 8*, there is a vertex $x \in X$ which is adjacent to at least two vertices of C whose distance in C is at least* 4*.*

Analogously, define Y-conformal *and* Y-chordal *for bipartite graphs.*

These notions are justified by the following simple facts.

Proposition 29.10 [28] *Let $B = (X, Y, E)$ be a bipartite graph.*

i. *B is X-conformal $\Longleftrightarrow$ $\mathcal{N}^Y(B)$ is conformal $\Longleftrightarrow$ $\mathcal{N}^X(B)$ has the Helly property.*

ii. *B is X-chordal $\Longleftrightarrow$ $2SEC(\mathcal{N}^Y(B))$ is chordal $\Longleftrightarrow$ $L(\mathcal{N}^X(B))$ is chordal.*

Corollary 29.9 *The following conditions are equivalent:*

i. *B is X-chordal and X-conformal;*

ii. *$\mathcal{N}^Y(B)$ is α-acyclic;*

iii. *$\mathcal{N}^X(B)$ is a hypertree.*

Maximum neighborhood orderings can be defined for bipartite graphs as well. For this we need the following notations: Let $B = (X, Y, E)$ be a bipartite graph, and let $(y_1, \ldots, y_n)$ be a vertex ordering of Y. Then let $B_i^Y = B[X \cup \{y_i, y_{i+1}, \ldots, y_n\}]$ and let $N_i(x)$ denote the neighborhood of x in the remaining subgraph B_i^Y.

Definition 29.24 *Let $B = (X, Y, E)$ be a bipartite graph.*

i. *For $y \in Y$, a vertex $x \in N(y)$ is a* maximum neighbor of y *if for all $x' \in N(y)$, $N(x') \subseteq N(x)$ holds.*

ii. *A linear ordering* $(y_1, \ldots, y_n)$ *of* Y *is a* maximum X-neighborhood ordering *of* B *if for all* $i \in \{1, \ldots, n\}$*, there is a maximum neighbor* $x_i \in N_i(y_i)$ *of* y_i*: for all* $x \in N(y_i)$*,* $N_i(x) \subseteq N_i(x_i)$ *holds.*

Analogously, maximum Y-neighborhood orderings *are defined.*

Theorem 29.19 [59] *Let* $B = (X, Y, E)$ *be a bipartite graph. The following conditions are equivalent:*

i. B *has a maximum* X*-neighborhood ordering;*

ii. B *is* X*-conformal and* X*-chordal.*

Moreover, $(y_1, \ldots, y_n)$ *is a maximum* X*-neighborhood ordering of* B *if and only if* $(y_1, \ldots, y_n)$ *is a p.e.o. of* $2SEC(\mathcal{N}^Y(B))$*.*

Proof. (i) $\Longrightarrow$ (ii): Let $\sigma = (y_1, \ldots, y_n)$ be a maximum X-neighborhood ordering of Y.

(a) B is X-conformal: Assume that the vertices in $S \subseteq Y$ have pairwise distance 2. Let $y_j \in S$ be the leftmost vertex of S in σ and let x be a maximum neighbor of y_j in B_j^Y. Since every $y' \in S$ has a common neighbor $x' \in X$ with y_j, also x is adjacent to y' which implies $S \subseteq N(x)$. Thus, B is X-conformal.

(b) B is X-chordal: Let $C = (x_{i_1}, y_{i_1}, \ldots, x_{i_k}, y_{i_k})$, $k \geq 4$, be a cycle in B. If C has a chord then it has an X-vertex which fulfills the condition. Now assume that C is a chordless cycle. Let $y_{i_1} = y_j$ be the leftmost Y-vertex of C in $(y_1, \ldots, y_n)$. Since $y_{i_k} \in N_j(x_{i_1}) \setminus N_j(x_{i_2})$ and $y_{i_2} \in N_j(x_{i_2}) \setminus N_j(x_{i_1})$, the sets $N_j(x_{i_1})$ and $N_j(x_{i_2})$ are incomparable with respect to set inclusion. Thus, neither x_{i_1} nor x_{i_2} are maximum neighbors of y_{i_1}. Let x be a maximum neighbor of $y_{i_1} = y_j$. Then $y_{i_1}, y_{i_2}, y_{i_k} \in N_j(x)$. Thus, B is X-chordal.

(ii) $\Longrightarrow$ (i): Let B be X-conformal and X-chordal. Then by Proposition 29.10, the line graph $G' = L(\mathcal{N}^X(B))$ is chordal and $\mathcal{N}^X(B)$ has the Helly property. Let $(y_1, \ldots, y_n)$ be a p.e.o. of G'. Thus $N_{G'}[y_1]$ is a a clique, that is, for all $y, y' \in N_{G'}[y_1]$, $N(y) \cap N(y') \neq \emptyset$. By the Helly property of $\mathcal{N}^X(B)$, the total intersection of all $N(y)$ such that $N(y) \cap N(y_1) \neq \emptyset$ is nonempty: there is a vertex $x \in X$ in all these neighborhoods. Now, x is a maximum neighbor of y_1, and the same argument can be repeated with the smaller graph $B - y_1$. ■

Corollary 29.10 *Let* $B = (X, Y, E)$ *be a bipartite graph. The following conditions are equivalent:*

i. B *has a maximum* X*-neighborhood ordering.*

ii. $\mathcal{N}^X(B)$ *is a hypertree.*

iii. $\mathcal{N}^Y(B)$ *is* α*-acyclic.*

Theorems 29.18 and 29.19 imply the following connection between maximum neighborhood orderings in graphs and in bipartite graphs.

Corollary 29.11 [59] *A graph* G *has a maximum neighborhood ordering if and only if* $B(G)$ *has a maximum* X*-neighborhood ordering* (*maximum* Y*-neighborhood ordering, respectively*)*.*

Proof. Recall that by Proposition 29.8, $B(G)$ is isomorphic to the bipartite incidence graph of $\mathcal{N}(G)$. By Theorem 29.18, G has a maximum neighborhood ordering if and only if $\mathcal{N}(G)$ is a hypertree. Now, it is easy to see that the underlying tree of $\mathcal{N}(G)$ immediately leads to the fact that $\mathcal{N}^X(B(G))$ is a hypertree as well, and for symmetry reasons the same happens for $\mathcal{N}^Y(B(G))$. Conversely, if $\mathcal{N}^X(B(G))$ is a hypertree then the underlying tree immediately leads to an underlying tree for $\mathcal{N}(G)$. ■

29.3.9 Further Matrix Notions

As already mentioned, a hypergraph $H = (V, \mathcal{E})$ can be described by its incidence matrix. The notion of a hypertree (see Definition 29.17) is also close to what is called *subtree matrix* in [74].

Definition 29.25

i. *A* Γ *matrix has the form*

$$\begin{array}{|cc|}\hline\hline 1 & 1 \\ 1 & 0 \\ \hline\hline\end{array}$$

ii. *A* subtree matrix *is the incidence matrix of a collection of subtrees of a tree, that is, it is a* $(0,1)$*-matrix with rows indexed by vertices of a tree* T *and columns indexed by some subtrees of* T *and with an entry of 1 if and only if the corresponding vertex is in the corresponding subtree.*

iii. *An ordered* $(0,1)$*-matrix* M *is* supported Γ *if for every pair* $r_1 < r_2$ *of row indices and* $c_1 < c_2$ *of column indices whose entries form a* Γ*, there is a row index* $r_3 > r_2$ *with* $M(r_3, c_1) = M(r_3, c_2) = 1$*. One says that* row r_3 supports the Γ.

Theorem 29.20 [74] *A* $(0,1)$*-matrix is a subtree matrix if and only if it is a matrix with supported* Γ *ordering.*

Proof. "$\Longrightarrow$": Let M be a subtree matrix for a collection $\mathcal{S}$ of subtrees of a tree T. Pick a vertex r of T and order the vertices of T by decreasing distance from r (breaking ties arbitrarily). The distance between r and a subtree S is the minimum distance between r and any vertex from S. Also order the subtrees from $\mathcal{S}$ by decreasing distance from r.

We claim that this is a supported Γ ordering of M, for suppose vertices $v_1 < v_2$ and subtrees $t_1 < t_2$ form a Γ in M: For $i \in \{1, 2\}$, let r_i be the vertex of t_i closest to r. Then $r_1 \geq v_2$ since v_2 is in t_1. We claim that r_1 supports the Γ: Since r_1 is in t_1, $M(r_1, t_1) = 1$. We have to show that r_1 is also in t_2, that is, $M(r_1, t_2) = 1$. If $r_1 = v_1$ or $r_1 = r_2$, we are done. Now suppose that $r_1 \neq v_1$ and $r_1 \neq r_2$.

Since $t_1 < t_2$, r_2 is closer to r than r_1 but t_1 and t_2 contain a common vertex v_1, and thus also r_1 is on the T path between v_1 and r_2, that is, r_1 is in subtree t_2 and supports the Γ.

"$\Longleftarrow$": If the ordered $n \times m$ matrix M is supported Γ, create a tree T on vertex set $\{1, 2, \ldots, n\}$ by setting for $i \in \{1, 2, \ldots, n-1\}$

$$f(i) = \begin{cases} \min\{k \mid M(i,k) = 1\} & \text{if there exists } j > i, M(j,k) = 1 \\ \text{not defined} & \text{otherwise} \end{cases}$$

and $b(i) = \max\{j \mid M(j, f(i)) = 1\}$ and creating the edges $(i, b(i))$ if $f(i)$ exists and (i, n) otherwise. We claim that T defined in this way is a tree: Since $b(i) > i$ when $f(i)$ (and thus also $b(i)$) exists, and the edges (i, n) otherwise, T is obviously cycle-free, and for the same reason, T is connected.

Finally, we show that each column of M is the incidence vector of a subtree of T. It suffices to show that for $i < j$, $M(i,k) = M(j,k) = 1$ implies $M(b(i), k) = 1$: If this were not true then $f(i) < k$ and rows $i, b(i)$ and columns $f(i), k$ would form an unsupported Γ in M. ■

Note that Theorem 29.20 gives a characterization of hypertrees and of α-acyclic hypergraphs in terms of a matrix property. This also gives corresponding characterizations of chordal as well as of dually chordal graphs.

Definition 29.26 *A* $(0,1)$*-matrix* M *is* doubly lexically ordered *if the rows* (*columns, respectively*) *form a lexicographically increasing sequence from top to bottom* (*from left to right, respectively*) *where for rows* (*columns, respectively*)*, the rightmost position* (*lowest position, respectively*) *has highest priority.*

Theorem 29.21 [74] *Every* $(0,1)$*-matrix* M *can be doubly lexically ordered by some suitable permutations of rows and columns.*

Proof. Let $M = (M_{ij})$ be an $m \times n$ matrix. We form a $m \cdot n$ vector $d(M)$ as follows: The entries of M will be ordered with respect to $i + j$, and for the same $i + j$ with respect to j:

$$d(M) = M_{11}, M_{21}, M_{12}, M_{31}, M_{22}, M_{13}, M_{41}, \ldots, M_{mn}$$

$$\begin{vmatrix} d_1 & d_3 & d_6 & \cdot & \ldots \\ d_2 & d_5 & \cdot & & \\ d_4 & \cdot & & & \\ \cdot & & & & \\ \ldots & & & & d_{m\cdot n} \end{vmatrix}$$

Claim 29.2 *If two rows* (*columns, respectively*) *of* M *are permuted which do not appear in lexical order then the result* $d(M)$ *will be lexically larger* (*with highest priority at* $d_{m\cdot n}$).

Proof.of Claim 29.2. Let k, l be row indices of M with $k < l$ and the property that the kth row is lexically larger than the lth row.

Let $j \in \{1, \ldots, n\}$ be the largest index for which $M_{kj} \neq M_{lj}$; then $M_{kj} > M_{lj}$. After permuting the kth and lth row, the part of $d(M)$ which was M_{lj} becomes M_{kj} and the parts on its right hand side do not change their value, and analogously for columns. This shows Claim 29.2.

By Claim 29.2, an ordering of M which maximizes $d(M)$, is a doubly lexical ordering of M. ■

Theorem 29.21 holds also for other ordered matrix entries instead of $\{0, 1\}$.

There is an efficient way for finding a doubly lexical ordering: Let $L := n + m +$ number of 1's in a $(0,1)$-matrix M.

Theorem 29.22 [75] *A doubly lexical ordering of an* $m \times n$ *matrix* M *over entries* $\{0, 1\}$ *can be determined in* $O(L \log L)$ *steps.* ■

29.4 TOTALLY BALANCED HYPERGRAPHS AND MATRICES

29.4.1 Totally Balanced Hypergraphs versus β-Acyclic Hypergraphs

Fagin [6,7] defined β-acyclic hypergraphs in connection with desirable properties of relational database schemes. Recall that for α-acyclic hypergraphs, edge-subhypergraphs are not necessarily α-acyclic.

Definition 29.27 [6,7] *A hypergraph* $H = (V, \mathcal{E})$ *is* β-acyclic *if each of its edge-subhypergraphs is* α*-acyclic, that is, for all* $\mathcal{E}' \subseteq \mathcal{E}$*,* $\mathcal{E}'$ *is* α*-acyclic.*

Fagin [6] gives a variety of equivalent notions of β-acyclicity in terms of certain forbidden cycles in hypergraphs (one of them goes back to Graham [56]) which Fagin in [6] shows to be equivalent.

Actually, β-acyclic hypergraphs appear under the name *totally balanced hypergraphs* much earlier in hypergraph theory (as it will turn out in Theorems 29.25 and 29.27).

Definition 29.28 [12,76] *Let $H = (V, \mathcal{E})$ be a hypergraph.*

i. *A sequence $C = (v_1, e_1, v_2, e_2, \ldots, v_k, e_k)$ of distinct vertices $v_1, v_2, \ldots, v_k$ and distinct hyperedges $e_1, e_2, \ldots, e_k$ is a* special cycle (*or* chordless cycle *or* induced cycle *or, in* [77], unbalanced circuit) *if $k \geq 3$ and for every i, $1 \leq i \leq k$, $v_i, v_{i+1} \in e_i$ (index arithmetic is done modulo k) and $e_i \cap \{v_1, \ldots, v_k\} = \{v_i, v_{i+1}\}$. The* length of cycle *$C$ is k.*

ii. *H is* balanced *if it has no special cycles of odd length $k \geq 3$.*

iii. *H is* totally balanced *if it has no special cycles of any length $k \geq 3$.*

Special cycles are called *weak* β*-cycles* by Fagin in [6], and a hypergraph is called β*-acyclic* if it has no weak β-cycles. Actually, [6] mentions four other conditions and shows that all five are equivalent.

We will see in Theorem 29.27 that a hypergraph is β-acyclic if and only if it is totally balanced. Totally balanced hypergraphs are a natural generalization of trees.

Balanced hypergraphs are a natural generalization of bipartite graphs. See the monograph of Berge [13] for many properties and characterizations of balanced hypergraphs, and in particular, the following theorems.

Theorem 29.23 [13] *A hypergraph is balanced if and only if its vertex-subhypergraphs are 2-colorable.* ■

Theorem 29.24 [78] *A hypergraph is balanced if and only if its vertex- and edge-subhypergraphs have the Kőnig property.* ■

Corollary 29.12 [13] *Balanced hypergraphs have the Helly property and are conformal.*

In this subsection, we focus on totally balanced hypergraphs.

Proposition 29.11 *Let $H = (V, \mathcal{E})$ be a totally balanced hypergraph. Then the following holds:*

i. *The dual H^* of H and any vertex- or edge-subhypergraph of H are totally balanced.*

ii. *H has the Helly property.*

iii. *$L(H)$ is a chordal graph.*

Proof. Let $H = (V, \mathcal{E})$ be a totally balanced hypergraph.

i. By definition, it immediately follows that the dual H^* and any vertex- or edge-subhypergraph of a totally balanced hypergraph H is totally balanced.

ii. Since the Helly property is satisfied by any hypergraph without special cycle of length three (see Theorem 29.8), H must have the Helly property.

iii. $L(H)$ is a chordal graph since H contains no special cycle. ■

Recall that vertex-subhypergraphs of hypertrees are not necessarily hypertrees. The next theorem gives a characterization of totally balanced hypergraphs in terms of hypertrees.

Theorem 29.25 [36,46] *A hypergraph H is totally balanced if and only if every vertex-subhypergraph of H is a hypertree.*

Proof. Let $H = (V, \mathcal{E})$ be a hypergraph. By Proposition 29.11, we have:

i. The dual of H and any vertex- or edge-subhypergraph are totally balanced.

ii. H has the Helly property.

iii. $L(H)$ is a chordal graph.

Now by Theorem 29.13, H (and every vertex-subhypergraph of H) must be a hypertree and vice versa. ■

Actually, Lehel [46] gives a complete structural characterization of totally balanced hypergraphs in terms of certain tree sequences. Lehel's result implies the following characterization of totally balanced hypergraphs which was originally found by Brouwer and Kolen [79] and nicely corresponds to the existence of simple vertices in strongly chordal graphs.

Theorem 29.26 [46,79] *A hypergraph H is totally balanced if and only if every vertex-subhypergraph H' has a vertex v (a so-called* nested point*) such that the hyperedges of H' containing v are linearly ordered by inclusion.* ■

By simple duality arguments, the next theorem follows immediately from Theorem 29.25.

Theorem 29.27 [80] *A hypergraph H is totally balanced if and only if H is β-acyclic.* ■

29.4.2 Totally Balanced Matrices

Definition 29.29 *Let B_k denote the $k \times k$ square $(0,1)$-matrix consisting of rows with exactly two consecutive 1's beginning with $10\ldots01$, then $110\ldots$ and so on; $00\ldots11$ is the last row.*

For example, B_4 is the following matrix:

$$\begin{array}{|cccc|}\hline 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ \hline\end{array}$$

Thus, a hypergraph is totally balanced if and only if its incidence matrix contains no square submatrix B_k for $k \geq 3$ (in any row and column order), and correspondingly for balanced hypergraphs and odd $k \geq 3$. Obviously, the dual of a balanced (totally balanced, resp.) hypergraph is balanced (totally balanced, respectively).

Lubiw in [74] defines totally balanced matrices in the following way.

Definition 29.30

i. *For $n \geq 3$, a* cycle matrix *is an $n \times n$ $(0,1)$-matrix with no identical rows and columns which has exactly two 1's in each row and in each column such that no proper submatrix has this property.*

ii. *A* totally balanced matrix *is a $(0,1)$-matrix with no cycle submatrices.*

Recall Definition 29.25 for the notion of a Γ submatrix.

Definition 29.31 *An ordered $(0,1)$-matrix M is Γ-free if M has no Γ submatrix.*

Theorem 29.28 [81–83] *A $(0,1)$-matrix is totally balanced if and only if it has a Γ-free ordering.* ■

This is shown in [74] as a consequence of the existence of doubly lexical orderings and the following.

Observation 29.1 *If a* $(0,1)$*-matrix has a cycle submatrix then for any ordering of the matrix there is a* Γ *submatrix* (*formed by a topmost, leftmost* 1 *of the cycle submatrix; the other* 1 *in its row in the cycle; and the other* 1 *in its column in the cycle*).

Theorem 29.29 [74] *Any doubly lexical ordering of a totally balanced matrix is* Γ*-free.* ■

For example, the matrix B_4 from Definition 29.29 has the following doubly lexical ordering which is not Γ-free:

1	1	0	0
1	0	1	0
0	1	0	1
0	0	1	1

(resulting from the matrix B_4 by first permuting rows 1 and 2 and then permuting columns 3 and 4).

29.5 STRONGLY CHORDAL AND CHORDAL BIPARTITE GRAPHS

29.5.1 Strongly Chordal Graphs

The subsequently defined *strongly chordal graphs* are an important subclass of chordal graphs for many reasons. Originally, they were introduced by Farber [27] as a subclass of chordal graphs for which the domination problem ([GT2] in [40]), which remains NP-complete for chordal graphs and even for split graphs [84], can be solved efficiently. Chang and Nemhauser [85,86] independently studied the same class and also showed that some problems such as domination can be solved efficiently. Later on, this has been extended to larger classes and other problems (see, e.g., [63,66,73,87]).

The motivation from the database community is the fact that strongly chordal graphs are the 2-section graphs of β-acyclic hypergraphs (as it will turn out in Theorem 29.32 as a consequence of Theorem 29.27).

29.5.1.1 Elimination Orderings of Strongly Chordal Graphs

Farber [27] defined strongly chordal graphs in terms of so-called *strong elimination orderings* which are closely related to neighborhood matrices of these graphs:

Definition 29.32 *Let* $\sigma = (v_1, \ldots, v_n)$ *be an ordering of the vertex set* V *of* G*. The* neighborhood matrix $N_\sigma(G)$ ($N(G)$ *if* σ *is understood*) *of* G *is the* $n \times n$ *matrix with entries*

$$n_{ij} = \begin{cases} 1 & \text{if } v_i \in N[v_j] \\ 0 & \text{otherwise} \end{cases}$$

Note that this matrix is symmetric and the main diagonal has values 1:

$$v_i \in N[v_j] \iff v_j \in N[v_i]$$

($N(G)$ is the incidence matrix of the closed-neighborhood hypergraph $\mathcal{N}(G)$).

The subsequent Definition 29.33 must be read as follows: If in the $(0,1)$ neighborhood matrix of graph G, for $i < j$ and $k < \ell$, the entries in row i and column k, in row i and column ℓ as well as in row j and column k are 1, then the entry in row j and column ℓ must be 1 as well (i.e., rows $i < j$ and columns $k < \ell$ do not form a Γ—see Definition 29.25).

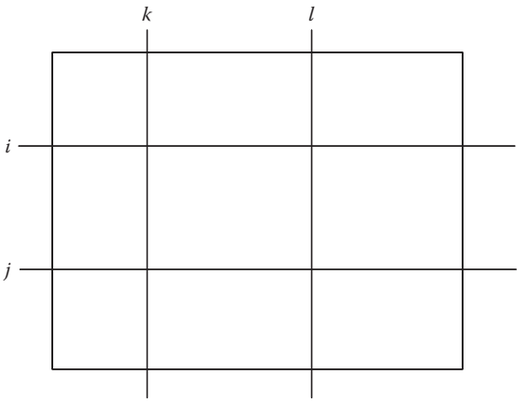

Figure 29.2 Selected rows and columns.

Definition 29.33 [27] *Let $G = (V, E)$ be a graph.*

i. *A vertex ordering $(v_1, \ldots, v_n)$ of G is a* strong elimination ordering (st.e.o.) *if for all i, j, k and ℓ with $i < j, k < \ell$ for $v_k, v_\ell \in N[v_i]$ holds: if $v_j \in N[v_k]$ then also $v_j \in N[v_\ell]$.*

ii. *G is* strongly chordal *if G has a st.e.o.*

Obviously, every st.e.o. is also a p.e.o. (let $i = k$ in condition (i)); thus, strongly chordal graphs are chordal.

Observation 29.2 *Let $\sigma = (v_1, \ldots, v_n)$ be an ordering of the vertex set V of graph G. Then σ is a st.e.o. of G if and only if the corresponding neighborhood matrix $N_\sigma(G)$ is Γ-free.*

Proof. Let $(v_1, \ldots, v_n)$ be a st.e.o. We consider the ith and the jth row as well as the kth and the lth column of the matrix, $i < j$, $k < l$.

Figure 29.2 schematically indicates the selected rows and columns.

If $n_{ik} = 1, n_{il} = 1$ and $n_{jk} = 1$ then $v_k \in N[v_i]$, $v_l \in N[v_i]$, $v_j \in N[v_k]$ and thus also $v_j \in N[v_l]$, that is, $n_{jl} = 1$.

Conversely, if the submatrix $\begin{smallmatrix}1&1\\1&0\end{smallmatrix}$ is forbidden then obviously $(v_1, \ldots, v_n)$ is a st.e.o.

Observation 29.2 describes an important matrix aspect of strongly chordal graphs. We will also show that strongly chordal graphs are the hereditarily dually chordal graphs. For this, we need the following notion:

Definition 29.34 [27] *Let $G = (V, E)$ be a graph.*

i. *A vertex $v \in V$ is called* simple *if the set of closed neighborhoods $\{N[u] \mid u \in N[v]\}$ is linearly ordered with respect to set inclusion.*

ii. *A vertex ordering $(v_1, \ldots, v_n)$ of V is a* simple elimination ordering (si.e.o.) *if for all $i \in \{1, \ldots, n\}$, v_i is simple in $G_i = G[\{v_i, \ldots, v_n\}]$.*

It is easy to see that every simple vertex is also simplicial, that is, whenever a graph has a simple elimination ordering, it is chordal.

For proving Theorem 29.30, we need the following property.

Lemma 29.6 *Let v be simple in $G = (V, E)$ and $u_0 \in N[v]$ be a vertex with smallest neighborhood $N[u_0]$. Then also u_0 is simple in G.*

Proof. Assume that u_0 is not simple. Then let $x, y \in N[u_0]$ be two vertices with incomparable neighborhoods $N[x], N[y]$. Since $\{N[u] \mid u \in N[v]\}$ is linearly ordered with respect to $\subseteq$, for all $u \in N[v]$ $N[u_0] \subseteq N[u]$ holds, in particular for $u = v$, $N[u_0] \subseteq N[v]$. Thus, v has two neighbors with incomparable neighborhood—contradiction. ■

Theorem 29.30 [27] *A graph G has a st.e.o. if and only if every induced subgraph of G contains a simple vertex.*

Proof. "$\Longrightarrow$": If G has a st.e.o. $(v_1, \ldots, v_n)$ then also every induced subgraph of G has such an ordering by Definition 29.33. We show that v_1 is simple.

Let $v_k, v_l \in N[v_1]$ with $k < l$ and $v_j \in N[v_k]$ with $1 < j$. By Definition 29.33, it follows immediately that $v_j \in N[v_l]$. Thus, $N[v_k] \subseteq N[v_l]$, and v_1 is simple (which means that the st.e.o. is also a si.e.o.).

"$\Longleftarrow$": Assume that every induced subgraph of G contains a simple vertex. We recursively construct a st.e.o. $(v_1, \ldots, v_n)$ of G as follows: For every $1 \leq i \leq n$, choose in $G_i = G(\{v_i, \ldots, v_n\})$ a simple vertex v_i with smallest $|N_i[v_i]|$.

We claim that this ordering is a st.e.o. Since the vertex v_i is simple in G_i, that is, for their neighbors from $N_i[v_i]$, the neighborhoods are linearly ordered with respect to $\subseteq$, we have the following.

The vertices from $N_i[v_i]$ appear in $(v_1, \ldots, v_n)$ in the same order (this follows by Lemma 29.6 for G_i). Now, for $i < j$ and $k < l$ let $v_k, v_l \in N[v_i]$ and $v_j \in N[v_k]$.

Case 1 $i < k$. Then v_i is simple in G_i and $v_k, v_l \in N_i[v_i]$ with $k < l$. Thus, $N_i[v_k] \subseteq N_i[v_l]$ and therefore also $v_j \in N[v_l]$.

Case 2 $i = k$. In this case, the assertion is fulfilled since any simple vertex is simplicial.

Case 3 $i > k$. Then v_k is simple in G_k and $v_i, v_j \in N_k[v_k], i < j$. Thus, $N_k[v_i] \subseteq N_k[v_j]$. From $v_l \in N[v_i]$, $l > k$, it follows that $v_l \in N_k[v_i]$, thus also $v_l \in N_k[v_j]$ and finally $v_j \in N[v_l]$. ■

Corollary 29.13 *The following conditions are equivalent:*

i. *G is strongly chordal.*

ii. *G has a si.e.o.*

iii. *G is hereditarily dually chordal, that is, every induced subgraph of G is dually chordal.*

iv. *$\mathcal{N}(G)$ is β-acyclic (i.e., by Theorem* 29.27*, totally balanced).*

Proof. Theorem 29.30 shows the equivalence of (i) and (ii).

For the equivalence of (ii) and (iii), assume first that G has a si.e.o. Then every induced subgraph of G has a si.e.o. as well, and note that a si.e.o. is also a maximum neighborhood ordering which means that every induced subgraph of G is dually chordal. Conversely, let G be a hereditarily dually chordal graph. Let v_1 have a maximum neighbor u_1, that is, the neighborhood of u_1 is largest among all $N[u]$, $u \in N[v_1]$. Then a straightforward discussion shows that also the subgraph of G induced by $N[v_1] - u_1$ has a maximum neighborhood ordering and so on which leads to a linear ordering of neighborhoods w.r.t. v_1, that is, v_1 is simple. Now the same can be repeated for $G[\{v_2, \ldots, v_n\}]$ which shows the equivalence.

The equivalence of (iii) and (iv) is a simple consequence of Theorem 29.18. ■

The equivalence of (i) and (iv) has been obtained independently by Iijima and Shibata [88]; they showed that a graph is sun-free chordal (see Theorem 29.33) if and only if its neighborhood matrix is totally balanced.

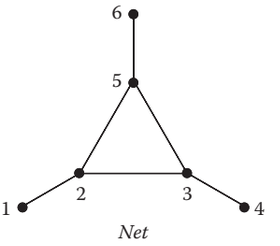

Figure 29.3 The *net*.

29.5.1.2 Γ*-Free Matrices and Strongly Chordal Graphs*

Definition 29.33 implies a useful characterization of strongly chordal graphs by matrices.

Observation 29.2 leads to the fastest known recognition algorithms for strongly chordal graphs by using doubly lexical orderings of matrices as given in Definition 29.26 (which permute rows and columns in a suitable way)—see the subsequent Theorem 29.31 and Corollary 29.14.

Example 29.3 Take the graph G from Figure 5.1 with vertices $1, \ldots, 6$, edges $12, 23, 25, 34, 35, 56$ and vertex ordering $\sigma_1 = (1, 2, 3, 4, 5, 6)$ (Figure 29.3).

The adjacency matrix M of this graph corresponding to σ_1:

	1	2	3	4	5	6
1	1	1	0	0	0	0
2	1	1	1	0	1	0
3	0	1	1	1	1	0
4	0	0	1	1	0	0
5	0	1	1	0	1	1
6	0	0	0	0	1	1

M is not Γ-free and not doubly lexically ordered (e.g., the third and fifth row together with the second and fourth column form a Γ, and likewise the fourth and fifth row together with the third and fourth column) and not doubly lexically ordered (e.g., the third column from the left is larger than the fourth column).

A strong elimination ordering for G is $\sigma_2 = (1, 4, 6, 2, 3, 5)$. The adjacency matrix M' of G resulting from this ordering σ_2 is as follows:

	1	4	6	2	3	5
1	1	0	0	1	0	0
4	0	1	0	0	1	0
6	0	0	1	0	0	1
2	1	0	0	1	1	1
3	0	1	0	1	1	1
5	0	0	1	1	1	1

M' is doubly lexically ordered.

Theorem 29.21 holds also for other ordered matrix entries instead of $\{0, 1\}$.

Recall Theorem 29.22 for an efficient way for finding a doubly lexical ordering. An efficient (but not linear-time) recognition of strongly chordal graphs results from the following property.

Theorem 29.31 [74] *A graph G is strongly chordal if and only if any doubly lexical ordering of its neighborhood matrix $N(G)$ is Γ-free.* ■

The connection to Γ-free matrices has been used by Paige and Tarjan in [75] as well as by Spinrad in [89] (see also [19]) to design fast (but not linear-time) recognition algorithms for strongly chordal graphs.

Corollary 29.14 *Recognition of strongly chordal graphs can be done in time $O(m \cdot \log n)$.*

It is an open problem whether strongly chordal graphs can be recognized in linear time.

Recall that a hypergraph is defined to be totally balanced if it contains no special cycle (Definition 29.28), and recall Corollary 29.13; this has been expressed in terms of totally balanced matrices.

Recall also (see Definition 29.30) that a $(0,1)$-matrix M is *totally balanced* if M contains no submatrix which is the vertex-edge incidence matrix of a cycle of length ≥ 3 of an undirected graph.

Example 29.4 The vertex-edge incidence matrix of C_3 with vertices v_1, v_2, v_3 and edges $e_1 = \{v_1, v_2\}, e_2 = \{v_2, v_3\}, e_3 = \{v_1, v_3\}$ is

	v_1	v_2	v_3
e_1	1	1	0
e_2	0	1	1
e_3	1	0	1

By Corollary 29.13, G is strongly chordal if and only if its closed neighborhood hypergraph $\mathcal{N}(G)$ is totally balanced. Thus, the next theorem is no surprise:

Theorem 29.32 [27] *A graph G is strongly chordal if and only if its neighborhood matrix $N(G)$ is totally balanced.* ■

29.5.1.3 Strongly Chordal Graphs as Sun-Free Chordal Graphs

Strongly chordal graphs have a variety of different characterizations, among them one in terms of forbidden induced subgraphs. Recall that we say a vertex x *sees* a vertex y if x is adjacent to y; otherwise we say x *misses* y.

Definition 29.35

i. *A* k-sun *is a chordal graph G with $2k$ vertices, $k \geq 3$, whose vertex set is partitioned into two sets $W = \{w_0, \ldots, w_{k-1}\}$ and $U = \{u_0, \ldots, u_{k-1}\}$, such that $U = \{u_0, \ldots, u_{k-1}\}$ induces a cycle (the* central clique *of the sun), W is a stable set and for all $i \in \{0, \ldots, k-1\}$, w_i sees exactly u_i and u_{i+1} (index arithmetic modulo k).*

ii. *A* complete k-sun *is a k-sun where $G[U]$ is a clique.*

See, for example, Figure 29.4 for 3-sun and complete 4-sun.

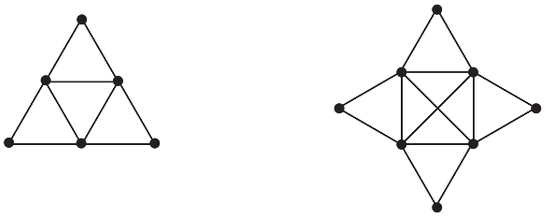

Figure 29.4 3-sun and complete 4-sun.

As shown in [27,85], the following holds.

Lemma 29.7 *In a chordal graph, every k-sun contains a complete k'-sun for some $k' \leq k$.*

Proof. [85] Let $U = \{u_1, \ldots, u_n\}$ and $W = \{w_1, \ldots, w_n\}$ describe the partition of the vertex set of an n-sun G. Since G is chordal and the degree of every vertex w_i in G is 2, its two neighbors u_i and u_{i+1} are adjacent. The proof is by induction on n. If $n = 3$ and $n = 4$ then the claim is obviously fulfilled. Suppose that $n > 4$ and that Lemma 29.7 holds for all suns on fewer than $2n$ vertices and suppose that G is an n-sun which is not complete. Let u_1 miss u_j for some j with $1 < j < n$. Since u_1 sees u_2 and u_n, there exist k and l such that u_1 sees u_k and u_l but misses u_p for any p with $k < p < l$. In that case,

$$G' := G[\{u_1, u_k, u_{k+1}, \ldots, u_l, w_k, \ldots, w_{l-1}\}]$$

is a smaller sun for which $U' := \{u_k, u_{k+1}, \ldots, u_l\}$ and $W' := \{u_1, w_k, w_{k+1}, \ldots, w_{l-1}\}$ gives the required partition. By induction, G' (and hence G) contains a complete sun. ■

Lemma 29.8 [90] *Let $p \geq 3$ be an integer and suppose G is a graph in which every cycle of length k, for $4 \leq k \leq 2p$, has a chord. Then, $\mathcal{N}(G)$ has an induced special cycle C_p if and only if G has an induced p-sun.*

Proof. Clearly, if K is some induced subgraph of G, $\mathcal{N}(K)$ is isomorphic to an induced partial subhypergraph of $\mathcal{N}(G)$. Thus, the *if* part of Lemma 29.8 is easy and left to the reader.

The converse is proved by contradiction: Suppose that every cycle of G with length k, $4 \leq k \leq 2p$, has a chord and suppose G has no induced sun while $\mathcal{N}(G)$ has an induced special cycle C_p with p vertices and p hyperedges. Thus, by definition, there exists a set $A = \{a_1, \ldots, a_p\}$ and a set $B = \{b_1, \ldots, b_p\}$ with the following properties:

1. $(a_1, N[b_1], \ldots, a_p, N[b_p])$ is a special cycle in $\mathcal{N}(G)$.
2. $N[b_j] \cap A = \{a_j, a_{j+1}\}$ for every $j, 1 \leq j \leq p$ (index arithmetic modulo p). (2) is clearly equivalent to
3. For $j \neq i$ or $i + 1$, $a_i \neq b_j$ and $a_i b_j$ is not an edge of G.

(Note that so far, we do not know whether $A \cap B = \emptyset$.)

Claim 29.3 *If $(v_1, v_2, \ldots, v_q)$ is a cycle C of G $(4 \leq q \leq 2p)$, then either $v_2 v_q$ is a chord of C or C has a chord of the form $v_1 v_k$ for some $k, 3 \leq k \leq q - 1$*

The proof easily follows from the assumption that every cycle of length k, for $4 \leq k \leq 2p$, has a chord.

Claim 29.4 *G contains an edge of the form a_ia_j $(i \neq j)$.*

Otherwise, by (1), $A \cap B = \emptyset$. Thus $(a_1, b_1, a_2, b_2, \ldots, a_p, b_p)$ is a cycle of length $2p$ in G which must have a chord. Claim 29.3 together with (3) implies that such a chord is an edge between two vertices from B, and since every cycle of length k, for $4 \leq k \leq 2p$, has a chord, it turns out that b_kb_{k+1} is a chord of this cycle for each k $(1 \leq k \leq p)$. Hence $A \cup B$ induces a (chordal) subgraph of G which is a sun of order p: B is the central clique, and A is the stable set. The contradiction proves Claim 29.4.

Claim 29.5 *If a_ia_j is an edge of G, then a_ia_{i+1} is also an edge of G.*

By symmetry, we may suppose $i = 1$. Let j be the smallest integer for which a_1 sees a_j. If $j > 2$, the vertices a_1, b_1, a_2, a_j are different. $(a_2, b_2, a_3, b_3, \ldots, a_{j-1}, b_{j-1}, a_j)$ is a walk not passing through a_1 or b_1, by (3). This walk induces a minimal path, say P, from a_2 to a_j. By (3) and the definition of j, the cycle (a_1, b_1, P) with length ≥ 4 has no chord containing a_1. Hence b_1 must see a_j (Claim 29.3), in contradiction with (3). So, $j = 2$.

Claim 29.6 *$(a_1, \ldots, a_p)$ is a cycle of G.*

Claim 29.6 is an easy consequence of the previous claim.

Claim 29.7 *$a_i \neq b_j$ for all i, j, that is, $A \cap B = \emptyset$.*

Otherwise $N[b_j]$ would contain a_{i-1}, a_i and a_{i+1} (Claims 29.5 and 29.6), in contradiction with (2) (i.e., the definition of a special cycle).

Claim 29.8 *G contains some edge b_ib_j.*

Otherwise, $A \cup B$ would induce a p-sun with central clique A and stable set B.

For obtaining the final contradiction, we observe that in the last claim i and j play a symmetrical role. So, we may assume without loss of generality that G contains an edge of the form b_1b_j with $j \neq 2$ (the arguments for $j = 2$ are similar). Then G has the following cycle:

$$(b_1, a_2, a_3, a_4, \ldots, a_j, b_j)$$

and, by Claim 29.3, some edge $b_1a_i (3 \leq i \leq j)$ or the edge b_ja_2 must exist, contradicting (3).

Theorem 29.33 [27] *A graph G is strongly chordal if and only if G is sun-free chordal.*

Proof. Theorem 29.33 follows by Lemma 29.8 and Corollary 29.13. ■

A similar characterization of dually chordal graphs was obtained in [60]: A graph G is dually chordal if and only if G is a Helly graph containing no isometric complete k-suns for $k \geq 4$. Recall that G is a *Helly graph* if its disk hypergraph $\mathcal{D}(G)$ is Helly. A subgraph S of G is *isometric* if $d_S(x, y) = d_G(x, y)$ for all vertices x and y of S.

An *odd chord* v_iv_j in an even cycle $(v_1, \ldots, v_{2k})$ is a chord with odd $|i - j|$.

Theorem 29.34 [27] *A graph G is strongly chordal if and only if it is chordal and every even cycle of length at least 6 in G has an odd chord.*

Proof. Let G be a chordal graph. If every even cycle of length at least 6 has an odd chord, then G contains no induced k-sun, $k \geq 3$. Thus, by Theorem 29.33, G is strongly chordal.

Conversely, we use Theorem 29.32: If G is strongly chordal then its neighborhood matrix $N(G)$ is totally balanced. If there is a cycle $(v_1, v_2, \ldots, v_{2k})$ in G without odd chord then the submatrix of $N(G)$ consisting of the rows corresponding to $v_1, v_3, \ldots, v_{2k-1}$ and the columns corresponding to $v_2, v_4, \ldots, v_{2k}$ is precisely the incidence matrix of a cycle of length k. Consequently, G is not strongly chordal. ■

Finally, we give yet another characterization.

Corollary 29.15 [27] *Graph G is strongly chordal if and only if $\mathcal{C}(G)$ is totally balanced.*

Proof. By Theorems 29.18 and 29.25 and Corollary 29.13, G is strongly chordal if and only if $\mathcal{C}(G)$ is totally balanced. ■

It follows from the basic properties that G is strongly chordal if and only if $G = L(H)$ for some totally balanced hypergraph H.

Recall that for a clique tree T of G, the intersections $Q \cap Q'$ of maximal cliques for which $QQ' \in E(T)$ form the minimal vertex separators in G. Let $\mathcal{S}(G)$ denote the separator hypergraph of G. It can be considered as the first derivative of a chordal graph. McKee [91] discusses this concept in detail.

Theorem 29.35 [92] *Graph G is strongly chordal if and only if G is chordal and its separator hypergraph $\mathcal{S}(G)$ is totally balanced.* ■

29.5.2 Chordal Bipartite Graphs

The most natural variant of chordality for bipartite graphs is the following.

Definition 29.36 [93] *A bipartite graph is* chordal bipartite *if each of its cycles of length at least six has a chord.*

In the terminology of Definition 29.5, this means that a bipartite graph is chordal bipartite if and only if it is $(6, 1)$-chordal. Note that chordal bipartite does not mean *chordal and bipartite* (as the name might suggest); if graph G is chordal and bipartite, G is a forest, whereas C_4 is chordal bipartite.

Thus, a better name for chordal bipartite graphs would have been *weakly chordal bipartite* since graph G is chordal bipartite if and only if it is bipartite and *weakly chordal*, that is, every cycle in G and in $\overline{G}$ of length at least five has a chord. See Chapter 28 and [94] for the important class of weakly chordal graphs and their perfection.

Chordal bipartite graphs have various characterizations in terms of elimination orderings and tree structure properties of related hypergraphs; see for example Theorem 29.36 and [3,4] for more details. They are closely related to strongly chordal graphs.

Theorem 29.36 *A bipartite graph $B = (X, Y, E)$ is $(6, 1)$-chordal (i.e., chordal bipartite) if and only if every induced subgraph of B is X-conformal, Y-conformal and X-chordal, Y-chordal.*

Proof. "$\Longrightarrow$": Let $B = (X, Y, E)$ be bipartite $(6, 1)$-chordal. Then every induced subgraph B' of B is also bipartite $(6, 1)$-chordal.

We first show that B is X- and Y-chordal. If C is a cycle of length at least 8 in B' then C has a chord $\{x, y\}, x \in X, y \in Y$. Let $x_1, x_2 \in X$ be the neighbors of y in C and let $y_1, y_2 \in Y$ be the neighbors of x in C. Let C_1, C_2 denote the subcycles defined by the chord $\{x, y\}$ subdividing C. Without loss of generality, assume $|C_1| \leq |C_2|$. Moreover, assume without loss of generality that x_2, y_2 are in C_2. Then y und y_2 have distance at least 4 in C and x is a neighbor of both vertices. Likewise, x and x_2 have distance at least 4 in C and y is a neighbor of both vertices. Thus, B' is X-chordal and Y-chordal.

Now we show that B is X- and Y-conformal. Let $S \subseteq Y$ be a vertex set with pairwise distance 2 in B'. We show inductively the existence of a vertex $x \in X$ with $S \subseteq N(x)$: For $|S| = 2$ and $|S| = 3$, the assertion is obviously fulfilled (for $|S| = 3$, the existence of a chord in any cycle of length 6 is used).

Now, by induction hypothesis, let the assertion be fulfilled for all S', $|S'| \leq k$, with pairwise distance 2 and let $S \subseteq Y$, $|S| = k + 1$ be a vertex set with pairwise distance 2. Then for every k-elementary subset $S_i \subseteq S$, $i \in \{1, \ldots, \binom{k+1}{k}\}$ (note $\binom{k+1}{k} = k + 1$) there is a vertex x_i for which $S_i \subseteq N(x_i)$. If there is an i with $S \subseteq N(x_i)$ then the assertion is fulfilled. Otherwise, we can assume that the vertices $x_1, \ldots, x_{k+1}$ have exactly one nonneighbor in S: Without loss of generality, let

$$x_i \notin N\left(y_{i+2(\text{mod } k+1)}\right)$$

Now there is a C_6 $(x_1, y_1, x_2, y_3, x_k, y_4)$—contradiction. Thus, there is an index i such that $S \subseteq N(X_i)$. Analogously, one shows Y-conformality of B'.

"$\Longleftarrow$": If every induced subgraph of B is X-conformal, Y-conformal, X-chordal, and Y-chordal then B cannot contain chordless cycles of length at least 6 since chordless cycles of length 6 are neither X- nor Y-conformal and chordless cycles of length at least 8 are neither X- nor Y-chordal. ■

By Corollary 29.9, Theorem 29.36 implies the following.

Corollary 29.16 *A bipartite graph $B = (X, Y, E)$ is $(6, 1)$-chordal (i.e., chordal bipartite) if and only if $\mathcal{N}^X(B)$ and $\mathcal{N}^Y(B)$ are β-acyclic.*

Strongly chordal graphs are closely related to chordal bipartite graphs.

Definition 29.37 *Let $G = (V, E)$ be a graph.*

i. *The* bipartite copy *$B(G) = (V', V'', F)$ of G is defined as follows: For every vertex $v \in V$, there are two copies $v' \in V'$ and $v'' \in V''$, and $x'y'' \in F$ if either $x = y$ or $xy \in E$.*

ii. *$B_C(G)$ denotes the bipartite incidence graph $\mathcal{I}(\mathcal{C}(G))$.*

Note that $B(G)$ is isomorphic to the bipartite incidence graph $\mathcal{I}(\mathcal{N}(G))$. It follows from the basic properties that a graph is chordal bipartite if and only if it is the bipartite incidence graph of a totally balanced hypergraph.

Lemma 29.9 [27] *A graph G is strongly chordal if and only if $B_C(G)$ is chordal bipartite.*

Proof. Lemma 29.9 is an obvious consequence of Theorem 29.36 and Corollary 29.15. ■

A similar connection is given in the following lemma.

Lemma 29.10 [95] *A graph G is strongly chordal if and only if $B(G)$ is chordal bipartite.*

Proof. Lemma 29.10 is an obvious consequence of Corollary 29.13, Corollary 29.11, Corollary 29.10, and Theorem 29.36. ■

For a bipartite graph $B = (X, Y, E)$, let $split_X(B)$ denote the one-sided completion when X becomes a clique. Another relation between chordal bipartite and strongly chordal graphs is the following (see also [59]).

Lemma 29.11 [96] *A bipartite graph $B = (X, Y, E)$ is chordal bipartite if and only if $split_X(B)$ is strongly chordal.* ■

Lemma 29.11 is a simple consequence of the following more general property.

Proposition 29.12 [59] *Let $B = (X, Y, E)$ be a bipartite graph. Then*

i. *$\mathcal{N}^X(B)$ has the Helly property if and only if $\mathcal{C}(split_X(B))$ has the Helly property;*

ii. *$L(\mathcal{N}^X(B))$ is chordal if and only if $L(\mathcal{C}(split_X(B)))$ is chordal.*

Spinrad [19] gives simple direct proofs of Lemmas 29.10 and 29.11 in terms of Γ-free matrices and discusses the relationship between fast recognition of strongly chordal graphs and fast recognition of chordal bipartite graphs; linear time for recognizing chordal bipartite graphs would imply linear time for recognizing strongly chordal graphs but not vice versa (a linear-time algorithm for recognizing chordal bipartite graphs as claimed in [97] turned out to contain a flaw). Linear-time recognition of strongly chordal graphs (chordal bipartite graphs, respectively) is still an open problem. See [98] for other characterizations of chordal bipartite graphs in terms of intersection graphs of compatible subtrees, and [99] for a relationship between dismantlable lattices and chordal bipartite graphs.

29.6 TREE STRUCTURE DECOMPOSITION OF GRAPHS

Various kinds of decomposition of graphs such as modular decomposition and clique separator decomposition lead to decomposition trees and algorithmic applications. In this section, we first describe cographs and modular decomposition (cographs are the completely decomposable graphs with respect to modular decomposition) and then mention some other decompositions, and in particular clique separator decomposition.

29.6.1 Cographs

In this subsection, we describe an auxiliary class, the *cographs*, which occur in many places and which are fundamental for distance-hereditary graphs and for clique-width. See [3] for additional information.

For disjoint vertex sets $A, B \subseteq V$, the *join operation* (denoted by ①) adds edges between all pairs x, y with $x \in A$, $y \in B$, and the *co-join operation* (denoted by ⓪) adds nonedges between all pairs x, y with $x \in A$ and $y \in B$. These notions are closely related to connectedness of a graph and its complement: G is disconnected if and only if G is decomposable into the co-join of two subgraphs, and $\overline{G}$ is disconnected if and only if G is decomposable into the join of two subgraphs. Subsequently we also use ① and ⓪ in order to denote the relationship between disjoint vertex sets.

Definition 29.38 *Graph G is a* cograph (complement-reducible graph) *if G can be constructed from single vertices by a finite number of join and co-join operations.*

See [3,100–102] for properties of this graph class.

Theorem 29.37 *G is a cograph if and only if G is P_4-free.*

Proof. "$\Longrightarrow$": By induction on the number of vertices in G. For single vertices, the assertion is obviously true. Now, let $G = G_1 ① G_2$ and G_1, G_2 being P_4-free. If G would contain a P_4 $P = abcd$ then P has vertices from G_1 and G_2. Assume first that P has exactly one vertex from G_1. If $a \in V(G_1), b, c, d \in V(G_2)$ then $ac \notin E$ contradicts to the join between G_1 and G_2, if $b \in V(G_1), a, c, d \in V(G_2)$ then $bd \notin E$ contradicts to the join between G_1 and G_2. Now assume that P has exactly two vertices from each of G_1, G_2. If $a, d \in V(G_1), b, c \in V(G_2)$ then $ac \notin E$ contradicts to the join between G_1 and G_2, if $b, d \in V(G_1), a, c \in V(G_2)$ then $ad \notin E$ contradicts to the join between G_1 and G_2, and if $a, b \in V(G_1), c, d \in V(G_2)$ then $ac \notin E$ contradicts to the join between G_1 and G_2. In every case, there is a nonedge of the P_4 between G_1 and G_2, and thus $G = G_1 ① G_2$ is again P_4-free.

In the same way one can show that $G = G_1 ⓪ G_2$ is again P_4-free if G_1 and G_2 are P_4-free.

"$\Longleftarrow$": Let G be a P_4-free graph. We will show that then G is decomposable with respect to the operations ①, ⓪ into subgraphs G_1, G_2, that is, either G or $\overline{G}$ is disconnected. Assume that not every P_4-free graph would have this property. Then let $G = (V, E)$ be a smallest P_4-free graph not having this property, that is, G is P_4-free connected and co-connected but for every $v \in V$, either $G - v$ is disconnected or $\overline{G - v}$ is disconnected. Note that in this case, G has at least four vertices.

Case 1 $G - v$ is disconnected. Let $H_1, \ldots, H_k$, $k \geq 2$, be the connected components of $G - v$, that is, there are no edges between H_i and H_j for $i \neq j$, $i, j \in \{1, \ldots, k\}$ but since G is connected, v has edges to each of $H_1, \ldots, H_k$. Let x_i be a neighbor of v in H_i. Since G is also co-connected, v has at least one nonneighbor in $V \setminus \{v\}$. Without loss of generality, let $y \in H_1$ be a vertex with $vy \notin E$. Since H_1 is connected, there is a path $P_{x_1 y}$ between x_1 and y in H_1. Let $x'y'$ be the first edge on this path for which $vx' \in E$ but $vy' \notin E$ holds. Since $vx_1 \in E$, $vy \notin E$, the existence of such an edge is guaranteed. But now the vertices x_2, v, x', y' induce a P_4 in G—contradiction.

Case 2 The case that $\overline{G - v}$ is disconnected can be handled in the same way as the previous case. ■

Theorem 29.37 implies that the property of being a cograph is a hereditary property, that is, if G is a cograph then every induced subgraph G' of G is a cograph as well.

The recursive generation of cographs by the two operations join and co-join is described in a tree structure—the *cotree*. This tree has the vertices of the graph as its leaves, and the internal nodes are labeled with ① and ⓪ according to the operations. If $G = G_1 ① G_2$ ($G = G_1 ⓪ G_2$, respectively) then the root vertex of the cotree of G carries the label ① (⓪, respectively), and its two children are the root nodes of G_1, G_2, respectively.

A cotree is not necessarily a binary tree; for example, a clique with k vertices is represented by one ① node with the k vertices as its children.

In [102], it is described how to recognize in linear time $\mathcal{O}(n + m)$ whether a given input graph G is a cograph; starting with a single vertex, the algorithm tries to incrementally construct a cotree T of G, that is, in every step, a new vertex is added and the new cotree is constructed if the graph is still a cograph; otherwise, an induced P_4 in G is given as output. The algorithm is performed by a complicated marking procedure which cannot be described here. However, it has a remarkable property: It does not only give the correct *Yes/No* answer to the recognition problem; if the answer is *Yes* then the algorithm gives a certificate namely a cotree, and it is easily checkable whether the cotree indeed represents the graph, and if the answer is *No*, it gives a certificate for this answer, that is, in the case of cograph recognition a P_4 in the input graph. Such recognition algorithms are called *certified algorithms* and are known for various graph classes [103].

A simpler recognition of cographs is described in [104] which time bound, however is $\mathcal{O}(n+m\log n)$ (and not linear). In [105], a simple multisweep LexBFS algorithm for recognizing cographs in linear time is given.

29.6.2 Optimization on Cographs

Various algorithmic graph problems being NP-complete in general, can be solved efficiently in a bottom-up procedure along the cotree of a cograph. As examples, we describe this for the problems MAXIMUM STABLE SET and MAXIMUM CLIQUE.

Let $G = (V, E)$ be a graph. A vertex set $U \subseteq V$ is *stable* (*independent*) if for all $x, y \in U$, $xy \notin E$. $U \subseteq V$ is *a clique* if U is stable in $\overline{G}$. If $G = (V, E)$ is a graph with vertex weight function w then for $U \subseteq V$, $w(U) := \Sigma_{x\in U} w(x)$.

Let $\alpha_w G$ be the maximum weight of a stable set in G, and let $\omega_w G$ be the maximum weight of a clique in G. Now, obviously the values of $\alpha_w(G)$ and $\omega_w(G)$ can be computed recursively for $G = G_1 ① G_2$ and $G = G_1 ⓪ G_2$:

- If $G = G_1 ① G_2$ then
$$\omega_w(G) = \omega_w(G_1) + \omega_w(G_2)$$
and
$$\alpha_w(G) = \max(\alpha_w(G_1), \alpha_w(G_2)).$$
- If $G = G_1 ⓪ G_2$ then
$$\omega_w(G) = \max(\omega_w(G_1), \omega_w(G_2))$$
and
$$\alpha_w(G) = \alpha_w(G_1) + \alpha_w(G_2).$$

This implies linear-time algorithms for these two problems on cographs.

As a further example, we show how to color cographs in an optimal way. The coloring problem of a graph is how to assign a minimum number of colors to the vertices such that adjacent vertices get different colors. The *chromatic number* $\chi(G)$ of the graph G is the minimum number of colors needed to color G. Obviously, for every graph G, $\omega(G) \leq \chi(G)$ holds. A graph is called χ*-perfect* if for every induced subgraph G' of G (including G itself), $\omega(G') = \chi(G')$ holds. Let $\kappa(G) = \chi(\overline{G})$. Obviously, $\alpha(G) \leq \kappa(G)$ holds. A graph is called κ*-perfect* if for every induced subgraph G' of G (including G itself), $\alpha(G') = \kappa(G')$ holds. The following theorem is a celebrated result by Laszló Lovász (see, e.g., Chapter 28):

Theorem 29.38 (Perfect graph theorem) *A graph is χ-perfect if and only if it is κ-perfect.*

Now, G is called *perfect* if G is χ-perfect (κ-perfect).

Corollary 29.17 *A graph G is perfect if and only if its complement graph $\overline{G}$ is perfect.*

Corollary 29.18 *Cographs are perfect.*

Proof. We show inductively on the number of vertices that cographs are perfect. For one-vertex graphs, the claim is obviously fulfilled. Now assume first that $G = G_1 ① G_2$ and $\omega(G_i) = \chi(G_i)$ holds for $i \in \{1, 2\}$. Since there is a join between G_1 and G_2, $\chi(G) = \chi(G_1) + \chi(G_2) = \omega(G_1) + \omega(G_2) = \omega(G)$ which shows the claim.

Now assume that $G = G_1 ⓪ G_2$ and $\omega(G_i) = \chi(G_i)$ holds for $i \in \{1, 2\}$. Since there is a co-join between G_1 and G_2, $\chi(G) = \max(\chi(G_1), \chi(G_2)) = \max(\omega(G_1), \omega(G_2)) = \omega(G)$ which again shows the claim. Thus, cographs are perfect. ■

See Chapter 28 for many other important subclasses of perfect graphs.

Another remarkable property of cographs is the fact that they are transitively orientable. Hereby a graph $G = (V, E)$ is called *transitively orientable* if its edge set E can be oriented as E' in such a way that for all oriented edges (x, y), $(y, z) \in E'$, $(x, z) \in E'$ holds. One can easily show by induction that cographs have this property. Hereby, for $G = G_1 ① G_2$, the edges of the join are oriented from G_1 to G_2 – this obviously gives again a transitive orientation if it is assumed that G_1 and G_2 are already transitively oriented—and for a co-join, there is nothing to show.

Subsequently, the modular decomposition of arbitrary graphs is described which generalizes cographs and cotrees and gives a strong algorithmic tool for many problems. See [106] for the connection between transitive orientation, cographs and modular decomposition.

29.6.3 Basic Module Properties

Let $G = (V, E)$ be a graph. A vertex set $M \subseteq V$ is a *module* in G if its vertices are indistinguishable from outside M. More formally: For all $u \in V \setminus M$, either $\{u\} ⓪ M$ or $\{u\} ① M$. Sets A and B *overlap* if $A \setminus B \neq \emptyset$, $B \setminus A \neq \emptyset$, and $A \cap B \neq \emptyset$.

Theorem 29.39 (Basic module properties) *Let G be a graph and let $\mathcal{M}(G)$ denote the set of modules in G. Then the following properties hold:*

i. *$\emptyset$, V and $\{v\}$ for all $v \in V$ are modules (the* trivial modules*);*

ii. *If $M_1, M_2 \in \mathcal{M}(G)$ then $M_1 \cap M_2 \in \mathcal{M}(G)$;*

iii. *If $M_1, M_2 \in \mathcal{M}(G)$ and $M_1 \cap M_2 \neq \emptyset$ then $M_1 \cup M_2 \in \mathcal{M}(G)$;*

iv. *If M_1 and M_2 are overlapping modules then $M_1 \setminus M_2 \in \mathcal{M}(G)$, $M_2 \setminus M_1 \in \mathcal{M}(G)$, $(M_1 \setminus M_2) \cup (M_2 \setminus M_1) \in \mathcal{M}(G)$;*

v. *If M is a module in G and $U \subseteq V$ then $M \cap U$ is a module in $G[U]$.*

Proof.

i. This property is obviously fulfilled.

ii. Let M_1 and M_2 be modules in G. If their intersection is empty then due to (i), the assertion is fulfilled. Now assume that $M_1 \cap M_2 \neq \emptyset$. If $M_1 \subseteq M_2$ or $M_2 \subseteq M_1$ then again the assertion holds true. Now assume that M_1 and M_2 are overlapping modules. Vertices outside $M_1 \cap M_2$ cannot distinguish two vertices from $M_1 \cap M_2$: if a vertex $x \notin M_1 \cup M_2$ would distinguish vertices $a, a' \in M_1 \cap M_2$, that is, $xa \in E$, $xa' \notin E$ then this would contradict to the module property of M_1 (M_2, respectively); if a vertex $x \in M_1 \setminus M_2$ would distinguish vertices $a, a' \in M_1 \cap M_2$, that is, $xa \in E$, $xa' \notin E$ then this would contradict the module property of M_2, and the same holds for $x \in M_2$.

iii. Let $M_1 \cap M_2 \neq \emptyset$ with $a \in M_1 \cap M_2$. If $M_1 \subseteq M_2$ or vice versa then the assertion is trivial. Now assume that M_1 and M_2 are overlapping modules. Due to condition (ii), vertices in $M_1 \cap M_2$ cannot be distinguished from outside. The same holds for two vertices in M_1 (M_2, respectively). Now assume that vertices $a' \in M_1 \setminus M_2$ and $a'' \in M_2 \setminus M_1$ could be distinguished by $x \notin M_1 \cup M_2$: $xa' \in E$ and $xa'' \notin E$. Since $xa' \in E$ and $a, a' \in M_1$, also $xa \in E$ holds but since $a, a'' \in M_2$, it follows that $xa'' \in E$—contradiction. Thus $M_1 \cup M_2$ is a module.

iv. We first show that $M_1 \setminus M_2$ is a module. Assume to the contrary that there are vertices $a, a' \in M_1 \setminus M_2$ and $x \notin M_1 \setminus M_2$ such that $ax \in E$, $a'x \notin E$. Then $x \in M_1$ since M_1 is a module, that is, $x \in M_1 \cap M_2$. Let $b \in M_2$. Since $x, b \in M_2$ and M_2 is a module, also $ab \in E$ and $a'b \notin E$ holds but now $b \notin M_1$ is a vertex outside M_1 distinguishing vertices $a, a' \in M_1$—contradiction. Analogously, M_2 is a module.

Now we show that $\Delta := (M_1 \setminus M_2) \cup (M_2 \setminus M_1)$ is a module: Let $a, a' \in \Delta$. Since $M_1 \setminus M_2$ $(M_2 \setminus M_1)$ is a module, we can assume that $a \in M_1 \setminus M_2$ and $a' \in M_2 \setminus M_1$. Due to (iii), $M_1 \cup M_2$ is a module. Thus, a and a' cannot be distinguished from outside $M_1 \cup M_2$. Assume that there is a vertex $x \notin \Delta$, $x \in M_1 \cap M_2$ such that $xa \in E$, $xa' \notin E$. Since $x, a \in M_1$, $a' \notin M_1$ and M_1 a module, $aa' \notin E$ holds. Since $x, a' \in M_2$, $a \notin M_2$ and M_2 a module, $aa' \in E$ holds—contradiction.

v. If $M \subseteq U$ then the assertion is obviously fulfilled. Assume now that $M \setminus U \neq \emptyset$. If $M \cap U = \emptyset$ then again the assertion is obviously fulfilled. Now assume that $M \cap U \neq \emptyset$ and $M \cap U$ is no module in $G[U]$, that is, there are vertices $a, a' \in M \cap U$ and a vertex $v \in U \setminus M$ distinguishing a and a' from outside M but then M is no module—contradiction. ■

Theorem 29.40 *In a connected and co-connected graph G, the nontrivial $\subseteq$-maximal modules are pairwise disjoint.*

Proof. Let M_1 and M_2 be nontrivial modules in G being maximal with respect to set inclusion and assume that $M_1 \cap M_2 \neq \emptyset$. This implies that they are overlapping modules. Then according to Theorem 29.39 (iii), $M_1 \cup M_2$ is a module. If $M_1 \cup M_2 \neq V$ then M_1 and M_2 are not maximal—thus $M_1 \cup M_2 = V$. Note that vertices from $M_1 \setminus M_2$ are either completely adjacent to M_2 or completely nonadjacent to M_2, and the same holds for vertices from $M_2 \setminus M_1$. Let $M_1^+ := \{x : x \in M_1 \setminus M_2$ and x has a join to $M_2\}$, $M_1^- := \{x : x \in M_1 \setminus M_2$ and x has a cojoin to M_2, and define the sets M_2^+ and M_2^- in a completely analogous way. Obviously, $M_1 \setminus M_2 = M_1^+ \cup M_1^-$ and $M_2 \setminus M_1 = M_2^+ \cup M_2^-$. If one of the sets M_1^+, M_1^-, M_2^+, and M_2^- is empty then G is not connected or not co-connected. Thus, all of these sets are nonempty. Now let $x \in M_1^+$, $x' \in M_1^-$ and $y \in M_2^+$. The fact that $xy \in E$ and M_1 is a module implies that $x'y \in E$ but now x' is adjacent to a vertex from M_2—contradiction. ■

A graph is *prime* if it contains no nontrivial module. The *characteristic graph* G^* of G is the graph obtained by contracting the maximal modules of G to one vertex.

Theorem 29.41 *The characteristic graph G^* of a connected and co-connected graph G is prime.*

Proof. By Theorem 29.40, the maximal nontrivial modules in $G = (V, E)$ are pairwise disjoint and thus define a partition of V into equivalence classes. Let v^* denote the equivalence class of a vertex v. Let $G^* = (V^*, E^*)$ and $U \subseteq V^*$ and denote by K_x the equivalence class in V belonging to $x \in V^*$. Then the *expansion* $E(U)$ of U is the union of the equivalence classes belonging to U, that is, the vertex set $E(U) = \bigcup_{x \in U} K_x$. We first claim that for a module M in G^*, its expansion $E(M)$ is a module in G. Assume to the contrary that there are $a, b \in E(M)$ and $x \notin E(M)$ such that $ax \in E$ and $bx \notin E$. Then obviously, a and b are not in the same class in $E(M)$ since the classes are modules. This means that $a^* \neq b^*$, $a^*, b^* \in M$ and $x^* \notin M$ but now M is no module—contradiction. This shows the claim.

Now assume that M is a nontrivial module in G^*. If M consists only of vertices whose classes are one-elementary then $E(M) = M$ and M is a module in G; thus, after shrinking the modules in G, M cannot have more than one element. If M contains at least one vertex u whose class U is a nontrivial module in G then $U \subset E(M)$ but U is a maximal module in G and $E(M)$ is a module in G—contradiction. Thus, G^* is a prime graph. ■

29.6.4 Modular Decomposition of Graphs

Theorems 29.39 and 29.40 lead to the following tree structure of a given graph G: Every vertex in G is contained in a unique (possibly one-elementary) maximal module different from V, and these modules define a partition of V. The *modular decomposition tree* has V as its root and the maximal modules smaller than V are the children of V in the tree. Then the children of an inner vertex M are the maximal modules in $G[M]$ smaller than M. Thus, if the inner vertex M of the modular decomposition tree has the partition $M_1, M_2, \ldots, M_k$ into its maximal modules then $M_1, M_2, \ldots, M_k$ are the children of M. Note that the leaf descendants of M are the vertices of M, and the edges in M between M_i and M_j are given by a sequence of join and co-join operations between the modules M_i and the vertices outside M_i. The graphs being completely decomposable by join and co-join are the cographs.

The following decomposition theorem is implicitly contained in the seminal paper by Tibor Gallai [107].

Theorem 29.42 (Modular decomposition theorem) *Let $G = (V, E)$ be an arbitrary graph. Then precisely one of the following conditions is satisfied:*

1. *G is disconnected (i.e., decomposable by the co-join operation);*
2. *$\overline{G}$ is disconnected (i.e., decomposable by the join operation);*
3. *G and $\overline{G}$ are connected: There is some $U \subseteq V$ and a unique partition $\mathcal{P}$ of V such that*
   a. *$|U| \geq 4$,*
   b. *$G[U]$ is a maximal prime subgraph of G, and*
   c. *for every class S of the partition $\mathcal{P}$, S is a module and $|S \cap U| = 1$.*

Each vertex of G forms a leaf of the decomposition tree. Each module M of G occuring as a node in the tree contains exactly the vertices that are leaves of the subtree rooted at M. According to the Decomposition Theorem, the tree has three kinds of nodes:

- Parallel nodes (co-join operation);
- Series nodes (join operation);
- Prime nodes.

Linear-time algorithms for finding the modular decomposition tree are given in [108,109] and in [106]. See [110,111] for simpler linear-time algorithms.

The modular decomposition is of crucial importance in many algorithmic applications; see [112] for many aspects of modular decomposition. Since for many algorithmic problems the operations join and co-join are easy to handle (cf. the case of cographs), it is important to look at prime graphs. There are some cases where prime graphs have *simple structure*.

A nice example for a graph class having simple prime graphs with respect to modular decomposition are P_4-sparse graphs.

A graph $G = (V, E)$ is *P_4-sparse* [113] if every five vertices induce at most one P_4 in G. Thus, cographs are P_4-sparse, and the only one-vertex extensions of a P_4 in a P_4-sparse graph G are the bull, gem and co-gem, that is, G is P_4-sparse if and only if all the other seven one-vertex extension (such as P_5, C_5, etc.) are forbidden induced subgraphs in G. Obviously, the complement of a P_4-sparse graph is P_4-sparse.

A graph is a *thin spider* if its vertex set can be partitioned into a clique Q and a stable set S such that the edges between Q and S form a matching, every vertex in S has exactly

one neighbor in Q, and at most one vertex in Q has no neighbor in S (the *head of the spider*). Obviously, thin spiders are prime graphs and P_4-sparse. A graph is a *thick spider* if it is the complement of a thin spider; it is a *spider* if it is a thin or thick spider (these graphs were called *turtles* in [113]).

Theorem 29.43 [113] *A graph is P_4-sparse if and only if its prime graphs are spiders.*

Various structural and algorithmic consequences are given in [113–117].

A lot of research has been done in generalizing, refining and modifying modular decomposition. *Split* (or *join*) *decomposition* was introduced and studied by Cunningham [118]. A graph is *split decomposable* if its vertex set has a partition into A_1, A_2 and B_1, B_2 such that $A = A_1 \cup A_2, B = B_1 \cup B_2$, and the set of all edges between A and B forms a join $A_1 ① B_1$. The decomposition is discussed in detail in the monograph [19] by Spinrad, mentioning the linear-time algorithm for split decomposition by Dahlhaus [119]. A simplified linear-time algorithm for split decomposition is given in [120].

The class of graphs such that every induced subgraph on at least four vertices is decomposable by the join decomposition is of particular interest. It turns out that these are exactly the distance-hereditary graphs which are the central topic of the next section.

Another interesting concept is the homogeneous decomposition where a third operation is added which is a combination of join and co-join. This approach is based on a different kind of connectedness—the *p-connectedness*—and is described in [121].

29.6.5 Clique Separator Decomposition of Graphs

A *clique separator* of a graph G is a separator of G which is a clique in G. For a chordal graph G which is not a clique and a simplicial vertex v in G, obviously $N(v)$ is a clique separator of G. Clique separator decomposition of a graph is generalizing chordal graphs by repeatedly choosing a clique separator in G until there is no longer a clique separator in the resulting subgraphs; such subgraphs are called *atoms* of G. Note that such decomposition trees are not uniquely determined. Obviously, chordal graphs are those graphs whose atoms are cliques. This kind of decomposition was introduced in [122,123] and has a number of algorithmic applications described in [122] among them efficiently solving the MWIS problem on a graph class whenever it is efficiently solvable on the atoms of the class. This refers to the weight modification approach described in the algorithm of Frank for the same problem on chordal graphs—see Theorem 29.4.

Various examples of such classes were studied: In [124], a subclass of hole-free graphs, namely hole- and paraglider-free graphs, is characterized by the structure of their atoms. Among others, this is motivated by a result of Alekseev [125] showing that atoms of (P_5,paraglider)-free graphs are $3K_2$-free which implies polynomial time for MWIS on this class. For P_5-free graphs, the complexity of the MWIS problem was open for a long time; meanwhile, it has been shown by Lokshtanov et al. [126] that it is polynomially solvable for P_5-free graphs. For hole-free graphs, the complexity of the MWIS problem is open.

29.7 DISTANCE-HEREDITARY GRAPHS, SUBCLASSES, AND γ-ACYCLICITY

29.7.1 Distance-Hereditary Graphs

Distance-hereditary graphs are another fundamental generalization of trees. They are closely related to γ-acyclic hypergraphs (see Definition 29.44) and have bounded clique-width. Originally, they were defined via a distance property.

Definition 29.39 [127] *A graph G is* distance hereditary *if for each connected induced subgraph F of G, the distance functions d_G in G and d_F in F coincide.*

Definition 29.40 *A u-v-*geodesic *is a u-v-path α such that $l(\alpha) = d_G(u, v)$. Let Φ be a cycle of G. A path α is an* essential part of Φ *if $\alpha \subset \Phi$ and $1/2l(\Phi) < l(\alpha)$.*

Theorem 29.44 [127] *The following conditions are equivalent:*

i. *G is distance hereditary.*

ii. *Every induced path of G is geodesic.*

iii. *No essential part of a cycle of G is induced.*

iv. *Each cycle of G of length ≥ 5 has at least two chords, and each 5-cycle of G has a pair of crossing chords.*

v. *Each cycle of G of length ≥ 5 has a pair of crossing chords.*

Proof. Howorka [127] has shown that (i) $\Longleftrightarrow$ (ii) $\Longleftrightarrow$ (iii) $\Longrightarrow$ (iv) $\Longrightarrow$ (v) $\Longrightarrow$ (iii); here, we give his proof.

(i) $\Longrightarrow$ (ii): Let α be an induced path of a distance-hereditary graph G and let u and v be the endpoints of α. Then $d_G(u, v) = d_\alpha(u, v) = l(\alpha)$. Hence α is a geodesic.

(ii) $\Longrightarrow$ (i): Suppose that F is a connected induced subgraph of G. Let u, v be arbitrary vertices of F, and let α be a u-v-geodesic of F. Thus naturally, α is an induced path of F and, consequently, also an induced path of G. Hence, by assumption, α is a u-v-geodesic of G. Thus $d_F(u, v) = l(\alpha) = d_G(u, v)$. This proves that G is distance hereditary.

(ii) $\Longrightarrow$ (iii): Since an essential part of a cycle cannot be a geodesic, (ii) clearly implies (iii).

(iii) $\Longrightarrow$ (ii): Let G be a graph satisfying (iii). Let $u \neq v$ be vertices of G and assume that $\alpha = (u = a_0, a_1, \ldots, a_m = v)$ is a u–v-path of G which is not a geodesic. Consider any u-v-geodesic $\beta = (u = b_0, b_1, \ldots, b_n = v)$, $n < m$. Let i be the largest index for which $b_i = a_i$, $0 \leq i < n$. Let t be the least index $> i$ for which $b_t \in \alpha$. Thus $b_t = a_j$ for some $j > t$. Consequently, the path $\delta = (a_i, a_{i+1}, \ldots, a_j)$ is an essential part of the cycle $(a_i, a_{i+1}, \ldots, a_j = b_t, b_{t-1}, \ldots, b_i = a_i)$. By assumption, δ is not induced. Hence α is not induced. This completes the proof.

(iii) $\Longrightarrow$ (iv): Let $\Phi = (a_0, a_1, \ldots, a_n = a_0)$, $n \geq 5$, be a cycle of a graph G satisfying (iii). By considering any essential part of Φ of length $\leq n - 2$, we see from (iii) that Φ must have at least one chord, say $a_i a_j$. Since, in turn, $(a_{i+1}, a_{i+2}, \ldots, a_{i-1})$ is an essential part of Φ, then Φ must have a chord distinct from $a_i a_j$. This proves that each cycle of G of length ≥ 5 has at least two chords. An easy verification shows that if (iii) holds then a 5-cycle of G must have a pair of crossing chords.

(iv) $\Longrightarrow$ (v): Assume that G satisfies (iv). We will prove by induction that each n-cycle of G, $n \geq 5$, has a pair of crossing chords. By assumption, the assertion is true for $n = 5$. Let $n > 5$ and suppose that each cycle of length m, $5 \leq m < n$, has a pair of crossing chords. Consider an n-cycle $\Phi = (a_0, a_1, \ldots, a_n = a_0)$ and let $a_i a_j$ and $a_r a_s$ be two distinct chords of Φ. If they do not cross one another, we may assume without loss of generality that $0 \leq i \leq j \leq r \leq s \leq n$. Consider the cycles $(a_i, a_j, a_{j+1}, \ldots, a_i)$ and $(a_r, a_s, a_{s+1}, \ldots, a_r)$. Since $n \geq 6$, at least one of them has length ≥ 5 and hence, by induction hypothesis, it must have a pair of crossing chords. This same pair is, of course, a pair of crossing chords of Φ. This completes the proof.

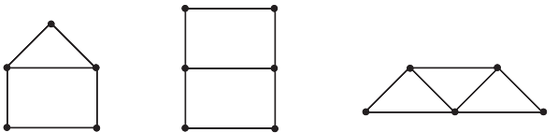

Figure 29.5 House (a), domino (b), and gem (c) are not distance-hereditary.

(v) $\Longrightarrow$ (iii): Let G be a graph satisfying (v). We will prove by induction on n that no essential part of an n-cycle of G is induced. This is trivially true if $n = 3$ or $n = 4$. Assume that $n > 4$ and that the assertion is true for all cycles of length $< n$. Let $\Phi = (a_0, a_1, \ldots, a_n = a_0)$ be a cycle of G and let α be an essential part of Φ, say $\alpha = (a_0, a_1, \ldots, a_k)$, where $n/2 < k < n$. Let a_ia_j and a_ra_s be a pair of crossing chords of Φ. Without loss of generality we may assume that $0 \leq i < j < n$, $0 \leq r < s < n$ and $i < r$. If $j \leq k$ then a_ia_j joins two vertices of α; hence α is not induced. If $i \geq k$ then α is an essential part of the cycle $(a_0, a_1, \ldots, a_k, \ldots, a_i, a_j, \ldots, a_n = a_0)$ of length $< n$. Hence, by induction hypothesis, α is not induced. We may assume therefore that $0 \leq i < k < j < n$. Applying the same argument to a_ra_s, we obtain $0 \leq r < k < s < n$. Since the chords a_ia_j and a_ra_s cross one another, it follows that $0 \leq i < r < k < j < s < n$. Denote $\alpha' = (a_0, a_1, \ldots, a_r)$ and $\alpha'' = (a_i, a_{i+1}, \ldots, a_k)$. We claim that either α' is an essential part of the cycle $(a_s, a_{s+1}, \ldots, a_r, a_s)$ or α'' is an essential part of the cycle $(a_i, a_{i+1}, \ldots, a_j, a_i)$. We have indeed: $l(\alpha') + l(\alpha'') \geq k + 1 \geq n - k + 2 > n - s + j - k + 2 = (n - s + 1) + (j - k + 1)$ and so, either $l(\alpha') > n - s + 1$, or $l(\alpha'') > j - k + 1$, which proves our claim. It follows now from an induction hypothesis that either α' or α'' is not an induced path. Hence α cannot be induced. This completes the inductive step and proves the theorem. ■

Chordless cycles with at least five vertices are called *holes*. Obviously, holes are not distance hereditary. Recall that, in connection with relational database schemes, (k, l)-chordal graphs were defined (see Definition 29.5) (Figure 29.5).

Theorem 29.45 *Let G be a graph.*

i. *G is $(5, 2)$-chordal if and only if G is (house, hole, domino)-free.*

ii. *G is distance-hereditary if and only if G is (house, hole, domino, gem)-free* [128].

Proof. (i): Obviously, every $(5, 2)$-chordal graph is (house, hole, domino)-free. For the other direction, let G be (house, hole, domino)-free, and let $C = (x_1, \ldots, x_k)$, $k \geq 5$, be a cycle in G. If $k = 5$ then C is no C_5 and no house, that is, C must have at least two chords. If $k = 6$ then C is no C_6 and no domino, and since G is C_5-free, C must have at least two chords. If $k \geq 7$ then C has a chord x_ix_j since G is hole-free. A cycle C' consisting of an essential part of C together with the chord x_ix_j has length at least 5 and thus has another chord (since G is hole-free) which shows the assertion.

(ii): Obviously, every distance-hereditary graph is (house, hole, domino, gem)-free. For the other direction, let G be (house, hole, domino, gem)-free, and let $C = (x_1, \ldots, x_k)$, $k \geq 5$, be a cycle in G. By Theorem 29.44, (v), it is sufficient to show that C has two crossing chords. If $k = 5$ then C is no C_5, no house and no gem, that is, C must have two crossing chords. If $k = 6$ then C is no C_6 and no domino, and since G is C_5- and gem-free, C must have two crossing chords. If $k \geq 7$ then C has a chord x_ix_j since G is hole-free. A cycle C' consisting of an essential part of C together with the chord x_ix_j has length at least 5 and thus, by an induction hypothesis, has two crossing chords which shows the assertion. ■

For most of the algorithmic applications, a characterization of distance-hereditary graphs in terms of three simple operations is crucial which is described in the next theorem:

Theorem 29.46 [128] *A connected graph G is distance-hereditary if and only if G can be generated from a single vertex by repeatedly adding a pendant vertex, a false twin or a true twin.*

Proof. Assume first that graph G can be generated from a single vertex by repeatedly adding a pendant vertex, a false twin or a true twin. Then it can easily be seen that G must be (house, hole, domino, gem)-free.

For the other direction, we give the short proof of Theorem 29.46 contained in [129]. Actually, [128] is claiming more namely that every distance-hereditary graph with at least two vertices contains either a pair of twins or two pendant vertices. In [130], an even slightly stronger version is given (and an incorrectness of the proof in [128] is corrected).

Let G be a distance-hereditary graph, thus having crossing chords in each cycle of length at least 5. It suffices to show that G has a pendant vertex or a pair of twins since every induced subgraph of G is again distance hereditary. This is trivially fulfilled if G is a disjoint union of cliques. We may assume that some component H of G is not a clique. Let Q be a minimal cutset of H and $R_1, \ldots, R_m$ be the components of $H - Q$. Suppose that $|Q| \geq 2$; we show that Q is a homogeneous set. If not, there are two vertices $p, q \in Q$ and a vertex $r \in V(H) - Q$ with $rp \in E$ and $rq \notin E$. Let $r \in R_1$. Since Q is a minimal cutset of H, vertex q has a neighbor $s \in R_1$. Note that there is an r-s-path P_1 in R_1. We choose s so that P_1 is as short as possible. Similarly p has a neighbor $t \in R_2$ and q has a neighbor $u \in R_2$. We choose t and u so that a shortest t-u-path P_2 in R_2 has smallest length (possibly $t = u$). The vertices s, q, u, t, p, r and the paths P_1 and P_2 form a cycle C of length at least 5. The only possible chords of C join p to q or to some vertices of P_1. Thus, C has no crossing chords, a contradiction.

Now if x is any vertex in R_1 which is adjacent to Q, it must be adjacent to all vertices of Q and thus Q is P_4-free (otherwise, G has a gem). We know that a nontrivial P_4-free graph has a pair of twins. They will also be twins in G because Q is homogeneous.

Now suppose that every minimal cutset contains only one vertex. Let R be a terminal block of H, that is, a maximal 2-connected subgraph of H that contains just one cut-vertex, say x, of H. If $|R| = 2$, the vertex in $R - x$ is a pendant vertex of G. If $|R| \geq 3$ and $R - x \subseteq N(x)$, the set $R - x$ must induce a P_4-free subgraph. So R contains a pair of twins, and clearly they are also twins in G.

If $R \setminus N(x) \neq \emptyset$, $N(x) \cap R$ is a cutset of H and so it contains a minimal cutset of size one but then R is not 2-connected, a contradiction which proves the theorem. ■

For a distance-hereditary graph G, a *pruning sequence* of G describes how G can be generated (dismantled, respectively) by repeatedly adding (deleting, respectively) a pendant vertex, a false twin or a true twin. Pruning sequences and pruning trees are a fundamental tool for most of the efficient algorithms on distance-hereditary graphs. There is a more general way, however, to efficiently solve problems on graph classes captured in the notion of clique-width described in the section on clique-width.

Definition 29.41 *Let G be a graph with vertices $v_1, \ldots, v_n$, and let $S = (s_2, \ldots, s_n)$ be a sequence of tuples of the form $((v_i, v_j), type)$, where $j < i$ and $type \in \{leaf, true, false\}$. S is a* pruning sequence *for G, if for all i, $2 \leq i \leq n$, the subgraph of G induced by $\{v_1, \ldots, v_i\}$ is obtained from the subgraph of G induced by $\{v_1, \ldots, v_{i-1}\}$ by adding vertex v_i and making it adjacent only to v_j if $type = leaf$, making it a true twin of v_j if $type = true$, and making it a false twin of v_j if $type = false$.*

By Theorem 29.46, a graph is distance hereditary if and only if it has a pruning sequence.

Definition 29.42 *Let G be a graph with vertices $v_1, \ldots, v_n$, and let $S = (s_2, \ldots, s_n)$ be a pruning sequence for G. The* pruning tree *corresponding to S is the labeled ordered tree T constructed as follows:*

1. *Set T_1 as the tree consisting of a single root vertex v_1, and set $i := 1$.*
2. *Set $i := i + 1$. If $i > n$ then set $T := T_n$ and stop.*
3. *Let $s_i = ((v_i, v_j), leaf)$ (respectively, $s_i = ((v_i, v_j), true)$, or $s_i = ((v_i, v_j), false)$), then set T_i as the tree obtained from T_{i-1} by adding the new vertex v_i and making it a rightmost son of the vertex v_j, and labeling the edge connecting v_i to v_j by* leaf (*respectively by* true *or* false).
4. *Go back to step* (2) *above.*

A linear-time recognition algorithm for distance-hereditary graphs using pruning sequences was claimed already in [129]; however, their algorithm contained a flaw. Damiand et al. [104] used the following characterization given by Bandelt and Mulder for linear-time recognition of distance-hereditary graphs.

Theorem 29.47 [128] *Let G be a connected graph and $L_1, \ldots, L_k$ be the distance levels of a hanging from an arbitrary vertex v of G. Then G is a distance-hereditary graph if and only if the following conditions hold for any $i \in \{1, \ldots, k\}$:*

i. *If x and y belong to the same connected component of $G[L_i]$ then $L_{i-1} \cap N(x) = L_{i-1} \cap N(y)$.*

ii. *$G[L_i]$ is a cograph.*

iii. *If $u \in L_i$ and vertices x and y from $L_{i-1} \cap N(u)$ are in different connected components X and Y of $G[L_{i-1}]$ then $X \cup Y \subseteq N(u)$ and $L_{i-2} \cap N(x) = L_{i-2} \cap N(y)$.*

iv. *If x and y are in different connected components of $G[L_i]$ then sets $L_{i-1} \cap N(x)$ and $L_{i-1} \cap N(y)$ are either disjoint or comparable with respect to set inclusion.*

v. *If $u \in L_i$ and vertices x and y from $L_{i-1} \cap N(u)$ are in the same connected component C of $G[L_{i-1}]$ then the vertices of C which are nonadjacent to u are either adjacent to both x and y or to none of them.*

The next theorem gives yet another characterization of distance-hereditary graphs. It will be used in the following subsection.

Theorem 29.48 [128,131] *For a graph G, the following conditions are equivalent:*

1. *G is distance-hereditary,*
2. *For each vertex v of G and every pair of vertices $x, y \in L_i(v)$, that are in the same connected component of the graph $G[V \setminus L_{i-1}(v)]$, we have*

$$N(x) \cap L_{i-1}(v) = N(y) \cap L_{i-1}(v).$$

Here, $L_1(v), \ldots, L_k(v)$ are the distance levels of a hanging from vertex v of G.

For many other graph classes defined in terms of metric properties in graphs, related convexity properties and connections to geometry, see the recent survey by Bandelt and Chepoi [132].

29.7.2 Minimum Cardinality Steiner Tree Problem in Distance-Hereditary Graphs

For a given graph $G = (V, E)$ and a set $S \subseteq V$ (of target vertices), a *Steiner tree* $T(S, G)$ is a tree with the vertex set $S \cup S'$ (i.e., $T(S, G)$ spans all vertices of S) and the edge set E' such that $S' \subseteq V$ and $E' \subseteq E$. The *minimum cardinality Steiner tree problem* asks for a Steiner tree with minimum $|S \cup S'|$.

An $O(|V||E|)$ time algorithm for the minimum cardinality Steiner tree problem on distance-hereditary graphs was presented in [131]. Later, in [133], a linear-time algorithm was obtained as a consequence of a linear-time algorithm for the connected r-domination problem on distance-hereditary graphs. Here, we present a direct linear-time algorithm for the minimum cardinality Steiner tree problem.

Algorithm ST-DHG (Find a minimum cardinality Steiner tree in a distance-hereditary graph)

Input: A distance-hereditary graph $G = (V, E)$ and a set $S \subseteq V$ of target vertices.
Output: A minimum cardinality Steiner tree $T(S, G)$.

begin
pick an arbitrary vertex $s \in S$ and build in G the distance levels $L_1(s), \ldots, L_k(s)$
of a hanging from vertex s;
for $i = k, k-1, \ldots, 2$ **do**
if $S \cap L_i(s) \neq \emptyset$ **then**
find the connected components $A_1, A_2, \ldots, A_p$ of $G[L_i(s)]$;
in each component A_j pick an arbitrary vertex x_j;
order these components in nondecreasing order with respect to $d'(A_j) = |N(x_j) \cap L_{i-1}(s)|$;
for all components A_j taken in nondecreasing order with respect to $d'(A_j)$ **do**
set $B := N(x_j) \cap L_{i-1}(s)$;
if ($S \cap A_j \neq \emptyset$ and $S \cap B = \emptyset$) **then**
add an arbitrary vertex y from B to set S;
$T(S, G) :=$ a spanning tree of a subgraph $G[S]$ of G induced by vertices S;
end

Clearly, this is a linear-time algorithm. The correctness proof is based on Theorem 29.47, Theorem 29.48 and the following claims.

Let $G = (V, E)$ be a distance-hereditary graph, $S \subseteq V$ be a set of target vertices, and $s \in S$ be an arbitrary vertex from S.

Claim 29.9 *There exists a minimum cardinality Steiner tree $T(S, G)$ such that $d_{T(S,G)}(x, s) = d_G(x, s)$ for any vertex x of $T(S, G)$.*

Proof. Let $L_1(s), \ldots, L_k(s)$ be the distance levels of a hanging of G from vertex $s \in S$. It is enough to show that there exists a minimum cardinality Steiner tree $T(S, G)$ such that if $T(S, G)$ is rooted at s then for any vertex x of $T(S, G)$ the following property holds:

(P^*) if x belongs to $L_i(s)$ ($i \in \{1, \ldots, k\}$) then its parent x^* in $T(S, G)$ belongs to $L_{i-1}(s)$.

Let $T(S, G)$ be a minimum cardinality Steiner tree with maximum number of vertices satisfying property (P^*) and let x be a vertex of $T(S, G)$ not satisfying (P^*) and with maximum $d_G(x, s)$. Assume x belongs to $L_i(s)$. Consider the (x, s)-path $P(x, s)$ in $T(S, G)$ and let

$y \in P(x,s)$ be the vertex closest to x in $P(x,s)$ with $y \in L_i(s)$ and $y^* \in L_{i-1}(s)$, where y^* is the parent of y in $T(S,G)$. From choices of vertices x and y, we conclude that the subpath of $P(x,s)$ between vertices x and y lays entirely in $L_i(s) \cup L_{i+1}(s)$. By Theorem 29.48, vertices x and y^* must be adjacent in G. Hence, we can modify tree $T(S,G)$ by removing edge xx^* and adding edge xy^*. The new tree obtained spans all vertices of S and has the same vertex-set. Since $T(S,G)$ was chosen to have maximum number of vertices satisfying property (P^*), such a vertex $x \in L_i(s)$ with $x^* \notin L_{i-1}(s)$ cannot exist, proving the claim. ■

Let $A_1, A_2, \ldots, A_p$ be the connected components of $G[L_i(s)]$. By Theorem 29.48, $N(x) \cap L_{i-1}(s) = N(y) \cap L_{i-1}(s)$ for every pair of vertices $x, y \in A_j$, $j \in \{1, \ldots, p\}$. Hence, $N(A_j) \cap L_{i-1}(s) = N(x_j) \cap L_{i-1}(s)$ for any vertex $x_j \in A_j$. Denote $d'(A_j) := |N(A_j) \cap L_{i-1}(s)|$. Assume, without loss of generality, that $d'(A_1) \leq d'(A_2) \leq \cdots \leq d'(A_p)$. Let $B_j := N(u) \cap L_{i-1}(s) = N(A_j) \cap L_{i-1}(s)$, where u is an arbitrary vertex of A_j.

Claim 29.10 *For any vertices $x, y \in B_j$, $N(x) \setminus (B_j \cup A_1 \cup \cdots \cup A_{j-1}) = N(y) \setminus (B_j \cup A_1 \cup \cdots \cup A_{j-1})$.*

Proof. We have $u \in L_i(s)$, $x, y \in L_{i-1}(s) \cap N(u)$ and every vertex of A_j is adjacent to both x and y. By Theorem 29.48, any vertex $z \in L_{i-2}(s)$ either adjacent to both x and y or to none of them. Since $d'(A_j) \leq d'(A_{j'})$ for $j' > j$, by Theorem 29.47(iv), any vertex from $A_{j+1} \cup \cdots \cup A_p = L_i(s) \setminus (A_1 \cup \cdots \cup A_j)$ is adjacent to both or neither one of x and y. Assume now that there is a vertex $z \in L_{i-1}(s) \setminus B_j$ which is adjacent to x but not to y. Since path (z, x, u, y) lays in $L_i(s) \cup L_{i-1}(s)$, by Theorem 29.48, there must exist a vertex w in $L_{i-2}(s)$ adjacent to all y, x, z. But then, it is easy to see that the vertices u, x, y, z, w induce either a house or a gem in G, which is impossible. ■

Let now i be the largest number such that $L_i(s) \cap S \neq \emptyset$ and, as before, $A_1, A_2, \ldots, A_p$ be the connected components of $G[L_i(s)]$ with $d'(A_1) \leq d'(A_2) \leq \cdots \leq d'(A_p)$. Let also j be the smallest number such that $A_j \cap S \neq \emptyset$. Set $B := N(A_j) \cap L_{i-1}(s)$. We know that any vertex of $A_j \cap S$ is adjacent to all vertices of B.

Claim 29.11 *Let $S \cap B \neq \emptyset$, $x \in S \cap A_j$ and $y \in S \cap B$. T' is a minimum cardinality Steiner tree of G for target set $S \setminus \{x\}$ if and only if T, obtained from T' by adding vertex x and edge xy, is a minimum cardinality Steiner tree of G for target set S.*

Proof. By Claim 29.9, for G and target set S, there exists a minimum cardinality Steiner tree T where vertex x is a leaf and its neighbor x^* in T belongs to $L_{i-1}(s)$, that is, to B. If $x^* \neq y$, we can get a new minimum cardinality Steiner tree for G and target set S by replacing edge xx^* in T with edge xy. We can do that since vertex y is in T and vertices x and y are adjacent in G. ■

Claim 29.12 *Let $S \cap B = \emptyset$, $x \in S \cap A_j$, and y is an arbitrary vertex from B. T' is a minimum cardinality Steiner tree of G for target set $S \cup \{y\} \setminus \{x\}$ if and only if T, obtained from T' by adding vertex x and edge xy, is a minimum cardinality Steiner tree of G for target set S.*

Proof. By Claim 29.9, for G and target set S, there exists a minimum cardinality Steiner tree T such that $d_T(v,s) = d_G(v,s)$ for any vertex v of T. In particular, vertex x is a leaf and its neighbor x^* in T belongs to $L_{i-1}(s)$, that is, to B. Furthermore, any neighbor of x^* in T must belong to $A_j \cup A_{j+1} \cup \cdots \cup A_p$ or to $L_{i-2}(s)$. If $x^* \neq y$, we can get a new minimum cardinality Steiner tree for G and target set S by replacing in T vertex x^* with y and any

edge ux^* of T with edge uy. We can do that since, by Claim 29.10, vertex y is adjacent in G to every vertex u to which vertex x^* was adjacent in T (recall, $u \in A_j \cup A_{j+1} \cup \cdots \cup A_p \cup L_{i-2}(s)$). ■

Thus, we have the following theorem.

Theorem 29.49 [133] *The minimum cardinality Steiner tree problem in distance-hereditary graphs can be solved in linear $O(|V| + |E|)$ time.* ■

29.7.3 Important Subclasses of Distance-Hereditary Graphs

29.7.3.1 Ptolemaic Graphs and Bipartite Distance-Hereditary Graphs

In this subsection, we describe the chordal and distance-hereditary graphs.

The *ptolemaic inequality* $(*)$ in metric spaces is defined as follows.

Definition 29.43 [134] *A connected graph G is* ptolemaic *if, for any four vertices u, v, w, x of G,*

$(*)$ $d(u, v)d(w, x) \leq d(u, w)d(v, x) + d(u, x)d(v, w)$.

Theorem 29.50 [135] *Let G be a graph. The following conditions are equivalent:*

i. *G is ptolemaic.*

ii. *G is distance hereditary and chordal.*

iii. *G is chordal and does not contain an induced gem.*

iv. *For all distinct nondisjoint cliques P and Q of G, $P \cap Q$ separates $P \setminus Q$ and $Q \setminus P$.*

The equivalence of (ii) and (iii) follows from Theorem 29.45: If G is distance-hereditary then obviously G is gem-free. Conversely, if G is gem-free chordal then G is (house, hole, domino, gem)-free and by Theorem 29.45, it is distance-hereditary.

Ptolemaic graphs are characterized in various other ways; see, for example, [136] where the laminar structure of maximal cliques of ptolemaic graphs is described. This is closely related to Bachman Diagrams as described in [6].

Recall that G is chordal if and only if $\mathcal{C}(G)$ is α-acyclic and G is strongly chordal if and only if $\mathcal{C}(G)$ is β-acyclic. A similar fact holds for ptolemaic graphs (see Definition 29.44 for γ-acyclicity).

Theorem 29.51 [80] *Graph G is ptolemaic if and only if the hypergraph $\mathcal{C}(G)$ of its maximal cliques is γ-acyclic.* ■

Theorems 29.45 and 29.44 imply the following corollary.

Corollary 29.19 *A graph is bipartite distance-hereditary if and only if it is bipartite $(6, 2)$-chordal.*

Proof. Obviously, bipartite $(6, 2)$-chordal graphs are (house, hole, domino, gem)-free and thus, by Theorem 29.45, are distance-hereditary. Conversely, let G be a bipartite distance-hereditary graph. Then, by Theorem 29.45, every cycle of length at least 5 has two (crossing) chords which shows the assertion. ■

29.7.3.2 Block Graphs

There is an even more restrictive subclass of chordal distance-hereditary graphs, namely the *block graphs* which can be defined as the connected graphs whose blocks (i.e., 2-connected components) are cliques. Let $K_4 - e$ denote the clique of four vertices minus an edge (also called *diamond*).

Buneman's four-point condition $(**)$ for distances in connected graphs requires that for every four vertices u, v, x and y the following inequality holds:

$$(**) \quad d(u,v) + d(x,y) \leq \max\{d(u,x) + d(v,y),\ d(u,y) + d(v,x)\}.$$

It characterizes the metric properties of trees as Buneman [137] has shown. A connected graph is a tree if and only if it is triangle-free and fulfills Buneman's four-point condition $(**)$.

Theorem 29.52 [138] *Let G be a connected graph. The following conditions are equivalent:*

i. *G is a block graph.*

ii. *G is $(K_4 - e)$-free chordal.*

iii. *G fulfills Buneman's four-point condition $(**)$.* ■

Theorem 29.53 [13] *G is a block graph if and only if $\mathcal{C}(G)$ is Berge-acyclic.* ■

There are various other characterizations of block graphs—see for example [3] for a survey.

29.7.3.3 γ-Acyclic Hypergraphs

The basic subject of this subsection are γ-acyclic hypergraphs. Fagin [6,7] gives various equivalent definitions of γ-acyclicity.

Definition 29.44 [6,7] *Let $H = (V, \mathcal{E})$ be a hypergraph.*

i. *A γ-cycle in a hypergraph $H = (V, \mathcal{E})$ is a sequence $C = (v_1, E_1, v_2, E_2, \ldots, v_k, E_k)$, $k \geq 3$, of distinct vertices $v_1, v_2, \ldots, v_k$ and distinct hyperedges $E_1, E_2, \ldots, E_k$ such that for all i, $1 \leq i \leq k$, $v_i \in E_i \cap E_{i+1}$ holds and for all i, $1 \leq i < k$, $v_i \notin E_j$ for $j \neq i, i+1$ holds (index arithmetic modulo k).*

ii. *A hypergraph is γ-acyclic if it has no γ-cycle.*

Note that the only difference to special cycles is the condition $1 \leq i < k$ instead of $1 \leq i \leq k$. Fagin [6] gives some other variants of γ-acyclicity and shows that all these conditions are equivalent. A crucial property among them is the following separation property:

Theorem 29.54 *A hypergraph $H = (V, \mathcal{E})$ is γ-cyclic if and only if there is a nondisjoint pair E, F of hyperedges such that in the hypergraph that results by removing $E \cap F$ from every edge, what is left of E is connected to what is left of F.* ■

This leads to the following tree structure of separators in γ-acyclic hypergraphs (it has been rediscovered under various names in subsequent papers on ptolemaic graphs, e.g., in [136]).

Definition 29.45 [6,139–141] *For a hypergraph $H = (V, \mathcal{E})$, we define:*

i. *Bachman (H) is the hypergraph obtained by closing $\mathcal{E}$ under intersection, that is, S is in Bachman(H) if it is the intersection of some hyperedges from H (including the hyperedges from $\mathcal{E}$ themselves).*

ii. *The* Bachman diagram *of H is the following undirected graph with Bachman (H) as its node set, and with an edge between two nodes S, T if S is a proper subset of T, that is, $S \subset T$ and there is no other W in Bachman (H) with $S \subset W \subset T$.*

iii. *A Bachman diagram is* loop-free *if it is a tree.*

The tree property of the Bachman diagram is closely related to uniqueness properties in data connections; see [6] for a detailed discussion of various properties which are equivalent to γ-acyclicity and related work on desirable properties of relational database schemes.

The main theorem on γ-acyclicity is the following:

Theorem 29.55 [6] *Let $H = (V, \mathcal{E})$ be a connected hypergraph. The following are equivalent*:

1. *H is γ-acyclic.*
2. *Every connected join expression over H is monotone.*
3. *Every connected, sequential join expression over H is monotone.*
4. *The join dependency $\bowtie H$ implies that every connected subset of H has a lossless join.*
5. *There is a unique relationship among each set of attributes for each consistent database over H.*
6. *The Bachman diagram of H is loop-free.*
7. *H has a unique minimal connection among each set of its nodes.* ■

29.8 TREEWIDTH AND CLIQUE-WIDTH OF GRAPHS

29.8.1 Treewidth of Graphs

Treewidth of a graph measures the tree-likeness of a graph. Treewidth of trees has value one, and if the treewidth of a graph class is bounded by a constant, this has important consequences for the efficient solution of many problems on the class. Treewidth was introduced by Robertson and Seymour in the famous graph minor project by Robertson and Seymour (see, e.g., [142–145] and is one of the most important concepts of algorithmic graph theory. It also came up as partial k-trees which have many applications (see e.g., [9]). A good survey is given by Bodlaender [11] and Kloks [146].

We first define k-trees recursively.

Definition 29.46 *Let $k \geq 1$ be an integer. The following graphs are k-trees:*

i. *Any clique K_k with k vertices is a k-tree.*

ii. *Let $G = (V, E)$ be a k-tree, let $x \notin V$ be a new vertex and let $C \subseteq V$ be a clique with k vertices. Then also $G' = (V \cup \{x\}, E \cup \{ux \mid u \in C\})$ is a k-tree.*

iii. *There are no other k-trees.*

It is easy to see that for $k = 1$, the k-trees are exactly the trees, and for any k, k-trees are chordal with maximum clique size $k + 1$ if the graph is no clique. More exactly, all maximal cliques have size $k + 1$ in this case. See [147] for simple characterizations of k-trees.

Definition 29.47 *Graph $G' = (V, E')$ is a* partial k-tree *if there is a k-tree $G = (V, E)$ with $E' \subseteq E$.*

Obviously, every graph with n vertices is a partial n-tree, and every k-tree is a partial k-tree. The following parameter is of tremendous importance for the efficient solution of algorithmic problems on graphs.

Definition 29.48 *The* treewidth *$tw(G)$ of a given graph G is the minimum value k for which G is a partial k-tree.*

Determining the treewidth of a graph is NP-hard [9].

Treewidth was defined in a different way by Robertson and Seymour (see, e.g., [142–145]) via tree decompositions of graphs:

Definition 29.49 *A* tree decomposition of a graph $G = (V, E)$ *is a pair $D = (S, T)$ with the following properties:*

i. *$S = \{V_i \mid i \in I\}$ is a finite collection of subsets of vertices (sometimes called* bags*).*

ii. *$T = (I, F)$ is a tree with one node for each subset from S.*

iii. *$\bigcup_{i \in I} V_i = V$.*

iv. *For all edges $(v, w) \in E$, there is a subset (i.e., a bag) $V_i \in S$ such that both v and w are contained in V_i.*

v. *For each vertex $x \in V$, the set of tree nodes $\{i \mid x \in V_i\}$ forms a subtree of T.*

Condition (v) corresponds to the join tree condition of α-acyclic hypergraphs and to the clique tree condition of chordal graphs. Thus, a graph is chordal if and only if it has a tree decomposition into cliques.

The *width* of a tree decomposition is the maximum bag size minus one. It is not hard to see that the following holds (see, e.g., [146]):

Lemma 29.12 *The treewidth of a graph equals the minimum width over all of its tree decompositions.* ■

The fundamental importance of treewidth for algorithmic applications is twofold: First of all, many problems can be solved by dynamic programming in a bottom-up way along a tree decomposition (or equivalently, an embedding into a k-tree) of the graph, and the running time is *quite good* for *small* k. The literature [10,148] give many examples for this approach. Second, there is a deep relationship to Monadic Second-Order Logic described in various papers by Courcelle [149] (and in many other papers of this author; see also Bodlaender's tourist guide [11]). Roughly speaking, the following holds.

Whenever a problem Π is expressible in Monadic second-order logic and $\mathcal{C}$ is a graph class of bounded treewidth (with given tree decomposition for each input graph) then problem Π can be efficiently solved on every input graph from $\mathcal{C}$.

As an example, consider 3-colorability of a graph (which is well known to be NP-complete):

$\exists W_1 \subseteq V \exists W_2 \subseteq V \exists W_3 \subseteq V \forall v \in V(v \in W_1 \vee v \in W_2 \vee v \in W_3) \wedge \forall v \in V \forall w \in V(vw \in E \Rightarrow (\neg(v \in W_1 \wedge w \in W_1) \wedge \neg(v \in W_2 \wedge w \in W_2) \wedge \neg(v \in W_3 \wedge w \in W_3)))$.

The detour via logic, however, leads to astronomically large constant factors in the running time of such algorithms. Therefore it is of crucial importance to have a tree decomposition of the input graph with very small width. We know already that the problem of determining treewidth is NP-complete.

Theorem 29.56 [150] *For each integer $k \geq 1$ there is a linear-time algorithm which for given graph G either determines that $tw(G) > k$ holds or otherwise finds a tree decomposition with width k.* ■

Some classes of graphs (cactus graphs, series-parallel graphs, Halin graphs, outerplanar graphs, etc.) have bounded treewidth. See [11] for more information.

Thorup [151] gives important examples of small treewidth in computer science applications.

Another closely related graph parameter called *tree-length* is proposed by Dourisboure and Gavoille [152]. It measures how close a graph is to being chordal. The tree-length of G is defined using tree decompositions of G (see Definition 29.49). Graphs of tree-length k are the graphs that have a tree decomposition where the distance in G between any pair of vertices that appear in the same bag of the tree decomposition is at most k. We discuss this and related parameters in Section 29.10.

29.8.2 Clique-Width of Graphs

The notion of *clique-width* of a graph, defined by Courcelle et al. (in the context of graph grammars) in [153], is another fundamental example of a width parameter on graphs which leads to efficient algorithms for problems expressible in some kind of Monadic second-order logic.

More formally, the clique-width $cw(G)$ of a graph G is defined as the minimum number of different integer labels which allow to generate graph G by using the following four kinds of operations on vertex-labeled graphs:

i. Creation of a new vertex labeled by integer l.

ii. Disjoint union of two (vertex-labeled and vertex-disjoint) graphs (i.e., co-join).

iii. Join between the set of all vertices with label i and the set of all vertices with label j for $i \neq j$ (i.e., all edges between the two sets are added).

iv. Relabeling of all vertices of label i by label j.

A *k-expression* for a graph G of clique-width k describes the recursive generation of G by repeatedly applying these operations using at most k different labels.

Obviously, any graph with n vertices can be generated using n labels (for each vertex a specific one). Thus $cw(G) \leq n$ if G has n vertices.

Clique-width is more powerful than treewidth in the sense that if a class of graphs has bounded treewidth then it also has bounded clique-width but not vice versa [154]—the clique-width of cliques of arbitrary size is two whereas their treewidth is unbounded. In particular, an upper bound for the clique-width of a graph is obtained from its treewidth as follows.

Theorem 29.57 [155] *For any graph G, $cw(G) \leq 3 \cdot 2^{tw(G)-1}$.* ■

Similarly as for treewidth, the concept of clique-width of a graph has attracted much attention due to the fact that there is a similarly close connection to Monadic second-order logic. In [156], Courcelle et al. have shown that every graph problem definable in LinMSOL(τ_1) (a variant of Monadic second-order logic using quantifiers on vertex sets but not on edge sets) is solvable in linear time on graphs with bounded clique-width if a k-expression describing the input graph is given.

The problems maximum weight stable set, maximum weight clique, k-coloring for fixed k, Steiner tree, and domination are examples of LinMSOL(τ_1) definable problems whereas coloring and Hamiltonian circuit are not.

Theorem 29.58 [156] *Let $\mathcal{C}$ be a class of graphs of clique-width at most k such that there is an $\mathcal{O}(f(|E|,|V|))$ algorithm, which for each graph G in $\mathcal{C}$, constructs a k-expression defining it. Then for every LinMSOL(τ_1) problem on $\mathcal{C}$, there is an algorithm solving this problem in time $\mathcal{O}(f(|E|,|V|))$.* ■

Moreover, for some other problems which are not expressible in this way, there are polynomial time algorithms for classes of bounded clique-width [157–159].

It is not hard to see that the class of cographs is exactly the class of graphs having clique-width at most 2, and a 2-expression can be found in linear time along the cotree of a cograph:

Proposition 29.13 *The clique-width of graph G is at most 2 if and only if G is a cograph.*

Clique-width is closely related to modular decomposition as the following proposition shows:

Proposition 29.14 [154,156] *The clique-width of a graph G is the maximum of the clique-width of its prime subgraphs, and the clique-width of the complement graph $\overline{G}$ is at most twice the clique-width of G.*

It is easy to see that the clique-width of thin spiders is at most 4. Thus, a simple consequence of Proposition 29.14 is that the clique-width of P_4-sparse graphs is bounded.

The fact that the clique-width of distance-hereditary graphs is at most three (which, at first glance, does not seem to be surprising but the proof is quite technical) is based on pruning sequences (see Theorem 29.46).

Theorem 29.59 [160] *The clique-width of distance-hereditary graphs is at most 3, and corresponding 3-expressions can be constructed in linear time.*

In the same paper [160] it is shown that unit interval graphs have unbounded clique-width. For very similar reasons, bipartite permutation graphs have unbounded clique-width [161]. Various other classes of bounded and unbounded clique-width are described in [162–167] and many other papers. See [168] for recent results on graph classes of bounded clique-width.

In [169], Fellows et al. show that determining clique-width is NP-complete. The recognition problem for graphs of clique-width at most three is solvable in polynomial time [170]. For any fixed $k \geq 4$, the problem of recognizing all graphs with clique-width at most k in polynomial time is open.

The notion of NLC-width introduced by Wanke [171] is closely related to clique-width. The NLC-width of a graph is not greater than its clique-width, and the clique-width of a graph is twice its NLC-width [172]. Computing the NLC-width of a graph is NP-complete [173]. The graphs of NLC-width 1 are the cographs, and the class of graphs of NLC-width at

most 2 can be recognized in polynomial time [174]. Similarly as for clique-width (with $k \geq 4$), recognition of NLC-width at most k is open for $k \geq 3$.

Oum and Seymour [175,176] investigated the important concept of rank-width and its relationship to clique-width, treewidth and branchwidth. Oum showed that a graph has rank-width 1 if and only if it is distance hereditary.

29.9 COMPLEXITY OF SOME PROBLEMS ON TREE-STRUCTURED GRAPH CLASSES

The most prominent classes with tree structure in this chapter are chordal and dually chordal graphs, strongly chordal graphs and chordal bipartite graphs as well as distance-hereditary graphs. In the following, we describe a variety of complexity results for some problems on these classes. See also [19] for a final chapter on such results.

Recall that the recognition problem for chordal and dually chordal graphs is solvable in linear time, while the recognition of strongly chordal and of chordal bipartite graphs can be done in time $\mathcal{O}(\min(n^2, m \log n))$ (see [19]). Recall also that distance-hereditary graphs can be recognized in linear time [104,129].

The graph isomorphism problem was shown to be isomorphism-complete, that is as hard as in the general case, for strongly chordal graphs and chordal bipartite graphs [177]. The graph isomorphism problem for distance-hereditary graphs is solvable in linear time [136] (a first step for this was done in [178]); see also [179].

The four basic problems independent set [GT20], clique [GT19], chromatic number [GT4], and partition into cliques [GT15] (see [40]), are known to be polynomial-time solvable for perfect graphs [180,181] and thus for chordal graphs as well as strongly chordal graphs and chordal bipartite graphs. In some cases, there are better time bounds using prefect elimination orderings and similar tools. For dually chordal graphs, however, these four problems are NP-complete [63].

Hamiltonian circuit ([GT37] of [40]) is NP-complete for strongly chordal graphs and for chordal bipartite graphs [182] (and thus it is NP-complete for chordal as well as for dually chordal graphs).

Dominating set [GT2] and Steiner tree [ND12] [40] are solvable in linear time for dually chordal graphs [63] and thus for strongly chordal graphs while they are NP-complete for chordal graphs (even for split graphs [84]) and for chordal bipartite graphs [183].

For a given graph $G = (V, E)$, the *maximum induced matching problem* asks for a maximum set of edges having pairwise distance at least 2. While it is well known that the maximum matching problem is solvable in polynomial time, the maximum induced matching problem was shown to be NP-complete even for bipartite graphs [184,185]. For chordal graphs and for chordal bipartite graphs, however, it is solvable in polynomial time [184,186] and for chordal graphs, it is solvable in linear time [187]. It is NP-complete for dually chordal graphs [188]. Maximum induced matching can be generalized to hypergraphs and is solvable in polynomial time for α-acyclic hypergraphs but NP-complete for hypertrees [188].

For a given hypergraph $H = (V, \mathcal{E})$, the *exact cover problem* ([SP2] of [40]) asks for the existence of a subset $\mathcal{E}' \subseteq \mathcal{E}$ such that every vertex of V is in exactly one of the sets in $\mathcal{E}'$. The exact cover problem is NP-complete even for 3-regular hypergraphs [42]. In [188], it is shown that the exact cover problem is NP-complete for α-acyclic hypergraphs but solvable in linear time for hypertrees.

For a given graph $G = (V, E)$, the *efficient domination problem* asks for the existence of a set of closed neighborhoods of G forming an exact cover of V; thus, the efficient domination problem for G corresponds to the Exact Cover problem for the closed neighborhood hypergraph of G. It was introduced by Biggs [189] under the name *perfect code*.

The efficient domination problem is NP-complete for chordal graphs [190] and for chordal bipartite graphs [191]. In [188], it is shown that the efficient domination problem is solvable in linear time for dually chordal graphs.

For a given graph $G = (V, E)$, the *efficient edge domination problem* is the efficient domination problem for the line graph $L(G)$. It appears under the name *dominating-induced matching problem* in various papers; see for example [192]. The efficient edge domination problem is solvable in linear time for chordal graphs [188] and for dually chordal graphs [188] as well as for chordal bipartite graphs (and even solvable in polynomial time for hole-free graphs) [194].

For distance-hereditary graphs, there is a long list of papers showing that certain problems are efficiently solvable on this class. Most of these papers were published before the clique-width aspect was found. Theorem 29.58 covers many of these problems; on the other hand, it might be preferable to have direct dynamic programming algorithms using the tree structure of distance-hereditary graphs since the constant factors in algorithms using Theorem 29.58 are astronomically large (and similarly for graphs of bounded treewidth). However, various problems such as Hamilton cycle (HC) and variants cannot be expressed in MSOL; see also the algorithm for Steiner tree on distance-hereditary graphs.

The four basic problems can be solved in time $\mathcal{O}(n)$ if a pruning sequence of the input graph is given [129].

HC was shown to be solvable in time $\mathcal{O}(n^3)$ [195,196], in time $\mathcal{O}(n^2)$ [197] and finally in time $\mathcal{O}(n+m)$ for the HC problem [198,199] for HC and variants giving a unified approach. In [199], a detailed history of the complexity results for HC on distance-hereditary graphs is given. For the subclass of bipartite distance-hereditary graphs, a linear-time algorithm for HC was given already in [200].

The dominating set problem was solved in linear time in [201,202] for distance-hereditary graphs. The efficient domination and efficient edge domination problems are expressible in MSOL and thus efficiently solvable for distance-hereditary graphs.

29.10 METRIC TREE-LIKE STRUCTURES IN GRAPHS

There are few other graph parameters measuring tree likeness of a (unweighted) graph from a metric point of view. Two of them are also based on the notion of tree-decomposition of Robertson and Seymour [145] (see Definition 29.49).

29.10.1 Tree-Breadth, Tree-Length, and Tree-Stretch of Graphs

The *length* of a tree-decomposition T of a graph G is $\lambda := \max_{i \in I} \max_{u,v \in V_i} d_G(u, v)$ (i.e., each bag V_i has diameter at most λ in G). The *tree-length* of G, denoted by $tl(G)$, is the minimum of the length over all tree-decompositions of G [152]. As chordal graphs are exactly those graphs that have a tree decomposition where every bag is a clique [16–18], we can see that tree-length generalizes this characterization and thus the chordal graphs are exactly the graphs with tree-length 1. Note that tree-length and treewidth are not related to each other graph parameters. For instance, a clique on n vertices has tree-length 1 and treewidth $n-1$, whereas a cycle on $3n$ vertices has treewidth 2 and tree-length n. One should also note that many graph classes with unbounded treewidth have bounded tree-length, such as chordal, interval, split, AT-free, and permutation graphs [152]. Analysis of a number of real-life networks, taken from different domains like Internet measurements, biological datasets, web graphs, social and collaboration networks, performed in [203,204] shows that those networks have sufficiently large treewidth but their tree-length is relatively small.

The *breadth* of a tree-decomposition T of a graph G is the minimum integer r such that for every $i \in I$ there is a vertex $v_i \in V$ with $V_i \subseteq N^r[v_i]$ (i.e., each bag V_i can be covered by a disk $N^r[v_i] := \{u \in V(G) : d_G(u, v_i) \leq r\}$ of radius at most r in G). Note that vertex v_i does not need to belong to V_i. The *tree-breadth* of G, denoted by $tb(G)$, is the minimum of the breadth over all tree-decompositions of G [205]. Evidently, for any graph G, $1 \leq tb(G) \leq tl(G) \leq 2tb(G)$ holds. Hence, if one parameter is bounded by a constant for a graph G then the other parameter is bounded for G as well.

Note that the notion of *acyclic* (R, D)*-clustering of a graph* introduced in [206] combines tree-breadth and tree-length into one notion. Graphs admitting acyclic (D, D)-clustering are exactly graphs with tree-length at most D, and graphs admitting acyclic $(R, 2R)$-clustering are exactly graphs with tree-breadth at most R. Hence, all chordal, chordal bipartite, and dually chordal graphs have tree-breadth 1 [206].

In view of tree-decomposition T of G, the smaller parameters $tl(G)$ and $tb(G)$ of G are, the closer graph G is to a tree metrically. Unfortunately, while graphs with tree-length 1 (as they are exactly the chordal graphs) can be recognized in linear time, the problem of determining whether a given graph has tree-length at most λ is NP-complete for every fixed $\lambda > 1$ (see [207]). Judging from this result, it is conceivable that the problem of determining whether a given graph has tree-breadth at most ρ is NP-complete, too. 3-Approximation algorithms for computing the tree-length and the tree-breadth of a graph are proposed in [152,204,205].

Proposition 29.15 [152] *There is a linear-time algorithm that produces for any graph G a tree-decomposition of length at most $3tl(G) + 1$.*

Proposition 29.16 [204,205] *There is a linear-time algorithm that produces for any graph G a tree-decomposition of breadth at most $3tb(G)$.*

It follows from results of [208] and [152] also that any graph G with small tree-length or small tree-breadth can be embedded to a tree with a small additive distortion.

Proposition 29.17 *For any (unweighted) connected graph $G = (V, E)$ there is an unweighted tree $H = (V, F)$ (on the same vertex set but not necessarily a spanning tree of G) for which the following is true:*

$$\forall u, v \in V, \quad d_H(u, v) - 2 \leq d_G(u, v) \leq d_H(u, v) + 3\ tl(G) \leq d_H(u, v) + 6\ tb(G).$$

Such a tree H can be constructed in $O(|E|)$ time.

Previously, these type of results were known for chordal graphs and dually chordal graphs [209], k-chordal graphs [210], and δ-hyperbolic graphs [211].

Graphs with small tree-length or small tree-breadth have many other nice properties. Every n-vertex graph with tree-length $tl(G) = \lambda$ has an additive 2λ-spanner with $O(\lambda n + n \log n)$ edges and an additive 4λ-spanner with $O(\lambda n)$ edges, both constructible in polynomial time [212]. Every n-vertex graph G with $tb(G) = \rho$ has a system of at most $\log_2 n$ collective additive tree $(2\rho \log_2 n)$-spanners constructible in polynomial time [213]. Those graphs also enjoy a 6λ-additive routing labeling scheme with $O(\lambda \log^2 n)$ bit labels and $O(\log \lambda)$ time routing protocol [214], and a $(2\rho \log_2 n)$-additive routing labeling scheme with $O(\log^3 n)$ bit labels and $O(1)$ time routing protocol with $O(\log n)$ message initiation time (by combining results of [213] and [215]). See appropriate papers for more details.

Here we elaborate a little bit more on a connection established in [205] between the tree-breadth and the tree-stretch of a graph (and the corresponding tree t-spanner problem).

The *tree-stretch* $ts(G)$ of a graph $G = (V, E)$ is the smallest number t such that G admits a *spanning* tree $T = (V, E')$ with $d_T(u, v) \leq td_G(u, v)$ for every $u, v \in V$. T is called a *tree t-spanner* of G and the problem of finding such tree T for G is known as the *tree t-spanner problem*. Note that as T is a spanning tree of G, necessarily $d_G(u, v) \leq d_T(u, v)$ and $E' \subseteq E$. It is known that the tree t-spanner problem is NP-hard [216]. The best known approximation algorithms have approximation ratio of $O(\log n)$ [205,217].

The following two results were obtained in [205].

Proposition 29.18 [205] *For every graph G, $tb(G) \leq \lceil ts(G)/2 \rceil$ and $tl(G) \leq ts(G)$.*

Proposition 29.19 [205] *For every n-vertex graph G, $ts(G) \leq 2tb(G) \log_2 n$. Furthermore, a spanning tree T of G with $d_T(u, v) \leq (2tb(G) \log_2 n)\ d_G(u, v)$, for every $u, v \in V$, can be constructed in polynomial time.*

Proposition 29.19 is obtained by showing that every n-vertex graph G with $tb(G) = \rho$ admits a tree $(2\rho \log_2 n)$-spanner constructible in polynomial time. Together with Proposition 29.18, this provides a $\log_2 n$-approximate solution for the tree t-spanner problem in general unweighted graphs.

29.10.2 Hyperbolicity of Graphs and Embedding Into Trees

δ-Hyperbolic metric spaces have been defined by Gromov [218] in 1987 via a simple 4-point condition: for any four points u, v, w, x, the two larger of the distance sums $d(u, v) + d(w, x), d(u, w) + d(v, x), d(u, x) + d(v, w)$ differ by at most 2δ. They play an important role in geometric group theory, geometry of negatively curved spaces, and have recently become of interest in several domains of computer science, including algorithms and networking. For example, (a) it has been shown empirically in [219] (see also [220]) that the Internet topology embeds with better accuracy into a hyperbolic space than into an Euclidean space of comparable dimension, (b) every connected finite graph has an embedding in the hyperbolic plane so that the greedy routing based on the virtual coordinates obtained from this embedding is guaranteed to work (see [221]).

A connected graph $G = (V, E)$ equipped with standard graph metric d_G is δ-*hyperbolic* if the metric space (V, d_G) is δ-hyperbolic. More formally, let G be a graph and u, v, w and x be its four vertices. Denote by S_1, S_2, S_3 the three distance sums, $d_G(u, v) + d_G(w, x)$, $d_G(u, w) + d_G(v, x)$ and $d_G(u, x) + d_G(v, w)$ sorted in nondecreasing order $S_1 \leq S_2 \leq S_3$. Define the *hyperbolicity of a quadruplet* u, v, w, x as $\delta(u, v, w, x) = \frac{S_3 - S_2}{2}$. Then the *hyperbolicity* $\delta(G)$ *of a graph* G is the maximum hyperbolicity over all possible quadruplets of G, that is,

$$\delta(G) = \max_{u,v,w,x \in V} \delta(u, v, w, x).$$

δ-Hyperbolicity measures the local deviation of a metric from a tree metric; a metric is a tree metric if and only if it has hyperbolicity 0. Note that chordal graphs have hyperbolicity at most 1 [222], while k-chordal graphs have hyperbolicity at most $k/4$ [223].

The best known algorithm to calculate hyperbolicity has time complexity of $O(n^{3.69})$, where n is the number of vertices in the graph; it was proposed in [224] and involves matrix multiplications. Authors of [224] also propose a 2-approximation algorithm for calculating hyperbolicity that runs in $O(n^{2.69})$ time and a $2 \log_2 n$-approximation algorithm that runs in $O(n^2)$ time.

According to [211], if a graph G has small hyperbolicity then it can be embedded to a tree with a small additive distortion.

Proposition 29.20 [211] *For any (unweighted) connected graph $G = (V, E)$ with n vertices there is an unweighted tree $H = (V, F)$ (on the same vertex set but not necessarily a spanning tree of G) for which the following is true:*

$$\forall u, v \in V, \quad d_H(u, v) - 2 \leq d_G(u, v) \leq d_H(u, v) + O(\delta(G) \log n).$$

Such a tree H can be constructed in $O(|E|)$ time.

Thus, the distances in n-vertex δ-hyperbolic graphs can efficiently be approximated within an additive error of $O(\delta \log n)$ by a tree metric and this approximation is sharp (see [211,218,225]). An earlier result of Gromov [218] established similar distance approximations, however Gromov's tree is weighted, may have Steiner points and needs $O(n^2)$ time for construction.

It is easy to show that every graph G admitting a tree T with $d_G(x, y) \leq d_T(x, y) \leq d_G(x, y) + r$ for any $x, y \in V$ is r-hyperbolic. So, the hyperbolicity of a graph G is an indicator of an embedability of G in a tree with an additive distortion.

Graphs and general geodesic spaces with small hyperbolicities have many other algorithmic advantages. They allow efficient approximate solutions for a number of optimization problems. For example, Krauthgamer and Lee [226] presented a PTAS for the traveling salesman problem when the set of cities lie in a hyperbolic metric space. Chepoi and Estellon [227] established a relationship between the minimum number of balls of radius $r + 2\delta$ covering a finite subset S of a δ-hyperbolic geodesic space and the size of the maximum r-packing of S and showed how to compute such coverings and packings in polynomial time. Chepoi et al. gave in [211] efficient algorithms for fast and accurate estimations of diameters and radii of δ-hyperbolic geodesic spaces and graphs. Additionally, Chepoi et al. showed in [228] that every n-vertex δ-hyperbolic graph has an additive $O(\delta \log n)$-spanner with at most $O(\delta n)$ edges and enjoys an $O(\delta \log n)$-additive routing labeling scheme with $O(\delta \log^2 n)$ bit labels and $O(\log \delta)$ time routing protocol.

The following relations between the tree-length and the hyperbolicity of a graph were established in [211].

Proposition 29.21 [211] *For every n-vertex graph G, $\delta(G) \leq tl(G) \leq O(\delta(G) \log n)$.*

Combining this with results from [205] (see Propositions 29.18 and 29.19), one gets the following inequalities.

Proposition 29.22 [229] *For any n-vertex graph G, $\delta(G) \leq ts(G) \leq O(\delta(G) \log^2 n)$.*

This proposition says, in particular, that every δ-hyperbolic graph G admits a tree $O(\delta \log^2 n)$-spanner. Furthermore, such a spanning tree for a δ-hyperbolic graph can be constructed in polynomial time (see [205]).

The problem of approximating a given graph metric by a *simpler* metric is well motivated from several different perspectives. A particularly simple metric of choice, also favored from the algorithmic point of view, is a tree metric, that is, a metric arising from shortest path distance on a tree containing the given points. In recent years, a number of authors considered problems of minimum distortion embeddings of graphs into trees (see [208,230–232]), most popular among them being a noncontractive embedding with minimum multiplicative distortion.

Let $G = (V, E)$ be a graph. The (multiplicative) *tree-distortion* $td(G)$ of G is the smallest number α such that G admits a tree (not necessarily a spanning tree, possibly weighted and with Steiner points) with

$$\forall u, v \in V, \quad d_G(u, v) \leq d_T(u, v) \leq \alpha \, d_G(u, v).$$

The problem of finding, for a given graph G, a tree $T = (V \cup S, F)$ satisfying $d_G(u,v) \leq d_T(u,v) \leq td(G)d_G(u,v)$, for all $u, v \in V$, is known as the *problem of minimum distortion noncontractive embedding of graphs into trees*. In a noncontractive embedding, the distance in the tree must always be larger that or equal to the distance in the graph, that is, the tree distances *dominate* the graph distances.

It is known that this problem is NP-hard, and even more, the hardness result of [230] implies that it is NP-hard to approximate $td(G)$ better than γ, for some small constant γ. The best known 6-approximation algorithm using layering partition technique was recently given in [208]. It improves the previously known 100-approximation algorithm from [232] and 27-approximation algorithm from [231].

The following interesting result was presented in [208].

Proposition 29.23 [208] *For any (unweighted) connected graph $G = (V, E)$ with n vertices there is an unweighted tree $H = (V, F)$ (on the same vertex set but not necessarily a spanning tree of G) for which the following is true:*

$$\forall u, v \in V, \quad d_H(u,v) - 2 \leq d_G(u,v) \leq d_H(u,v) + 3\ td(G).$$

Such a tree H can be constructed in $O(|E|)$ time.

Surprisingly, a multiplicative distortion is turned into an additive one. Moreover, while a tree $T = (V \cup S, F)$ satisfying $d_G(u,v) \leq d_T(u,v) \leq td(G)d_G(u,v)$, for all $u, v \in V$, is NP-hard to find, tree H of Proposition 29.23 is constructible in $O(|E|)$ time. Furthermore, H is unweighted and has no Steiner points.

By adding at most $n = |V|$ new Steiner points to tree H and assigning proper weights to edges of H, the authors of [208] achieve a good noncontractive embedding of a graph G into a tree.

Proposition 29.24 [208] *For any (unweighted) connected graph $G = (V, E)$ there is a weighted tree $H'_\ell = (V \cup S, F)$ for which the following is true:*

$$\forall u, v \in V, \quad d_G(x,y) \leq d_{H'_\ell}(x,y) \leq 3td(G)(d_G(x,y) + 1).$$

Such a tree H'_ℓ can be constructed in $O(|V||E|)$ time.

As pointed out in [208], tree H'_ℓ provides a 6-approximate solution to the problem of minimum distortion noncontractive embedding of an unweighted graph into a tree.

We conclude this section with one more chain of inequalities establishing relations between the tree-stretch, the tree-length, and the tree-distortion of a graph.

Proposition 29.25 [229] *For every n-vertex graph G, $tl(G) \leq td(G) \leq ts(G) \leq 2td(G)\log_2 n$.*

Proposition 29.25 says that if a graph G is noncontractively embeddable into a tree with distortion $td(G)$ then it is embeddable into a spanning tree with stretch at most $2td(G)\log_2 n$. Furthermore, a spanning tree with stretch at most $2td(G)\log_2 n$ can be constructed for G in polynomial time.

References

[1] A. Hajnal and J. Surányi, Über die Auflösung von Graphen in vollständige Teilgraphen, *Ann. Univ. Sci. Budapest, Eötvös Sect. Math.* **1** (1958), 113–121.

[2] J.R.S. Blair and B. Peyton, An introduction to chordal graphs and clique trees, In *Graph Theory and Sparse Matrix Computation*, A. George, J.R. Gilbert, and J.W.H. Liu (Eds.), Springer, New York, 1993, 1–29.

[3] A. Brandstädt, V.B. Le, and J.P. Spinrad, Graph classes: A survey, *SIAM Monographs on Discrete Math. Appl.,* Vol. 3, SIAM, Philadelphia, PA, 1999.

[4] M.C. Golumbic, *Algorithmic Graph Theory and Perfect Graphs*, Academic Press, New York 1980; 2nd edition: *Ann. Discrete Math.* 57, Elsevier Science B.V., Amsterdam, the Netherlands, 2004.

[5] T.A. McKee and F.R. McMorris, Topics in intersection graph theory, *SIAM Monographs on Discrete Math. and Appl.* Vol. 2, Society for Industrial and Applied Mathematics, Philadelphia, PA, 1999.

[6] R. Fagin, Degrees of Acyclicity for hypergraphs and relational database schemes, *Journal ACM* **30** (1983), 514–550.

[7] R. Fagin, Acyclic database schemes (of various degrees): A painless introduction, *Proc. CAAP83 8th Colloquium on Trees in Algebra and Programming*, G. Ausiello and M. Protasi (Eds.), Springer LNCS 159 (1983), pp. 65–89.

[8] N. Robertson and P.D. Seymour, Graph minors. I. Excluding a forest, *J. Comb. Theory (B)* **35** (1983), 39–61.

[9] S. Arnborg, D.G. Corneil, and A. Proskurowski, Complexity of finding embeddings in a k-tree, *SIAM J. Alg. Discr. Meth.* **8** (1987), 277–284.

[10] S. Arnborg and A. Proskurowski, Linear time algorithms for NP-hard problems restricted to partial k-trees, *Discrete Applied Math.* **23** (1989), 11–24.

[11] H.L. Bodlaender, A tourist guide through treewidth, *Acta Cybernetica* **11** (1993), 1–23.

[12] C. Berge, *Graphs and Hypergraphs*, American Elsevier Publishing Co., North-Holland, 1973.

[13] C. Berge, *Hypergraphs*, Elsevier Publishing Co., North-Holland, 1989.

[14] G. Dirac, On rigid circuit graphs, *Abhandl. Math. Seminar Univ. Hamburg* **25** (1961), 71–76.

[15] D.R. Fulkerson and O.A. Gross, Incidence matrices and interval graphs, *Pacific J. Math.* **15** (1965), 835–855.

[16] A. Buneman, A characterization of rigid circuit graphs, *Discrete Math.* **9** (1974), 205–212.

[17] F. Gavril, The intersection graphs of subtrees in trees are exactly the chordal graphs, *J. Comb. Theory (B)* **16** (1974), 47–56.

[18] J.R. Walter, *Representations of Rigid Cycle Graphs*, PhD dissertation, Wayne State University, Detroit, MI, 1972.

[19] J.P. Spinrad, *Efficient Graph Representations, Fields Institute Monographs*, American Mathematical Society, Providence, RI, 2003.

[20] R.E. Tarjan and M. Yannakakis, Simple linear-time algorithms to test chordality of graphs, test acyclicity of hypergraphs, and selectively reduce acyclic hypergraphs, *SIAM J. Computing* **13** (1984), 566–579; Addendum *SIAM J. Computing* **14** (1985), 254–255.

[21] W.W. Barrett, C.R. Johnson, and M. Lundquist, Determinantal formulae for matrix completions associated with chordal graphs, *Linear Algebra Appl.* **121** (1989), 265–289.

[22] C.-W. Ho and R.C.T. Lee, Counting clique trees and computing perfect elimination schemes in parallel, *Inf. Proc. Letters* **31** (1989), 61–68.

[23] A. Frank, Some polynomial algorithms for certain graphs and hypergraphs, In *Proceedings of the 5th British Combinatorial Conference* (1975), *Congressus Numerantium* **XV** (1976), 211–226.

[24] S. Főldes and P.L. Hammer, Split graphs, In *8th South–Eastern Conf. on Combinatorics, Graph Theory and Computing*, F. Hoffman, L. Lesniak-Foster, D. McCarthy, R.C. Mullin, K.B. Reid, and R.G. Stanton (Eds.), Louisiana State University, Baton Rouge, LA (1977), *Congressus Numerantium* **19** (1977), 311–315.

[25] P.L. Hammer and B. Simeone, The splittance of a graph, *Combinatorica* 1 (1981), 275–284.

[26] R.I. Tyshkevich, O.I. Melnikow, and V.M. Kotov, On graphs and degree sequences: The canonical decomposition (in Russian), *Kibernetika* **6** (1981), 5–8.

[27] M. Farber, Characterizations of strongly chordal graphs, *Discrete Math.* **43** (1983), 173–189.

[28] G. Ausiello, A. D'Atri, and M. Moscarini, Chordality properties on graphs and minimal conceptual connections in semantic data models, *J. Comput. Syst. Sci.* **33** (1986), 179–202.

[29] C. Beeri, R. Fagin, D. Maier, and M. Yannakakis, On the desirability of acyclic database schemes, *J. ACM* **30** (1983), 479–513.

[30] E.F. Codd, A relational model of data for large shared data banks, *Communications of the ACM* **13** (1970), 377–387.

[31] P. Honeyman, R. E. Ladner, and M. Yannakakis, Testing the universal instance assumption, *Inf. Proc. Letters* **10** (1980), 14–19.

[32] N. Goodman and O. Shmueli, Syntactic characterization of tree database schemas, *J. ACM* **30** (1983), 767–786.

[33] G. Gottlob, N. Leone, and F. Scarcello, Hypertree decompositions: A survey, In *Proc. MFCS 2001*, J. Sgall, A. Pultr, and P. Kolman, (Eds.), LNCS 2136, Springer, Mariánské Lázně, Czech Republic, 2001, 37–57.

[34] H. Gaifman, On local and nonlocal properties, In *Logic Colloquium'81* (J. Stern ed.,) Elsevier, North-Holland, Amsterdam, the Netherlands, 1982, 105–135.

[35] D. Maier, *The Theory of Relational Databases*, Computer Science Press, Rockville, MD, 1983.

[36] H.J. Ryser, Combinatorial configurations, *SIAM J. Appl. Math.* **17** (1969), 593–602.

[37] F.S. Roberts and J. H. Spencer, A characterization of clique graphs, *J. Comb. Theory (B)* **10** (1971), 102–108.

[38] J. L. Szwarcfiter, A survey on clique graphs, In *Recent Advances in Algorithmic Combinatorics*, C. Linhares-Sales and B. Reed (Eds.), CMS Books in Mathematics, Springer, 2003, 109–136.

[39] L. Alcón, L. Faria, C.M.H. de Figueiredo, and M. Gutierrez, Clique graph recognition is NP-complete, F. Fomin (ed.), WG 2006, *Lecture Notes in Comp. Sci.* **4271** (2006), 269–277; full version in: The complexity of clique graph recognition. *Theor. Comp. Sci.* **410**(21–23) (2009), 2072–2083.

[40] M.R. Garey and D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, Freeman & Co., San Francisco, CA, 1979.

[41] L. Lovász, Coverings and colorings of hypergraphs, In *Proc. 4th Southeastern Conf. on Combinatorics, Graph Theor. Comput., Util. Math. Publ.*, Congr. Numerantium **VIII** (1973), 3–12.

[42] R.M. Karp, Reducibility among combinatorial problems, In *Complexity of Computer Computations*, R.E. Miller and J.W. Thatcher (Eds.), Plenum Press, New York, 1972, 85–103.

[43] N. Linial and M. Tarsi, Deciding hypergraph 2-colorability by H-resolution, *Theor. Comp. Sci.* **38** (1985), 343–347.

[44] M.R. Garey, D.S. Johnson, and L. Stockmeyer, Some simplified NP-complete graph problems, *Theor. Comp. Sci.* **1** (1976), 237–267.

[45] R. Fagin and M.Y. Vardi, The theory of data dependencies—A survey, *Mathematics of Information Processing*, In *Proc. Symp. Appl. Math.*, M. Anshel, W. Gewirtz, (Eds.) Vol. **34** (1986), 19–71, American Mathemaical Society, Providence, RI.

[46] J. Lehel, A characterization of totally balanced hypergraphs, *Discrete Math.* **57** (1985), 59–65.

[47] M.C. Golumbic, Algorithmic aspects of intersection graphs and representation hypergraphs, *Graphs and Combinatorics* **4** (1988), 307–321.

[48] P. Duchet, Propriété de Helly et problèmes de représentation, *Colloqu. Internat. CNRS 260, Problemes Combinatoires et Theorie du Graphs, Orsay, France* (1976), 117–118.

[49] C. Flament, Hypergraphes arborés, *Discrete Math.* **21** (1978), 223–226.

[50] P.J. Slater, A characterization of SOFT hypergraphs, *Canad. Math. Bull.* **21** (1978), 335–337.

[51] P.A. Bernstein and N. Goodman, Power of natural semijoins, *SIAM J. Comput.* **10** (1981), 751–771.

[52] T.A. McKee, How chordal graphs work, *Bull. ICA* **9** (1993), 27–39.

[53] B.D. Acharya and M. las Vergnas, Hypergraphs with cyclomatic number zero, triangulated graphs, and an inequality, *J. Comb. Theor. (B)* **33** (1982), 52–56.

[54] P. Hansen and M. Las Vergnas, On a property of hypergraphs with no cycles of length greater than two, In *Hypergraph Seminar, Lecture Notes in Math.* **411** (1974), 99–101.

[55] M. Lewin, On hypergraphs without significant cycles, *J. Comb. Theor. (B)* **20** (1976), 80–83.

[56] M.H. Graham, On the universal relation, *Tech. Report*, University of Toronto, Ontario, Canada, 1979.

[57] C.T. Yu and M.Z. Ozsoyoglu, An algorithm for tree-query membership of a distributed query, *Proc. 1979 IEEE COMPSAC*, IEEE, New York, 1979, 306–312.

[58] R. Fagin, A.O. Mendelzon, and J.D. Ullman, A simplified universal relation assumption and its properties, *ACM Trans. Database Syst.* **7** (1982), 343–360.

[59] A. Brandstädt, F.F. Dragan, V.D. Chepoi, and V.I. Voloshin, Dually chordal graphs, Technical report SM-DU-225, University of Duisburg 1993; extended abstract in: Proceedings of WG 1993, LNCS 790, 237–251, 1993; full version in *SIAM J. Discr. Math.* **11** (1998), 437–455.

[60] F.F. Dragan, HT-graphs: Centers, connected r-domination and steiner trees, *Comp. Sci. J. Moldova* **1** (1993), 64–83.

[61] F.F. Dragan, C.F. Prisacaru, and V.D. Chepoi, Location problems in graphs and the Helly property (in Russian) (1987) (appeared partially in *Diskretnaja Matematika* **4** (1992), 67–73).

[62] H. Behrendt and A. Brandstädt, Domination and the use of maximum neighborhoods, Technical report SM-DU-204, University of Duisburg, Germany, 2002.

[63] A. Brandstädt, V.D. Chepoi, and F.F. Dragan, The algorithmic use of hypertree structure and maximum neighbourhood orderings, Technical report SM-DU-244, University of Duisburg 1994; extended abstract in: Proceedings of WG 1994, LNCS 903, 65–80, 1994; full version in *Discrete Applied Math.* **82** (1998), 43–77.

[64] J.L. Szwarcfiter and C.F. Bornstein, Clique graphs of chordal and path graphs, *SIAM J. Discrete Math.* **7** (1994) 331–336.

[65] M. Gutierrez and L. Oubiña, Metric characterizations of proper interval graphs and tree-clique graphs, *J. Graph Theor.* **21** (1996), 199–205.

[66] A. Brandstädt, V.D. Chepoi, and F.F. Dragan, Clique r-domination and clique r-packing problems on dually chordal graphs, *SIAM J. Discrete Math.* **10** (1997), 109–127.

[67] P. De Caria, *A Joint Study of Chordal and Dually Chordal Graphs*, PhD thesis, Universidad Nacional de la Plata, Argentina, 2012.

[68] P. De Caria and M. Gutierrez, On minimal vertex separators of dually chordal graphs: Properties and characterizations, *Discrete Appl. Math.* **160** (2012), 2627–2635.

[69] P. De Caria and M. Gutierrez, Comparing trees characteristic to chordal and dually chordal graphs, *Electronic Notes in Discrete Math.* **37** (2011), 33–38.

[70] P. De Caria and M. Gutierrez, On the correspondence between tree representations of chordal and dually chordal graphs, *Discrete Appl. Math.* **164** (2014), 500–511.

[71] A. Leitert, Das Dominating Induced Matching Problem für azyklische Hypergraphen, Diploma thesis, University of Rostock, Germany, 2012.

[72] M. Moscarini, Doubly chordal graphs: Steiner trees and connected domination, *Netw.* **23** (1993), 59–69.

[73] A. Brandstädt, F.F. Dragan, and F. Nicolai, Homogeneously orderable graphs, *Theor. Comput. Sci.* **172** (1997), 209–232.

[74] A. Lubiw, Doubly lexical orderings of matrices, *SIAM J. Comput.* **16** (1987), 854–879.

[75] R. Paige and R.E. Tarjan, Three partition refinement algorithms, *SIAM J. Comput.* **16** (1987), 973–989.

[76] L. Lovász, *Combinatorial Problems and Exercises*, North-Holland, Amsterdam, the Netherlands, 1979.

[77] L. Lovász and M.D. Plummer, *Matching Theory*, North-Holland, Amsterdam, the Netherlands, Math. Studies Vol. 29, 1986.

[78] C. Berge and M. Las Vergnas, Sur un théorème du type Kőnig pour hypergraphes, *Annals NY Acad. Sci.* **175** (1970), 32–40.

[79] A.E. Brouwer and A. Kolen, A super-balanced hypergraph has a nest point, Report ZW 146/80, Mathematisch Centrum, Amsterdam, the Netherlands, 1980.

[80] A. D'Atri and M. Moscarini, On hypergraph acyclicity and graph chordality, *Inf. Proc. Letters* **29** (1988), 271–274.

[81] R.P. Anstee and M. Farber, Characterizations of totally balanced matrices, *J. Algorithms* **5** (1984), 215–230.

[82] A.J. Hoffman, A.W.J. Kolen, and M. Sakarovitch, Totally balanced and greedy matrices, *SIAM J. Alg. Discrete Meth.* **6** (1985), 721–730.

[83] A. Lubiw, Γ-*free matrices*, Master's thesis, Department of Combinatorics and Optimization, University of Waterloo, Canada, 1982.

[84] A.A. Bertossi, Dominating sets for split graphs and bipartite graphs, *Inf. Proc. Letters* **19** (1984), 37–40.

[85] G.J. Chang, Labeling algorithms for domination problems in sun-free chordal graphs, *Discrete Appl. Math.* **22** (1988), 21–34.

[86] G.J. Chang and G.L. Nemhauser, The k-domination and k-stability problem on sun-free chordal graphs, *SIAM J. Alg. Discrete Meth.* **5** (1984), 332–345.

[87] G.J. Chang, M. Farber, and Z. Tuza, Algorithmic aspects of neighbourhood numbers, *SIAM J. Discrete Math.* **6** (1993), 24–29.

[88] K. Iijima and Y. Shibata, A bipartite representation of a triangulated graph and its chordality, Deptartment of Computer Science, Gunma University, Maebashi, Japan, CS 79–1, 1979.

[89] J.P. Spinrad, Doubly lexical ordering of dense 0–1 matrices, *Inf. Proc. Letters* **45** (1993), 229–235.

[90] A.E. Brouwer, P. Duchet, and A. Schrijver, Graphs whose neighborhoods have no special cycles, *Discrete Math.* **47** (1983), 177–182.

[91] T.A. McKee, Strong clique trees, neighborhood trees, and strongly chordal graphs, *J. Graph Theor* **33** (2000), 125–183.

[92] S. Ma and J. Wu, Characterizing strongly chordal graphs by using minimal relative separators, *Combinatorial Designs and Applications*, W.D. Wallis, H. Shen, W. Wei, and L. Shu (Eds.): Lecture Notes in Pure and Applied Mathematics 126, Marcel Dekker, New York, 1990, 87–95.

[93] M.C. Golumbic and C.F. Goss, Perfect elimination and chordal bipartite graphs, *J. Graph Theor.* **2** (1978), 155–163.

[94] R.B. Hayward, Weakly triangulated graphs, *J. Comb. Theory (B)* **39** (1985), 200–208.

[95] A. Brandstädt, Classes of bipartite graphs related to chordal graphs, *Discrete Appl. Math.* **32** (1991), 51–60.

[96] E. Dahlhaus, Chordale Graphen im besonderen Hinblick auf parallele Algorithmen, Habilitation Thesis, Universität Bonn, Germany, 1991.

[97] R. Uehara, Recognition of chordal bipartite graphs, Proceedings of ICALP, *Lecture Notes in Comp. Sci.* **2380** (2002), 993–1004.

[98] J. Huang, Representation characterizations of chordal bipartite graphs, *J. Combinatorial Theor. B* **96** (2006) 673–683.

[99] A. Berry and A. Sigayret, Dismantlable lattices in the mirror, *Proc. ICFCA* 2013, 44–59.

[100] D.G. Corneil, H. Lerchs, and L. Stewart-Burlingham, Complement reducible graphs, *Discrete Appl. Math.* **3** (1981), 163–174.

[101] D.G. Corneil, Y. Perl, and L.K. Stewart, Cographs: Recognition, applications, and algorithms, *Congressus Numer.* **43** (1984), 249–258.

[102] D.G. Corneil, Y. Perl, and L.K. Stewart, A linear recognition algorithm for cographs, *SIAM J. Comput.* **14** (1985), 926–934.

[103] D. Kratsch, R.M. McConnell, K. Mehlhorn, and J. Spinrad, Certifying algorithms for recognizing interval graphs and permutation graphs, *SIAM J. Comput.* **36**(2) (2006), 326–353.

[104] G. Damiand, M. Habib, and Ch. Paul, A simple paradigm for graph recognition: Application to cographs and distance-hereditary graphs, *TCS* **263** (2001), 99–111.

[105] A. Bretscher, D.G. Corneil, M. Habib, and C. Paul, A simple linear time LexBFS cograph recognition algorithm, *Conference Proceedings of International Workshop on Graph-Theoretic Concepts in Computer Science*, In *Lecture Notes in Comp. Sci.* 2880, Hans L. Bodlaender (Ed.), Elspeet, the Netherlands, 2003, 119–130.

[106] R.M. McConnell and J.P. Spinrad, Modular decomposition and transitive orientation, *Discrete Math.* **201** (1999), 189–241.

[107] T. Gallai, Transitiv orientierbare Graphen, *Acta Math. Acad. Sci. Hung.* **18** (1967), 25–66.

[108] A. Cournier and M. Habib, A new linear algorithm for modular decomposition, *LIRMM, University Montpellier* (1995), Preliminary version in: *Trees in Algebra and Programming—CAAP*, LNCS **787** (1994), 68–84.

[109] E. Dahlhaus, J. Gustedt, and R.M. McConnell, Efficient and practical modular decomposition, *J. Algorithms* **41**(2) (2001), 360–387.

[110] M. Habib, F. de Montgolfier, and C. Paul, A simple linear time modular decomposition algorithm for graphs, using order extension, *Proc. 9th Scandinav. Workshop on Algorithm Theory*, *Lecture Notes in Comp. Sci.* **3111** (2004), 187–198.

[111] M. Tedder, D.G. Corneil, M. Habib, and C. Paul, Simpler linear-time modular decomposition via recursive factorizing permutations, *35th International Colloquium on Automata, Languages and Programming*, *Lecture Notes in Comput. Sci.* **5125** (2008), 634–645.

[112] R.H. Möhring and F.J. Radermacher, Substitution decomposition for discrete structures and connections with combinatorial optimization, *Annals of Discrete Math.* **19** (1984), 257–356.

[113] C.T. Hoàng, *A class of perfect graphs*, MSc Thesis, School of Computer Science, McGill University, Montreal, Canada, 1983.

[114] B. Jamison and S. Olariu, A tree representation for P_4-sparse graphs, *Discrete Appl. Math.* **35**(2) (1992), 115–129.

[115] B. Jamison and S. Olariu, Recognizing P_4-sparse graphs in linear time, *SIAM J. Comput.* **21**(2) (1992), 381–406.

[116] B. Jamison and S. Olariu, Linear time optimization algorithms for P_4-sparse graphs, *Discrete Appl. Math.* **61**(2) (1995), 155–175.

[117] C.T. Hoàng, *Perfect graphs*, PhD thesis, School of Computer Science, McGill University, Montreal, Canada, 1985.

[118] W.H. Cunningham, Decomposition of directed graphs, *SIAM J. Algebraic and Discrete Meth.* **3** (1982), 214–228.

[119] E. Dahlhaus, Parallel algorithms for hierarchical clustering and applications to split decomposition and parity graph recognition, *J. Algorithms* **36** (2000), 205–240.

[120] P. Charbit, F. de Montgolfier, and M. Raffinot, A simple linear time split decomposition algorithm of undirected graphs, CoRR abs/0902.1700, 2009.

[121] L. Babel and S. Olariu, On the p-connectedness of graphs—A survey, *Discrete Appl. Math.* **95** (1999), 11–33.

[122] R.E. Tarjan, Decomposition by clique separators, *Discrete Math.* **55** (1985), 221–232.

[123] S.H. Whitesides, A method for solving certain graph recognition and optimization problems, with applications to perfect graphs, In *Topics on Perfect Graphs*, Berge, C. and V. Chvátal (Eds.), North-Holland, Amsterdam, the Netherlands, 1984.

[124] A. Brandstädt, V. Giakoumakis, and F. Maffray, Clique-separator decomposition of hole-free and diamond-free graphs, *Discrete Appl. Math.* **160** (2012), 471–478.

[125] V.E. Alekseev, On easy and hard hereditary classes of graphs with respect to the independent set problem, *Discrete Appl. Math.* **132** (2004), 17–26.

[126] D. Lokshtanov, M. Vatshelle, and Y. Villanger, Independent set in P_5-free graphs in polynomial time, Tech. Report 2013, *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms*.

[127] E. Howorka, A characterization of distance-hereditary graphs, *Quart. J. Math. Oxford* Ser. **2**(28) (1977), 417–420.

[128] H.-J. Bandelt and H.M. Mulder, Distance-hereditary graphs, *J. Combin. Theor. (B)* **41** (1986), 182–208.

[129] P.L. Hammer and F. Maffray, Completely separable graphs, *Discrete Appl. Math.* **27** (1990), 85–99.

[130] H.-J. Bandelt, A. Henkmann, and F. Nicolai, Powers of distance-hereditary graphs, *Discrete Math.* **145** (1995), 37–60.

[131] A. D'Atri and M. Moscarini, Distance-Hereditary Graphs, Steiner Trees, and Connected Domination, *SIAM J. Comput.* **17** (1988), 521–538.

[132] H.-J. Bandelt and V.D. Chepoi, Metric graph theory and geometry: A survey, *Contemporary Mathematics* 453, 2006, *Surveys on Discrete and Computational Geometry, Twenty Years Later, AMS-IMS-SIAM Joint Summer Conference*, Snowbird, UT, J.E. Goodman, J. Pach, and R. Pollack (Eds.), American Mathematical Society, June 18–22, 2006, 49–86.

[133] A. Brandstädt and F.F. Dragan, A linear-time algorithm for connected r-domination and Steiner Tree on distance-hereditary graphs, *Netw.* **31** (1998), 177–182.

[134] D.C. Kay and G. Chartrand, A characterization of certain ptolemaic graphs, *Canad. J. Math.* **17** (1965), 342–346.

[135] E. Howorka, A characterization of ptolemaic graphs, *J. Graph Theor.* **5** (1981), 323–331.

[136] R. Uehara and T. Uno, Laminar structure of ptolemaic graphs and its applications, Proc. ISAAC 2005, X. Deng, D. Du (Eds.), *Lecture Notes in Comp. Sci.* **3827** (2005), 186–195; *Discrete Appl. Math.* **157** (2009), 1533–1543.

[137] A. Buneman, A note on the metric properties of trees, *J. Comb. Theory (B)* **1** (1974), 48–50.

[138] E. Howorka, On metric properties of certain clique graphs, *J. Comb. Theory (B)* **27** (1979), 67–74.

[139] C.W. Bachman, Data structure diagrams, *Data Base* **1**(2) (1969), 4–10.

[140] Y.E. Lien, On the equivalence of database models, *J. ACM* **29**(2) (1982), 333–363.

[141] M. Yannakakis, Algorithms for Acyclic Database Schemes, In *Proc. of Int. Conf. on Very Large Data Bases*, C. Zaniolo, C. Delobel (Eds.), Cannes, France, 1981, 82–94.

[142] N. Robertson and P.D. Seymour, Graph minors. III. Planar tree-width, *J. Comb. Theor.(B)* **36** (1984), 49–64.

[143] N. Robertson and P.D. Seymour, Graph width and well-quasi ordering: A survey, *Progress in Graph Theory*, J. Bondy and U. Murty (Eds.), Academic Press, New York, 1984, 399–406.

[144] N. Robertson and P.D. Seymour, Graph minors—A survey, *Surveys in Combinatorics*, I. Anderson (Ed.), London Mathematical Society, Lecture Note Series 103, Invited papers for the 10th British Combinatorial Conference, Cambridge University Press, 1985, 153–171.

[145] N. Robertson and P.D. Seymour, Graph minors. II. Algorithmic aspects of tree width, *J. Algorithms* **7** (1986), 309–322.

[146] T. Kloks, Treewidth—Computations and approximations, *Lecture Notes in Comput. Sci.* **842** (1994), 1–209.

[147] D.J. Rose, On simple characterizations of k-trees, *Discrete Math.* **7** (1974), 317–322.

[148] S. Arnborg, J. Lagergren, and D. Seese, Easy problems for tree-decomposable graphs, *J. Algorithms* **12** (1991), 308–340.

[149] B. Courcelle, The monadic second-order logic of graphs III: Tree-decompositions, minor and complexity issues, *Informatique Theorique et Applications* **26** (1992), 257–286.

[150] H.L. Bodlaender, A linear time algorithm for finding tree-decompositions of small treewidth, *SIAM J. Comput.* **25** (1996), 1305–1317.

[151] M. Thorup, All structured programs have small tree width and good register allocation, *Information and Computation* **142**(2) (1988), 159–181.

[152] Y. Dourisboure and C. Gavoille, Tree-decompositions with bags of small diameter, *Discrete Math.* **307** (2007), 2008–2029.

[153] B. Courcelle, J. Engelfriet, and G. Rozenberg, Handle-rewriting hypergraph grammars, *J. Comput. Syst. Sci.* **46** (1993), 218–270.

[154] B. Courcelle and S. Olariu, Upper bounds to the clique width of graphs, *Discrete Appl. Math.* **101** (2000), 77–114.

[155] D.G. Corneil and U. Rotics, On the relationship between clique-width and treewidth, *Internat. Workshop on Graph-Theoretic Concepts in Computer Science*, *Lecture Notes in Comput. Sci.* **2204** (2001), 78–90; *SIAM J. Computing* **34** (2005), 825–847.

[156] B. Courcelle, J.A. Makowsky, and U. Rotics, Linear time solvable optimization problems on graphs of bounded clique width, *Theor. Comput. Syst.* **33** (2000), 125–150.

[157] W. Espelage, F. Gurski, and E. Wanke, How to solve NP-hard graph problems on clique-width bounded graphs in polynomial time, *Internat. Workshop on Graph-Theoretic Concepts in Computer Science*, *Lecture Notes in Comput. Sci.* **2204** (2001), 117–128.

[158] D. Kobler and U. Rotics, Edge dominating set and colorings on graphs with fixed clique-width, *Discrete Appl. Math.* **126** (2002), 197–221.

[159] M.U. Gerber and D. Kobler, Algorithms for vertex partitioning problems on graphs with fixed clique-width, *Theor. Comput. Sci.* **1–3** (2003), 719–734.

[160] M.C. Golumbic and U. Rotics, On the clique-width of perfect graph classes, *Int. J. Foundations Comput. Sci.* **11** (2000), 423–443.

[161] A. Brandstädt and V.V. Lozin, On the linear structure and clique width of bipartite permutation graphs, RUTCOR Research Report, Rutgers University, New Brunswick, NJ, 29–2001 (2001); *Ars Combinatoria* (2003) 273–281.

[162] A. Brandstädt, F.F. Dragan, H.-O. Le, and R. Mosca, New graph classes of bounded clique width, *Theor. Comput. Syst.* **38** (2005), 623–645.

[163] A. Brandstädt, J. Engelfriet, H.-O. Le, and V.V. Lozin, Clique-width for four-vertex forbidden subgraphs, *Theor. Comput. Syst.* **39** (2006), 561–590.

[164] A. Brandstädt, Hoàng-Oanh Le, and R. Mosca, Gem- and co-gem-free graphs have bounded clique width, *Internat. J. Foundat. Computer Science* **15** (2004), 163–185.

[165] A. Brandstädt, Hoàng-Oanh Le, and R. Mosca, Chordal co-gem-free graphs and (P_5,gem)-free graphs have bounded clique width, *Discrete Appl. Math.* **145** (2005), 232–241.

[166] H.-O. Le, Contributions to clique-width of graphs, Dissertation, University of Rostock, Germany, 2003.

[167] J.A. Makowsky and U. Rotics, On the clique-width of graphs with few P_4's, *Int. J. Foundat. Comput. Sci.* **10** (1999), 329–348.

[168] M. Kamiński, V.V. Lozin, and M. Milanič, Recent developments on graphs of bounded clique-width, *Discrete Appl. Math.* **157** (2009), 2747–2761.

[169] M.R. Fellows, F.A. Rosamond, U. Rotics, and S. Szeider, Clique-width is NP-complete, *SIAM J. Alg. Discr. Math.* **23**(2) (2009), 909–939.

[170] D.G. Corneil, M. Habib, J.M. Lanlignel, B. Reed, and U. Rotics, Polynomial time recognition of clique-width ≤ 3 graphs, *Proceedings of LATIN*, *Lecture Notes in Comput. Sci.* **1776** (2000), 126–134.

[171] E. Wanke, k-NLC graphs and polynomial algorithms, *Discrete Appl. Math.* **54** (1994), 251–266.

[172] Ö. Johansson, Clique decomposition, NLC decomposition, and modular decomposition—Relationships and results for random graphs, *Congressus Numerantium* **132** (1998), 39–60.

[173] F. Gurski and E. Wanke, Line graphs of bounded clique-width, *Discrete Math.* **307** (2007), 2734–2754.

[174] V. Limouzy, F. de Montgolfier, and M. Rao, NLC_2 recognition and isomorphism, Proceedings of WG 2007, *Lecture Notes in Comput. Sci.* **4769** (2007), 86–98.

[175] S.-I. Oum, Approximating rank-width and clique-width quickly, *ACM Transactions on Algorithms* **5** (2008), 1–20.

[176] S.-I. Oum and P.D. Seymour, Approximating clique-width and branch-width, *J. Comb. Theory (B)* **96** (2006), 514–528.

[177] R. Uehara, S. Toda, and T. Nagoya, Graph isomorphism completeness for chordal bipartite graphs and strongly chordal graphs, *Discr. Appl. Math.* **145** (2005), 479–482.

[178] A. D'Atri, M. Moscarini, and H.M. Mulder, On the isomorphism problem for distance-hereditary graphs, Econometric Institute Tech. Report EI9241, A Rotterdam School of Economics, 1992.

[179] S.-I. Nakano, R. Uehara, and T. Uno, A new approach to graph recognition and applications to distance-hereditary graphs, *J. Comput. Sci. Technol.* **24**(3) (2009), 517–533.

[180] M. Grötschel, L. Lovász, and A. Schrijver, Polynomial algorithms for perfect graphs, *Ann. Discr. Math.* **21** (1984), 325–356.

[181] M. Grötschel, L. Lovász, and A. Schrijver*, Geometric Algorithms and Combinatorial Optimization*, Springer, 1988.

[182] H. Müller, Hamilton circuits in chordal bipartite graphs, *Discr. Math.* **156** (1996), 291–298.

[183] P. Damaschke, H. Müller, and D. Kratsch, Domination in convex and chordal bipartite graphs, *Inf. Proc. Letters* **36** (1990), 231–236.

[184] K. Cameron, Induced matchings, *Discr. Appl. Math.* **24** (1989), 97–102.

[185] L.J. Stockmeyer and V.V. Vazirani, NP-completeness of some generalizations of the maximum matching problem, *Inform. Process. Lett.* **15** (1982), 14–19.

[186] K. Cameron, R. Sritharan, and Y. Tang, Finding a maximum induced matching in weakly chordal graphs, *Discrete Math.* **266** (2003), 133–142.

[187] A. Brandstädt and C.T. Hòang, Maximum induced matching for chordal graphs in linear time, *Algorithmica* **52**(4) (2008), 440–447.

[188] A. Brandstädt, A. Leitert, and D. Rautenbach, Efficient dominating and edge dominating sets for graphs and hypergraphs, extended abstract in: *Proceedings of ISAAC*, Taiwan, 2012; LNCS 7676, 267–277.

[189] N. Biggs, Perfect codes in graphs, *J. Combinatorial Theor. (B)* **15** (1973), 289–296.

[190] C.-C. Yen and R.C.T. Lee, The weighted perfect domination problem and its variants, *Discrete Appl. Math.* **66** (1996), 147–160.

[191] C.L. Lu and C.Y. Tang, Weighted efficient domination problem on some perfect graphs, *Discr. Appl. Math.* **117** (2002), 163–182.

[192] D.M. Cardoso, N. Korpelainen, and V.V. Lozin, On the complexity of the dominating induced matching problem in hereditary classes of graphs, *Discr. Appl. Math.* **159** (2011), 521–531.

[193] C.L. Lu, M.-T. Ko, and C.Y. Tang, Perfect edge domination and efficient edge domination in graphs, *Discr. Appl. Math.* **119** (2002), 227–250.

[194] A. Brandstädt, C. Hundt, and R. Nevries, Efficient edge domination on hole-free graphs in polynomial time, Extended abstract in: *Conference Proceedings LATIN*, LNCS **6034** (2010), 650–661.

[195] F. Nicolai, Strukturelle und algorithmische Aspekte distanz-erblicher Graphen und verwandter Klassen, Dissertation, Gerhard-Mercator-Universität Duisburg, Germany, 1994.

[196] F. Nicolai, Hamiltonian problems on distance-hereditary graphs, Schriftenreihe des Fachbereichs Mathematik der Universität Duisburg SM-DU-255 (1994); corrected version 1996.

[197] R.-W. Hung, S.-C. Wu, and M.-S. Chang, Hamiltonian cycle problem on distance-hereditary graphs, *J. Inform. Sci. Engg.* **19** (2003), 827–838.

[198] S.-Y. Hsieh, C.-W. Ho, T.-S. Hsu, and M.-T. Ko, The Hamiltonian problem on distance-hereditary graphs, *Discr. Appl. Math.* **154** (2006), 508–524.

[199] R.-W. Hung and M.-S. Chang, Linear-time algorithms for the Hamiltonian problems on distance-hereditary graphs, *Theor. Comput. Sci.* **341** (2005) 411–440.

[200] H. Müller and F. Nicolai, Polynomial time algorithms for the Hamiltonian problems on bipartite distance-hereditary graphs, *Inf. Proc. Lett.* **46** (1993), 225–230.

[201] M.S. Chang, S.C. Wu, G.J. Chang, and H.G. Yeh, Domination in distance-hereditary graphs, *Discrete Appl. Math.* **116** (2002), 103–113.

[202] F. Nicolai and T. Szymczak, Homogeneous sets and domination: A linear time algorithm for distance-hereditary graphs, Schriftenreihe des Fachbereichs Mathematik der Universität Duisburg SM-DU-336 (1996); *Netw.* **37** (2001), 117–128.

[203] F. de Montgolfier, M. Soto, and L. Viennot, Treewidth and hyperbolicity of the Internet, *Proceedings of the 10th IEEE International Symposium on Networking Computing and Applications*, NCA 2011, August 25–27, 2011, Cambridge, MA. IEEE Computer Society, 2011, 25–32.

[204] M. Abu-Ata and F.F. Dragan, Metric tree-like structures in real-life networks: An empirical study, Manuscript 2013.

[205] F.F. Dragan and E. Köhler, An approximation algorithm for the tree t-spanner problem on unweighted graphs via generalized chordal graphs, approximation, randomization, and combinatorial optimization. algorithms and techniques. *Proceedings of the 14th International Workshop, APPROX* 2011, and *15th International Workshop, RANDOM*, Princeton, NJ, August 17–19, 2011, *Lecture Notes in Computer Science* 6845, Springer, 171–183; *Algorithmica* (in print 2014).

[206] F.F. Dragan and I. Lomonosov, On compact and efficient routing in certain graph classes, *Discrete Appl. Math.* **155** (2007), 1458–1470.

[207] D. Lokshtanov, On the complexity of computing tree-length, *Discrete Appl. Math.* **158** (2010), 820–827.

[208] V.D. Chepoi, F.F. Dragan, I. Newman, Y. Rabinovich, and Y. Vaxes, Constant approximation algorithms for embedding graph metrics into trees and outerplanar graphs, *Discrete & Computational Geometry* **47** (2012), 187–214.

[209] A. Brandstädt, V.D. Chepoi, and F.F. Dragan, Distance approximating trees for chordal and dually chordal graphs, *J. Algorithms* **30** (1999), 166–184.

[210] V.D. Chepoi and F.F. Dragan, A note on distance approximating trees in graphs, *European J. Combin.* **21** (2000), 761–766.

[211] V.D. Chepoi, F.F. Dragan, B. Estellon, M. Habib, and Y. Vaxes, Diameters, centers, and approximating trees of δ-hyperbolic geodesic spaces and graphs, *Proceedings of the 24th Annual ACM Symposium on Computational Geometry*, June 9–11, 2008, College Park, MD, pp. 59–68.

[212] Y. Dourisboure, F.F. Dragan, C. Gavoille, and C. Yan, Spanners for bounded tree-length graphs, *Theor. Comput. Sci.* **383** (2007) 34–44.

[213] F.F. Dragan and M. Abu-Ata, Collective additive tree spanners of bounded tree-breadth graphs with generalizations and consequences, *SOFSEM: Theory and Practice of Computer Science*, *Lecture Notes in Comput. Sci.* **7741** (2013), 194–206.

[214] Y. Dourisboure, Compact routing schemes for generalised chordal graphs, *J. Graph Algorithms Appl.* **9** (2005), 277–297.

[215] F.F. Dragan, C. Yan, and I. Lomonosov, Collective tree spanners of graphs, *SIAM J. Discrete Math.* **20** (2006), 241–260.

[216] L. Cai and D.G. Corneil, Tree spanners, *SIAM J. Discrete Math.* **8** (1995), 359–387.

[217] Y. Emek and D. Peleg, Approximating minimum max-stretch spanning trees on unweighted graphs, *SIAM J. Comput.* **38** (2008), 1761–1781.

[218] M. Gromov, Hyperbolic groups, In *Essays in Group Theory*, S.M. Gersten (Ed.), MSRI Series **8** (1987), 75–263.

[219] Y. Shavitt and T. Tankel, On internet embedding in hyperbolic spaces for overlay construction and distance estimation, In *INFOCOM*, 2004.

[220] I. Abraham, M. Balakrishnan, F. Kuhn, D. Malkhi, V. Ramasubramanian, and K. Talwar, Reconstructing approximate tree metrics, *Proceedings of the 26th Annual ACM Symposium on Principles of Distributed Computing*, Portland, OR, August 12–15, 2007, ACM, pp. 43–52.

[221] R. Kleinberg, Geographic routing using hyperbolic space, In *INFOCOM*, 2007, pp. 1902–1909.

[222] G. Brinkmann, J. Koolen, and V. Moulton, On the hyperbolicity of chordal graphs, *Ann. Comb.* **5** (2001), 61–69.

[223] Y. Wu and Ch. Zhang, Hyperbolicity and chordality of a graph, *Electr. J. Comb.* **18** (2011), P43.

[224] H. Fournier, A. Ismail, and A. Vigneron, Computing the Gromov hyperbolicity of a discrete metric space, *CoRR abs/1210.3323* (2012), http://arxiv.org/abs/1210.3323.

[225] C. Gavoille and O. Ly, Distance labeling in hyperbolic graphs, In *ISAAC*, 2005, pp. 171–179.

[226] R. Krauthgamer and J.R. Lee, Algorithms on negatively curved spaces, In *Proceedings of the 47th FOCS*, Berkeley, CA, 2006, pp. 119–132.

[227] V. Chepoi and B. Estellon, Packing and covering δ-hyperbolic spaces by balls, In *APPROX-RANDOM*, 2007, pp. 59–73.

[228] V.D. Chepoi, F.F. Dragan, B. Estellon, M. Habib, Y. Vaxès, and Y. Xiang, Additive spanners and distance and routing labeling schemes for δ-hyperbolic graphs, *Algorithmica* **62** (2012), 713–732.

[229] F.F. Dragan, Tree-like structures in graphs: A metric point of view, In *Graph-Theoretic Concepts in Computer Science—39th International Workshop*, Lübeck, Germany, June 19–21, 2013, Springer, *Lecture Notes in Comp. Sci.* **8165**, 1–4.

[230] R. Agarwala, V. Bafna, M. Farach, B. Narayanan, M. Paterson, and M. Thorup, On the approximability of numerical taxonomy (fitting distances by tree metrics), *SIAM J. Comput.* **28** (1999), 1073–1085.

[231] M. Bădoiu, E.D. Demaine, M.T. Hajiaghayi, A. Sidiropoulos, and M. Zadimoghaddam, Ordinal embedding: Approximation algorithms and dimensionality reduction, In *Proceedings of the 11th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems*, Boston, MA, August 25–27, 2008, Springer, *Lecture Notes in Computer Science* 5171, 21–34.

[232] M. Bădoiu, P. Indyk, and A. Sidiropoulos, Approximation algorithms for embedding general metrics into trees, In *Proceedings of the 18th Annual ACM-SIAM Symposium on Discrete Algorithms*, New Orleans, LA, January 7–9, 2007, ACM/SIAM, 512–521.