Epilogue 2004

1. Introduction – Foundations and Applications

In this Epilogue chapter, we will take a "short tour" of some of the new results in algorithmic graph theory and perfect graphs. So many new research directions have been the subject of investigation since this book was first published in 1980, that it is impossible to survey them all. The algorithms and applications associated with structured families of graphs have grown to maturity in these 24 years. The literature has increased tenfold, and the world of perfect graphs has grown to include over 200 special graph classes. Perfect graphs now have their own Mathematical Reviews Classification 05C17, as so do geometric and intersection representations 05C62.

We present here a sample of the many results of the Second Generation of Algorithmic Graph Theory from the author's biased view. Necessarily, it must be only a small fraction of what would otherwise require a large sequel volume. Fortunately, the availability of several new books, listed earlier in the Prologue of this edition, can also aid the reader eager to pursue further exploration in this area.

The sections of this Epilogue are numbered to correspond with the chapters of the book. Our intention is, as with the former chapters, to send the reader back to the literature, laboratory and library to continue research.

Intersection Graphs

We saw many of the early uses of the intersection graph model in the sneak preview Section 1.3, in the application sections on permutation graphs, interval graphs, and elsewhere in the book. But the volume and scope of research in this general area has expanded significantly both from the modeling and algorithmic points of view. Some of these applications include mobile frequency assignment (Osput and Roberts [1983]), pavement deterioration analysis (Gattass and

Nemhauser [1981]), relational databases (see Golumbic [1988]), evolutionary trees (see Waterman [1995]), physical mapping of DNA (Goldberg, et al. [1995], Golumbic, Kaplan and Shamir [1994]), container ship stowage (Avriel, Penn and Shpirer [2000]), and VLSI circuit design (Dagan, Golumbic and Pinter [1988]). Intersection graphs have become a necessary and important tool for solving real-world problems. McKee and McMorris [1999] is devoted to the topic of intersection graph models and their application. We will present several other examples in this new Epilogue chapter.

Temporal Reasoning

One of the "traditional" applications of interval graphs is reasoning about time intervals, which started with the original question of Hajös (Section 8.1, page 171). Temporal reasoning is an essential part of many applications in artificial intelligence. Given a set of explicit relationships between certain events, we would like to be able to infer additional relationships which are implicit in those given. For example, the transitivity of "before" and "contains" may allow us to derive information regarding the sequence of events. Seriation problems, like the example in Section 8.4, ask for a mapping of temporal events onto the time line such that all the given relations are satisfied, that is, a consistent scenario. Similarly, there are problems of scheduling, planning, and story understanding in which one is interested in constructing a time line where each particular event or phenomenon or task corresponds to an interval representing its duration.

Allen [1983] introduced a model for temporal reasoning using the thirteen primitive interval relations obtained by considering all possible orderings of their four endpoints. Several authors working in AI have studied and adapted Allen's model further, and have incorporated such models into reasoning systems. The paper by Golumbic and Shamir [1993] has provided a bridge linking some of these temporal reasoning notions from the AI community with those of the combinatorics community. Their approach has been to simplify Allen's model in order to study its complexity using graph theoretic techniques. We refer the reader to Golumbic [1999] which is a survey paper[1] on the topic, written in the same spirit as this book. It describes a number of directions of current work on reasoning about time, many of which employ graph algorithms.

2. The Design of Efficient Algorithms

Maximum Network Flow Problem

Progress on lowering the computational complexity of the maximum network flow (MAXFLOW) problem was presented in Table 2.1 as one of several illustrations

[1] This survey paper also includes some of the author's newest illustrative stories, "Goldie and the Four Bears", "Will Allan get to Judy's in time?", and "Five Autonomous Golumbic Women".

of closing the gap between the best known algorithms and the lower complexity bound. Further improvement for MAXFLOW have been given by Goldberg and Tarjan [1988] $O(ne\log(n^2/e))$, King, Rao and Tarjan [1994] $O(ne + n^{2+\varepsilon})$ and $O(ne\log_{e/(n\log n)} n)$, Philips and Westbrook [1993] $O(ne\log_{e/n} n + n^2(\log n)^{2+\varepsilon})$. We refer the interested reader to Cook, Cunningham, Pulleyblank and Schrijver [1998], Corman, Leiserson Rivest, second edition [2001], Goldberg [1998] and Johnson and McGeoch [1993].

Although it is not a central topic in studying perfect graphs, maximum network flow algorithms do play an important role in certain optimization problems on perfect graphs. For example, we invoked its complementary relative MINFLOW at the end of Section 5.7 when sketching a polynomial method for finding the stability number $\alpha(G)$ of a cocomparability graph. This method has subsequently been used as part of the algorithm by Narasimhan and Manber [1992] for the stability number of tolerance graphs.

Graph Sandwich Problems

In this book, we placed a major focus on the minimum coloring, maximum clique, and recognition problems for special families of graphs. Another important algorithmic direction has been the study of various completion problems. For example, the *minimum completion problem* requires adding a minimum number edges to an arbitrary graph G in order to obtain a new graph G' which satisfies the desired property Π, such as being an interval graph or a triangulated graph, see Garey and Johnson [1979], Yannakakis [1981]. One motivation for such completion problems is as a heuristic for coloring G, since $\chi(G) \leq \chi(G')$.

Another variation of the completion problem, called the *graph sandwich problem*, is defined by allowing only some of the nonedges to be eligible to be added to the original graph. Specifically, given a graph $G = (V, E)$ and a subset $E_0 \subseteq \overline{E}$ of (optional) nonedges, we ask whether there exists a completion $G' = (V, E')$ which satisfies the desired property Π, such that $E \subseteq E' \subseteq E \cup E_0$.

Sandwich problems arise in applications where only partial information about the graph is known. The interval graph sandwich problem was shown to be NP-complete by Golumbic and Shamir [1993]. It arises in molecular biology in problems of physical mapping of DNA and in problems of temporal reasoning, similar in spirit to the early work described in Applications 8.2 and 8.3 on pages 182–183. For example, see Atkins and Middendorf [1996] and Golumbic, Kaplan and Shamir [1994].

Golumbic, Kaplan and Shamir [1995] investigates graph sandwich problems for other special families of graphs. Specifically, the sandwich problem is polynomial for split graphs, cographs, and threshold graphs, but is NP-complete for chordal graphs, proper interval graphs, comparability graphs, permutation graphs, and others (see also Golumbic and Trenk [2004, Sections 4.7–4.8]).

There is also a body of literature on graph modification problems, where one may add edges and delete edges to transform a graph G into a modified graph G' satisfying a property Π. The problem is to minimize the total number of additions and deletions, thus changing the edge set as little as possible. For a good survey and introduction to this area, see Natanzon, Shamir and Sharan [2001].

3. Perfect Graphs

The Strong Perfect Graph Conjecture/Theorem

In May of 2002, the announcement was made that the Strong Perfect Graph Conjecture (SPGC) of Claude Berge had been proven by the team of researchers consisting of Maria Chudnovsky, Neil Robertson, Paul Seymour and Robin Thomas [2002]. This most important result, after 42 years, is currently submitted for publication.

The term *Berge graph* has been defined as an undirected graph which contains neither an odd chordless cycle C_{2k+1} (an odd hole) nor its complement $\overline{C}_{2k+1}$ (for $k \geq 2$) (an odd antihole) as an induced subgraph. Thus, what should now be called the *Strong Perfect Graph Theorem* states that an undirected graph is perfect if and only if it is a Berge graph, the "only if" direction being immediate.

During the decades preceding this solution, a large body of research developed involving the structure of minimally imperfect graphs, (see Brandstädt, Le and Spinrad [1999, Chapter 14]). The spin-off effect of these investigations has been the birth of many new children in the world of perfect graphs, both new problems and a generation of young researchers. The collections edited by Berge and Chvátal [1984] and Ramirez-Alfonsin and Reed [2001] provide a good cross-section of the work in this area.

Although proving the SPGC was a major mathematical challenge rather than an algorithmic one, it raised several related interesting algorithmic questions: Is there a polynomial time algorithm which recognizes whether or not an undirected graph G has an odd chordless cycle of length ≥ 5? Is there a polynomial time algorithm which recognizes Berge graphs? The second problem has been solved, by Chudnovsky and Seymour [2002], Chudnovsky, Cornuéjols, Liu, Seymour and Vušković [2002], Cornuéjols, Liu and Vušković [2002], and awaits publication. Combining this algorithm with the Strong Perfect Graph Theorem shows that *there is a polynomial time algorithm which recognizes whether or not a graph is perfect*. A solution to the first question has so far not been found (as of spring 2003), but when/if a solution is found, it will give an alternate solution to the second question by applying it to the graph and its complement.

Finally, Vashek Chvátal maintains a *perfect website*[2] which contains a long list of references and historical notes. A useful technical report by Hougardy [1998]

[2] http://www.cs.rutgers.edu/~chvatal/perfect/problems.html

lists, for each ordered pair (C_1, C_2) of 96 classes of perfect graphs, whether the class C_1 is a subset of the class C_2, and if this is not the case, provides and example of a graph which is in C_1 but not in C_2. An on-line tool for checking the class containment for a larger set of more than 200 graph classes has been developed at Rostock University[3]. It also contains the known complexity results for recognition of the classes, the maximum stable set and domination problems. Several unresolved containments and complexities remain as open problems. Figure 13.3, at the end of this Epilogue (p. 305), gives a complete hierarchy of some of the main classes of perfect graphs, ordered by inclusion (reprinted from Golumbic and Trenk [2004] where it appears together with separating examples).

Stable Sets

On the stable set problem for perfect graphs, there has also been progress. Grötschel, Lovász and Schrijver [1981] have shown that the ellipsoid method of solving linear programming problems can be applied to obtain a polynomial algorithm to find maximum stable sets and minimum colorings for perfect graphs. Also, since G is perfect if and only if its complement $\overline{G}$ is perfect, this same approach can be used to find maximum cliques and minimum clique covers.

The major importance of this result is that it generalizes what had been known for many classes of perfect graphs. Although the complexity of the algorithm is polynomial, it may not be practical to implement. As the authors point out, it is not intended to compete with the special purpose algorithms designed to solve these problems for interval graphs, cocomparability graphs, triangulated graphs, and other classes of perfect graphs which so often arise in applications. For further reading on algorithms for the stable set problem and the clique problem, see Hertz [1995] and Johnson and Trick [1996].

4. Triangulated Graphs – Chordal Graphs

Throughout this book, we have followed the French tradition by using the name *triangulated graph* for an undirected graph containing no chordless cycle C_k $(k \geq 4)$. In this section, however, we will use its popular synonym *chordal graph*. Two important variations of chordal graphs will be briefly presented here, namely the strongly chordal and the weakly chordal graphs. As their names indicate, every strongly chordal graph is chordal, and every chordal graph is weakly chordal. Weakly chordal graphs are also perfect graphs.

[3] http://wwwteo.informatik.uni-rostock.de/isgci

Strongly Chordal Graphs

Strongly chordal graphs, introduced by Farber [1985], specialize chordal graphs in several ways. They are characterized by several equivalent definitions using chords of a cycle, forbidden subgraphs and elimination orderings. We will also encounter these graphs later in Section 12 of this Epilogue in relation to chordal bipartite graphs.

Let $C = [u_1, u_2, \ldots, u_{2k}, u_1]$ be a cycle of even length $2k \geq 6$. A chord $u_i u_j \in E$ is called an *odd chord* if one of i and j is even and the other is odd, that is, it divides C into two even length cycles. A graph $G = (V, E)$ is defined to be *strongly chordal* if it is chordal and every cycle of even length greater than or equal to 6 has an odd chord. For example, referring to Figure F1 on page 275, the graphs G_2, G_3, G_4 and G_5 are strongly chordal, however, the others are not strongly chordal, as follows: G_1 and G_6 are not chordal so they are not strongly chordal; the graph G_7 is chordal, but the 6-cycle going around the outside of the graph has no odd chord, so the graph is not strongly chordal.

The graph G_7 is often called the 3-*sun* and is one of a family of forbidden subgraphs characterizing strongly chordal graphs. The *k-sun* S_k ($k \geq 3$) consists of $2k$ vertices, a stable set $X = \{x_1, x_2, \ldots, x_k\}$ and a clique $Y = \{y_1, y_2, \ldots, y_k\}$ and edges $E_1 \cup E_2$ where $E_1 = \{x_1y_1, y_1x_2, x_2y_2, y_2x_3, \ldots, x_ky_k, y_kx_1\}$ forms the *outer cycle* and $E_2 = \{y_iy_j \mid i \neq j\}$ forms the inner clique. The suns are split graphs, so they are chordal by Theorem 6.3, but they are not strongly chordal since the outer cycle has no odd chord.

A *trampoline* of order k ($k \geq 3$) is a graph obtained from a k-cycle C by adding for each edge of C a new vertex adjacent only to the two endpoints of that edge, and then adding enough chords to C to make it chordal. A *complete trampoline* is one in which all the chords are added to the cycle C, making it a clique and identical to the k-sun S_k. It is not difficult to show that a chordal graph which contains an induced trampoline also contains a (smaller) complete trampoline.

A vertex x is called *simple* if for every pair of neighbors y and z of x, either $N(y) \subseteq N(z)$ or $N(z) \subseteq N(y)$. An ordering of the vertices $[v_1, v_2, \ldots, v_n]$ is called a *simple elimination ordering* for G if v_i is a simple vertex in the induced subgraph H_i, for all i, where $H_i = G_{\{v_i, \ldots, v_n\}}$ is the subgraph remaining after $v_1, \ldots, v_{i-1}$ have been eliminated. Note that the 3-sun S_3 (G_7 on page 275) has no simple vertex, so it does not have a simple elimination ordering.

A *strong elimination ordering* is defined to be an ordering of the vertices $[v_1, v_2, \ldots, v_n]$ where, for all $i < j < k < \ell$, if $v_iv_k, v_iv_\ell, v_jv_k \in E$ then $v_jv_\ell \in E$. It is an easy exercise to verify that simple elimination orderings and strong elimination orderings are special cases of perfect elimination orderings.

The next Theorem, due to Farber [1983], provides the following characterizations of strongly chordal graphs:

Theorem 13.1. The following conditions are equivalent for an undirected graph $G = (V, E)$:
(i) G is strongly chordal.
(ii) G has a simple elimination ordering.
(iii) G has a strong elimination ordering.
(iv) G is chordal and sun-free.
(v) G is chordal and trampoline-free.

Strongly chordal graphs are closely related to the class of chordal bipartite graphs (Section 12.4, page 261). We will present this connection below in Theorems 13.15 and 13.17. For further reading on strongly chordal graphs, and additional characterizations, see Brandstädt, Le and Spinrad [1999] and McKee and McMorris [1999].

Weakly Chordal Graphs

Hayward [1985] introduced the class of *weakly chordal* graphs (also called *weakly triangulated*) as those having no induced subgraph isomorphic to C_n or to $\overline{C}_n$ for $n \geq 5$. The class of weakly chordal graphs contains the class of chordal graphs, since $C_5 = \overline{C}_5$ and $\overline{C}_n$ contains induced copies of C_4 for $n \geq 6$. Also, the *weakly chordal graphs are perfect graphs*. This result now follows immediately from the Strong Perfect Graph Theorem, however, the first proof was obtained by combining a result by Chvátal [1985], that neither a minimally imperfect graph G nor its complement $\overline{G}$ can contain a "star-cutset", with a result by Hayward [1985], that if G is a weakly chordal graph (with at least 3 vertices) then either G or $\overline{G}$ must contain a "star-cutset".

We call vertices x and y a *two-pair* if every chordless path between x and y has exactly two edges. The weakly chordal graphs have been characterized using two-pairs as follows.

Theorem 13.2. The following are equivalent:
(i) G is a weakly chordal graph.
(ii) Every induced subgraph of G is either a clique or has a two-pair.
(iii) If edges are repeatedly added between two-pairs in G, the result is eventually a clique.

The implication (ii) $\Longrightarrow$ (i) follows from the observation that nonadjacent vertices in C_n or $\overline{C}_n$ ($n \geq 5$) are not a two-pair. The implication (i) $\Rightarrow$ (ii) is due to Hayward, Hoang and Maffray [1990], and (i) $\Longleftrightarrow$ (iii) is due to Spinrad and Sritharan [1995]. The latter equivalence also leads to an $O(n^4)$ recognition algorithm for weakly chordal graphs.

LexBFS

In Section 4.3, we presented lexicographic breadth first search (LexBFS), using it to obtain the linear time algorithm for recognizing chordal graphs. We also mentioned that maximum cardinality search (MCS) provides a conceptually and computationally simpler method for getting a perfect elimination scheme by changing the *label* of a vertex x from a "list" of the marked neighbors to a simple "counter" of those marked neighbors. Tarjan's proof that MCS orderings correctly recognize chordal graphs can be found in Golumbic [1984].

Fans and friends might have been temporarily disappointed when MCS seemed to make LexBFS obsolete, but not for long! LexBFS has become useful in other contexts, including recognizing proper interval graphs, recognizing asteroidal triple-free graphs and several other algorithms. There are also results for LexBFS on the powers of chordal graphs and on distance hereditary graphs which have no analogous result for MCS. See Brandstädt, Le and Spinrad [1999, Chapter 5] which also surveys characterizations of many perfect graph families in terms of special kinds of vertex orderings.

Intersection Graphs on Trees

Let T be a tree and let $\{T_i\}$ be a collection of subtrees (connected subgraphs) of T. We may think of the host tree T either (1) as a continuous model of a tree embedded in the plane, thus generalizing the real line from the one-dimensional case, or (2) as a finite discrete model of a tree, namely, a connected graph of vertices and edges having no cycles, thus generalizing the path P_k from the one dimensional case.

The distinction between these two models becomes important when measuring the size of the intersection of two subtrees. For example, in the continuous model (1), we might take the size of the intersection to be the Euclidean distance of a longest common path of the two subtrees. In the discrete model (2), we might count the number of common vertices or common edges. We use the expressions "nonempty intersection" and "vertex intersection" to mean sharing a vertex or point of T, and "nontrivial intersection" and "edge intersection" to mean sharing an edge or otherwise measurable segment of T.

Using this terminology, Theorem 4.8 (page 92) stated the following.

Theorem 4.8. A graph is the *vertex intersection graph* of a set of subtrees of a tree if and only if it is a chordal graph.

In contrast to this, Golumbic and Jamison [1985a] observed that the family of *edge intersection graphs* of subtrees of a tree yield all possible graphs, proving the following:

Theorem 13.3. Every graph can be represented as the edge intersection graph of substars of a star.

Table 1. Graph classes involving trees

Type of Interaction	Objects	Host	Graph Class
vertex intersection	subtrees	tree	chordal graphs
vertex intersection	subtrees	star	split graphs
edge intersection	subtrees	star	all graphs
vertex intersection	paths	path	interval graphs
vertex intersection	paths	tree	path graphs or VPT graphs
edge intersection	paths	tree	EPT graphs
containment	intervals	line	permutation graphs
containment	paths	tree	? (open question)
containment	subtrees	star	comparability graphs

Proof. Let $G = (V,E)$ be any graph, and let $E = \{e_1,\ldots,e_m\}$. Consider the star T formed by a central node u and leaves $\overline{e}_1,\ldots,\overline{e}_m$. Define the substar corresponding to v_i to be the substar T_i of T induced by $\{u\} \cup \{\overline{e}_\ell \mid v_i \in e_\ell\}$. Clearly, $v_iv_j \in E$ if and only if T_i and T_j share an edge, namely edge ue_k of T where $e_k = v_iv_j$. ∎

We will see below, in Theorem 13.6, that a graph is the *vertex intersection graph* of substars of a star if and only if it is a split graph.

Two different classes of intersection graphs also arise when considering simple paths (instead of subtrees) of an arbitrary host tree T. The *path graphs*, which we mentioned on page 94, are the subfamily of chordal graphs obtained as the "vertex intersection graphs of paths in a tree" and are also called *VPT graphs*. However, the graphs obtained as the "edge intersection graphs of paths in a tree", called *EPT graphs,* are not necessarily chordal. The class of EPT graphs are not perfect graphs, and the recognition problem for them is NP-complete, Golumbic and Jamison [1985a, 1985b]. See also Monma and Wei [1986] and Sysło [1985].

Table 1 summarizes the subtree graph classes we have discussed here and in Sections 13.5 and 13.6 below. A full treatment can be found in Golumbic and Trenk [2004, Chapter 11].

5. Comparability Graphs and the Dimension of Ordered Sets

Comparability Invariants

A graph can have many different transitive orientations, so there may be different partial orders with the same comparability graph. A property of partially

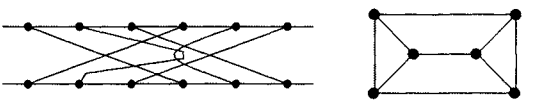

Fig.13.1. A function diagram and its intersection graph (which is isomorphic to $\overline{C}_6$).

ordered sets is called a *comparability invariant* if either all orders with a given comparability graph have that property, or none have that property. For example, we saw in Theorem 8.13 (page 187) and in Exercise 8.7 (page 194) that the properties of being a semiorder or an interval order are comparability invariants.

The dimension of a partial order is also a comparability invariant, that is, dim P = dim Q whenever P and Q have the same comparability graph G; hence, we can denote this common value by dim G. The proof we gave for Theorem 5.39 (page 139) is incomplete; a full correct proof can be found in Trotter [1992] or in Golumbic and Trenk [2004]. Those references may be consulted for further study on comparability invariant properties.

Function Diagrams: the Intersection Model for Cocomparability Graphs

Golumbic, Rotem and Urrutia [1983] have characterized the family of cocomparability graphs as the intersection graphs of function lines in a diagram which generalizes the matching diagrams (page 162) which represent permutation graphs. Their function diagrams are constructed as follows.

Let L_1 and L_2 be two horizontal lines. A continuous curve f connecting a point on L_1 with a point on L_2 is called a *function line* if, whenever two points (x,y) and (x',y') on f have the same horizontal value $y = y'$, the points must be equal, i.e., $x = x'$. A *function diagram* consists of L_1 and L_2 and a set of n function lines connecting points on L_1 and L_2. The function diagram in Figure 13.1 has six function lines. We note that a matching diagram is the special case in which the function lines are straight lines.

Consider the following special type of function diagram in which the curves are piecewise linear. Let $L_1, L_2, \ldots, L_{k+1}$ be horizontal lines each labeled from left to right by a permutation of the numbers $1, 2, \ldots, n$. For each i $(1 \leq i \leq n)$ the curve f_i consists of the union of the k straight line segments which join i on L_t with i on L_{t+1} $(1 \leq t \leq k)$. When $k = 1$, this is just a matching diagram; when $k \geq 2$, it is called the *concatenation* of k matching diagrams.

The following theorem is due to Golumbic, Rotem and Uruttia [1983], and a proof can also be found in Golumbic and Trenk [2004].

Theorem 13.4. The following are equivalent.

(i) G is the intersection graph of a function diagram.
(ii) G is a cocomparability.
(iii) G is the intersection graph of a concatenation of matching diagrams.

Moreover, if ℓ is the minimum value for which G is the intersection graph of a concatenation of ℓ matching diagrams, then $\dim G = \ell + 1$.

Containment Graphs

The *containment graph* $G = (V, E)$ of a collection $\mathcal{F}=\{S_i\}$ of distinct subsets of a set $\mathbf{S}$ has vertex set $V = \{1, \ldots, n\}$ and edge set $E = \{ij \mid$ either $S_i \subset S_j$ or $S_j \subset S_i\}$. A graph with such a representation is called a *containment graph*. The class of containment graphs is equivalent to the class of comparability graphs. Moreover, Golumbic and Scheinerman [1989] observed that every comparability graph can be represented as the containment graph of a collection of subtrees (substars) of a star. Dushnik and Miller [1941] characterized the containment graphs of intervals on the line as precisely those having partial order dimension 2, and from Chapter 7, we recall that these are equivalent to the permutation graphs. Generalizing interval containment, Golumbic and Scheinerman [1989] also showed the following.

Theorem 13.5. A graph G is the containment graph of rectilinear boxes[4] in d-space if and only if $\dim(G) \leq 2d$.

Yannakakis [1982] has shown that the complexity of determining whether an order P has dimension $\leq k$, for any fixed $k \geq 3$ is NP-complete. This answers the open problem stated in the footnote on page 138. A proof can be found in Mahadav and Peled [1995, Chapter 7]. Therefore, as a corollary of Theorem 13.5, we conclude that *the recognition problem for the containment graphs of boxes in the plane is NP-complete.*

As early as the Banff Conference in 1984, we posed the problem, "Characterize the containment orders of circles in the plane and their comparability graphs", see Rival [1985, page 583]. Progress on this question can be found in Fishburn [1988], Scheinerman and Wierman [1988], and Scheinerman [1992]. Sphere orders are the generalization to higher dimension, and are also found in the literature. Characterizing the containment graphs of paths in a tree is still an open problem.

New Complexities for Comparability Graphs, Transitive Orientation and Permutation Graphs

New algorithms have been found for recognizing comparability graphs and permutation graphs, based on fast modular decomposition of graphs. Modular decomposition is the recursive version of the method we saw in Section 5.2. The

[4] Boxes with sides parallel to the axes.

first of these new algorithms were due to Spinrad [1985]. McConnell and Spinrad [1999] show how to find an orientation F of an arbitrary graph $G = (V, E)$ such that F is a transitive orientation (TRO) of G if and only if G is a comparability graph. This is very good if there is other information guaranteeing that G is a comparability graph. However, this alone does not recognize comparability graphs, since the algorithm simply produces an orientation which is *not* transitive when G is *not* a comparability graph. Hence, to complete it to a recognition algorithm, one must test F to determine if it is transitive.

The complexity of their method uses $O(n+e)$ time to produce a linear ordering L_1 of the vertices which is then applied to E to produce an orientation F_1. It then uses $O(n^\alpha)$ time to test whether F_1 is transitive, where $O(n^\alpha)$ is the complexity to perform transitive closure or $n \times n$ matrix multiplication (currently $n^{2.376}$). In a similar fashion, they can produce another linear ordering L_2 of the vertices which if applied to $\overline{E}$ will produce an orientation F_2 which will be transitive if and only if G is a cocomparability graph. So the complexity of recognizing cocomparabilty graphs is currently also $O(n^\alpha)$.

Interestingly, their method allows recognizing permutation graphs in $O(n+e)$ time, by first producing L_1, L_2 and F_1 and calculating the in-degrees and out-degrees of F_1 and F_2, but without actually producing F_2, since otherwise the complexity will hit $O(n^2)$. These enable us to follow the construction in the proof of our Theorem 7.1 (pages 158–159), yielding a permutation representation for G, if G is a permutation graph, or a contradiction among the degrees if G is not a permutation graph.

6. Split Graphs

Theorem 4.8 stated that a graph is the (vertex) intersection graph of a set of subtrees of a tree if and only if it is a triangulated graph. McMorris and Shier [1983] give an analogous version for split graphs, which we recall are characterized as being both triangulated and cotriangulated (Theorem 6.3, page 151). If the host tree T is a star $K_{1,n}$, then each induced subtree consists of either a substar containing the central node or just a single leaf node. It is easy to see that the graphs obtained in this restriction are precisely the class of split graphs, as observed in McMorris and Shier [1983].

Theorem 13.6. A graph G is the vertex intersection graph of distinct induced subtrees of a star $K_{1,n}$ if and only if G is a split graph.

Proof. Recall that a graph G is a split graph if its vertices can be partitioned into a clique $K = \{x_1, \ldots, x_k\}$ and a stable set $S = \{y_1, \ldots, y_\ell\}$. If G is a split graph, consider the star T formed by a central node u and leaves $\overline{x}_1, \ldots, \overline{x}_k, \overline{y}_1, \ldots, \overline{y}_\ell$, where the subtree corresponding to $y_i \in S$ is the single

leaf $\bar{y}_i$ in T and the subtree corresponding to $x_i \in K$ consists of the substar of T induced by $\{u, \bar{x}_i\} \cup \{\bar{y}_j \mid y_j \in Adj(x_i)\}$. Clearly, this is an intersection representation for G. Conversely, if we are given a representation for G as the intersection graph of distinct induced substars of a star, then those substars containing the central node correspond to a clique in G and the remaining subtrees (the single leaves) correspond to a stable set of G. ∎

7. Permutation Graphs and Applications from Circuit Design

Matching diagrams, like those we studied in Section 7.4, are used in circuit design for channel routing problems where a set of numbered pins on the upper side of the diagram must be connected (electrically) to a set of pins on the lower side. The area between the upper and lower horizontals is called the *channel*. When a pair of line segments connecting matched pins intersect, they must be placed on different silicon layers, similar to the altitudes for the aircraft in Application 7.1. Thus, the minimum number of layers needed to realize the diagram equals the clique number $\omega(G)$ of its permutation graph G, the value of which can be calculated in $O(n \log n)$ time. We will discuss briefly two similar graph problems originally motivated by circuit design. The books by Lengauer [1990] and Shrwani [1995] give a comprehensive treatment of other VLSI design and routing algorithms.

Cell Flipping in Matching Diagrams

Golumbic and Kaplan [1998] have considered the following generalization of the channel routing problem above, which is motivated by "standard cell" technology. The numbers on each side of the channel are partitioned into consecutive subsequences, or *cells*, each of which can be left unchanged or flipped (i.e., reversed). This takes place at a stage where the cell placement on horizontal rows has already been performed, and the only remaining degree of freedom is replacing some of the cells with their "mirror image" with respect to the vertical axis, i.e., cell flipping. The questions asked are:

MINFLIP: For what choice of flippings will the resulting clique number be minimized?

MAXFLIP: For what choice of flippings will the resulting clique number be maximized?

For example, let the upper sequence be partitioned $[3,4,7],[2,6],[1,5,8]$ and let the lower sequence be $[6',2',5'],[4',1',7',8',3']$, where the brackets indicate cells. The clique number $\omega(G)$ is 4 with no flipping but is reduced to 3 if we flip $[2,6]$, or is increased to 5 if we flip $[2,6]$, $[6',2',5']$ and $[4',1',7',8',3']$.

The complexity of the MAXFLIP problem is $O(n^2)$ using a dynamic programming algorithm, whereas the MINFLIP problem is NP-complete (see Golumbic and Kaplan [1998]). When one side of the channel is fixed (no flipping on that side), the problem of finding a flipping for the other side which *maximizes* the clique number can be found in $O(n \log n)$ time. Verbin [2002] has shown the complexity of the one side flipping problem for *minimizing* the clique number to be NP-complete.

We note that the cell flipping problems could have been defined to minimize or maximize the stability number. It is clear, however, that these stability problems are computationally equivalent to our clique problems, since $\alpha(G) = \omega(\overline{G})$ and G is a permutation graph if and only if $\overline{G}$ is a permutation graph. For example, the 4 different one side flipping problems can be restated as follows: Given a permutation of the numbers $1, 2, \ldots, n$ partitioned into cells, find flippings which minimize/maximize the longest increasing/decreasing subsequence.

Trapezoid Graphs

Dagan, Golumbic and Pinter [1988] extended the notion of a permutation diagram by replacing each matching segment i-i' by a trapezoid obtained from 4 points a_i, b_i, c_i, d_i where interval $I_i = [a_i, b_i]$ lies on the upper line of the diagram and interval $I_i' = [c_i, d_i]$ lies on the lower line of the diagram. The intersection graphs of these trapezoid diagrams are called *trapezoid graphs*. Trapezoid graphs are a subclass of cocomparability graphs, and their associated (trapezoid) orders are precisely those having interval dimension 2 (see Trotter [1992]).

Langley [1995] and Ma and Spinrad [1994] gave polynomial time algorithms for recognizing trapezoid graphs, and Felsner, Müller and Wernisch [1997] have given optimal maxclique and maxstable set algorithms for the class. Further investigation of the class can be found in Cheah and Corneil [1996]. As we will mention below, trapezoid graphs are equivalent to the bounded bitolerance graphs.

Cographs and Factoring Read-Once Functions

An important subfamily of permutation graphs are the *complement reducible* graphs, or *cographs*, which were investigated by Corneil, Lerchs and Burlingham [1981] and Corneil, Perl and Stewart [1985]. Cographs can be defined recursively as follows: (1) a single vertex is a cograph; (2) the disjoint union of cographs is a cograph; (3) the complement of a cograph is a cograph. Alternately, they can be defined by a restricted type of decomposition, as in our Section 5.2 (page 111) where G_R would be either a clique or a stable set. One can recognize whether a graph G is a cograph by repeatedly decomposing it in this way, and in linear time

obtain a representation called its *cotree*. The cotree is useful in many algorithms for the class. The next theorem gives several characterizations of cographs.

Theorem 13.7. The following are equivalent for an undirected graph G.
(i) G is a cograph.
(ii) G is P_4-free.
(iii) For every subset X of vertices, either the induced subgraph G_X is disconnected or its complement $\overline{G}_X$ is disconnected.

Recognizing cographs and building the associated cotree has been recently used in an application involving multi-level logic synthesis. Golumbic and Mintz [1999] presented a factoring algorithm for general Boolean functions which is based on graph partitioning, and at the lower levels of the recursion, read-once functions are handled in a special manner. Read-once functions are very closely related to cographs, as we shall see.

A Boolean function F is called a *read-once function* if it has a factored form in which each variable appears exactly once. For example, the function $F_1 = aq + acp + ace$ is a read-once function since it can be factored into the formula $F_1 = a(q + c(p + e))$. The function $F_2 = ab + ac + bc$ is not a read-once function. Read-once functions have interesting special properties and account for a large percentage of functions which arise at the lower level real circuit applications. They have also gained recent interest in the field of computational learning theory.

Let $F = \alpha_1 + \cdots + \alpha_t$ be given in disjunctive normal form (sum-of-products), where each α_k is a product term, and let $V = \{v_1, \ldots, v_n\}$ be its set of variables. We consider the graph $\Gamma_F = (V, E)$ of F where $v_i v_j \in E$ whenever v_i and v_j appear together in some product term α_k. The function F is called *normal* if every clique of Γ_F is contained in one of the product terms. Our example F_2 above is not normal since Γ_{F_2} is a triangle, but all product terms are of size 2. The dual F^* of F is the function obtained from F by replacing products by sums and sums by products. By taking the disjunctive normal form of F^* one can construct the graph Γ_{F^*}. The next theorem is due to Gurvich [1991]:

Theorem 13.8. The following are equivalent:
(i) F is a read-once function.
(ii) Γ_F is P_4-free (i.e., a cograph) and F is a normal function.
(iii) $\overline{\Gamma}_F = \Gamma_{F^*}$.

Golumbic, Mintz and Rotics [2001] present a very fast method for recognizing and factoring read-once functions based on algorithms for cograph recognition and on checking normality.

8. Interval Graphs and Circular-arc Graphs

AT-Free Graphs

Three vertices v_1, v_2, v_3 in a graph $G = (V, E)$ form an *asteroidal triple* (*AT*) if, for all permutations i, j, k of $\{1, 2, 3\}$, there is a path from v_i to v_j which avoids using any vertex in the neighborhood $N(v_k) = v_k \cup Adj(v_k)$. An easy way to verify this for v_k is to delete $N(v_k)$ and test whether v_i and v_j remain in the same connected component of $G - N(v_k)$. It also follows from the definition that the three vertices of an asteroidal triple are pairwise nonadjacent. For example, the graph in Figure 8.15 (page 196) has 8 asteroidal triples[5].

A graph is called *asteroidal triple free* or *AT-free* if it contains no asteroidal triple. We saw in Theorem 8.4 (page 174) that interval graphs are characterized by being chordal and AT-free. Golumbic, Monma and Trotter [1984] proved the following result.

Theorem 13.9. Every cocomparability graph is an AT-free graph.

More recently, Corneil, Olariu and Stewart [1997, 1999] have given new algorithmic and structural results for AT-free graphs. Köhler [2000] also provides efficient recognition algorithms for the class.

Tolerance Graphs

Tolerance graphs were introduced by Golumbic and Monma [1982] to generalize some of the applications associated with interval graphs. Their motivation was the need to solve scheduling problems, more general than what we saw in Section 1.3, in which resources such as rooms, vehicles, support personnel, etc. may be needed on an exclusive basis, but where a measure of flexibility or tolerance would be allowed for sharing or relinquishing the resource when total exclusivity prevented a solution. The recent book by Golumbic and Trenk [2004] contains a thorough study of tolerance graphs and related topics.

An undirected graph $G = (V, E)$ is a *tolerance graph* if there exists a collection $\mathcal{I} = \{I_v\}_{v \in V}$ of closed intervals on the real line and an assignment of positive numbers $t = \{t_v\}_{v \in V}$ such that

$$vw \in E \Leftrightarrow |I_v \cap I_w| \geq \min\{t_v, t_w\}.$$

Here $|I_u|$ denotes the length of the interval I_u. The positive number t_v is called the *tolerance* of v, and the pair $\langle \mathcal{I}, t \rangle$ is called an *interval tolerance representation* of G. A tolerance graph is said to be *bounded* if it has a tolerance representation

[5] In this book, we incorrectly spelled these "astroidal" rather than asteroidal.

in which $t_v \leq |I_v|$ for all $v \in V$. Tolerance graphs generalize both interval graphs and permutation graphs, and in Golumbic and Monma [1982] it was shown that every bounded tolerance graph is a cocomparability graph. Golumbic, Monma and Trotter [1984] proved that tolerance graphs are perfect and are contained in the class of weakly chordal graphs.

Coloring bounded tolerance graphs in polynomial time is an immediate consequence of their being cocomparability graphs. Narasimhan and Manber [1992] use this fact (as a subroutine) to find the chromatic number of any (unbounded) tolerance graph in polynomial time, but not the coloring itself. Golumbic and Siani [2002] give an $O(qn + n \log n)$ algorithm for coloring a tolerance graph, given the tolerance representation with q vertices having unbounded tolerance (see Golumbic and Trenk [2004]). The complexity of recognizing tolerance graphs and bounded tolerance graphs remain open questions.

A variety of "variations on the theme of tolerance" in graphs have been defined and studied over the past years. By substituting a different "host" set instead of the real line and with a specified type for the subsets of that host instead of intervals, we obtain classes such as neighborhood subtree tolerance (NeST) graphs, tolerance graphs of paths on a tree or tolerance competition graphs. By changing the function *min* for a different binary function ϕ (for example, *max, sum, product*, etc.), we obtain a class that will be called *ϕ-tolerance* graphs. By replacing the measure of the length of an interval by some other measure μ of the intersection of the two subsets (for example, cardinality in the case of discrete sets, or number of branching nodes or maximum degree in the case of subtrees of trees), we could obtain yet other variations of tolerance graphs. When we restrict the tolerances to be 1 or ∞, we obtain the class of *interval probe graphs*. By allowing a separate leftside tolerance and rightside tolerance for each interval, various bitolerance graph models can be obtained. For example, Langley [1993] showed that *the bounded bitolerance graphs are equivalent to the class of trapezoid graphs*. Directed graph analogues to several of these models have also been defined and studied. For further study of tolerance graphs and related topics, we refer the reader to Golumbic and Trenk [2004].

Circular-Arc Graphs

In Section 8.6, we presented the early work on circular-arc graphs, the intersection graphs of arcs on a circle. The first polynomial time algorithm for recognizing circular-arc graphs was given by Tucker [1980] and had complexity $O(n^3)$. Over the years, more efficient algorithms were designed for the recognition problem first by Hsu [1995] in $O(ne)$ and Eshen and Spinrad [1993] in $O(n^2)$, and most recently by McConnell [2001, 2003] in $O(n + e)$ which is optimal. The coloring problem for circular-arc graphs was shown to be NP-

complete by Garey, Johnson, Miller and Papadimitriou [1980]. The maximum stable set and the maximum clique problems are polynomial, see Golumbic and Hammer [1988], Hsu and Spinrad [1995], Hsu [1985], Apostolico and Hambrus [1987].

9. Superperfect Graphs

Superperfect Noncomparability Graphs

Progress has been made on the question asked in Section 9.4, "When does superperfect equal comparability?" Gallai [1967] published a complete list of the minimal noncomparability graphs, that is, the noncomparability graphs with the property that removing any vertex makes the remaining subgraph a comparability graph, see also Berge and Chvátal [1984, p. 78]. Using this list, Andreae [1985] determined all of the minimal noncomparability graphs which are superperfect. One of these is the graph in our Figure 1.18 on p.18 which we *incorrectly* placed in the position of non-superperfect in Figure 9.9 on p. 212. This graph *is* superperfect and a simple proof is also given in Kloks and Bodlander [1992, Theorem 3.2].

In Corollary 9.8, we saw that for split graphs, G is a comparability graph if and only if G is superperfect. This motivated our asking the question of whether or not this equivalence holds for triangulated graphs or for cotriangulated graphs. Using his forbidden subgraph characterization of superperfect noncomparability graphs, Andreae [1985] answered this question with "false" for triangulated graphs and "true" for cotriangulated graphs, showing the following.

Theorem 13.10.

(i) For cotriangulated graphs, G is a comparability graph if and only if G is superperfect.

(ii) For triangulated graphs, G is a comparability graph if and only if G is superperfect and contains no induced subgraph isomorphic to G_1, G_2 or G_3 of Figure 13.2.

As Andreae [1985] points out, this list of the minimal superperfect noncomparability graphs could also be used to answer our equivalence question for other hereditary classes of graphs. In a similar investigation for k-trees,

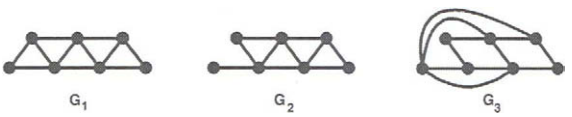

Fig.13.2. The minimal triangulated, superperfect noncomparability graphs.

Kloks and Bodlander [1992] gave a complete characterization, by means of forbidden subgraphs, of the superperfect 2-trees. The graph G_1 is a superperfect 2-tree. They also give an algorithm for testing superperfection in k-trees, whose complexity is linear for any fixed k, and can produce a complete forbidden subgraph characterization of superperfect k-trees.

10. Threshold Graphs

The undisputed authority on threshold graphs and related topics is the book by Mahadev and Peled [1995]. It is a masterpiece of clarity, and it presents a comprehensive coverage of Ferrers digraphs, threshold dimension, measures, weights, orders, enumeration, and a variety of generalizations, variations and specializations of threshold graphs. Any intention of updating or expanding upon Chapter 10 must therefore defer to their book.

New complexity results on the threshold dimension of a graph deserve mentioning. In Section 10.1, we defined the threshold dimension $\theta(G)$ of a graph G. Threshold graphs are those having threshold dimension at most 1, and they can be recognized in linear time. The complexity of determining $\theta(G)$ is NP-complete (page 223), and a stronger result is that determining whether $\theta(G) \leq k$, for any fixed $k \geq 3$, is NP-complete (Golumbic [1981], Yannakakis [1982]). So the remaining question, open for the next dozen years, was whether recognizing threshold dimension 2 could be done in polynomial time. A positive answer to this question follows from the next interesting result.

Two edges of G are said to *conflict* if their endpoints induce in G one of the forbidden subgraphs C_4, P_4 or $2K_2$ characterizing threshold graphs (see Theorem 10.7, page 227). The conflict graph G^* of G is defined by taking $V(G^*) = E(G)$ and by joining two vertices of G^* by and edge in $E(G^*)$ if and only if their corresponding edges in G conflict. For example, the conflict graph of a threshold graph is a stable set, and the conflict graph of C_4 is $2K_2$. We noted in Section 10.1, that we can take $\theta(G)$ to be the minimum number of threshold graphs needed to cover the edges of G. Thus, Chvátal and Hammer [1977] observed that such an edge covering of G leads to a valid coloring of G^*, that is, $\chi(G^*) \leq \theta(G)$ where each color represents a threshold graph in the edge covering. From this inequality, one can see that whenever G has threshold dimension 2, its conflict graph G^* must be a bipartite graph. The converse of this implication for $\theta(G) = 2$ was conjectured to hold by Ibaraki and Peled [1981], and it was finally proven by Raschle and Simon [1995], which we record as follows.

Theorem 13.11. A graph G has threshold dimension 2 if and only if its conflict graph G^* is a bipartite graph.

A proof of this theorem can be found in Mahadev and Peled [1995, Section 8.5]. Since constructing the conflict graph and testing whether it is 2-colorable can be done in polynomial time, we immediately obtain the following corollary.

Theorem 13.12. The problem of determining whether $\theta(G) \leq 2$ has polynomial time complexity.

Threshold Hypergraphs

In Section 10.4, we presented the definition of threshold hypergraph and gave an application in which they might be useful for the allocation of resources. Threshold hypergraphs are closely related to the class of threshold Boolean functions. We also posed a Research Problem (on page 233) giving three properties of r-regular hypergraphs, each being a generalization of threshold graphs to hypergraphs. Reiterman, et al. [1985] have shown that these three properties are different, and they have given a characterization of the most general of them (T_3) which are known as *shift stable* r-regular hypergraphs. Boros [1991] has generalized this further, giving a characterization of shift stable hypergraphs (not necessarily regular ones).

11. Circle Graphs

Circle graphs are the intersection graphs of chords of a circle. In Chapter 11, we called these "not so perfect graphs" rather as a joke, needing an excuse to include them in the book even though they are not perfect graphs. Circle graphs are an important and natural extension of permutation graphs, and their equivalence with overlap graphs raises their status even further. The recognition problem for circle graphs, which had been open, was solved independently by Bouchet [1987] and Gabor, Supowit, and Hsu [1989] who gave polynomial algorithms with complexity of $O(n^4)$ and $O(n^3)$, respectively. Subsequently, an $O(n^2)$ method was given by Spinrad [1994]. A further characterization of circle graphs appears in Bouchet [1994].

12. Chordal Bipartite Graphs

We defined the class of *chordal bipartite* graphs in Section 12.4 as those bipartite graphs which, for all $k > 4$, have no induced chordless cycle C_k. Thus, a chordal bipartite graph is not necessarily a chordal graph since the 4-cycle C_4

is chordal bipartite but not chordal[6]. Recalling the definition of weakly chordal graphs (Section 13.4 above), it is a simple exercise to show the following:

Theorem 13.13. An undirected graph G is chordal bipartite if and only if G is bipartite and weakly chordal.

Chordal bipartite graphs are also related to strongly chordal graphs and to totally balanced matrices. A 0/1 matrix is called *totally balanced* if it does not contain as a submatrix the (vertex-edge) incidence matrix of a cycle of length ≥ 3. A 0/1 matrix is called Γ-*free* if it does not contain a submatrix of the form $\left(\begin{smallmatrix} 1 & 1 \\ 1 & 0 \end{smallmatrix}\right)$. Anstee and Farber [1984], Hoffman, Kolen and Sakarovitch [1985] and Lubiw [1987] proved the following:

Theorem 13.14. A 0/1 matrix is *totally balanced* if and only if its rows and columns can be permuted so that the resulting matrix is Γ-*free*.

Let $G = (V, E)$ be an undirected graph with $V = \{v_1, \ldots, v_n\}$. We define the *clique-vertex incidence graph* of G to be the bipartite graph $H(G)$ with the vertices of G on one side and the maximal cliques of G on the other side, such that a vertex v of G is adjacent to a clique K of G in $H(G)$ if and only if v is a member of K in G. In Section 8.2 (page 174), studying the characterizations of interval graphs, we defined a similar notion, the *clique-vertex incidence matrix* of G which we denote by $\mathbf{M} = \mathbf{M}(G)$.

Finally, we define $B(G) = (X, Y, E')$ to be the bipartite graph where $X = \{x_1, \ldots, x_n\}$, $Y = \{y_1, \ldots, y_n\}$ and $E' = \{x_i y_i \mid v_i \in V\} \cup \{x_i y_j \mid v_i v_j \in E\}$. The graph $B(G)$ is, in fact, the same as the *closed neighborhood-vertex incidence graph* of G.

The following theorem is due to Farber [1983].

Theorem 13.15. The following conditions are equivalent:
(i) G is a strongly chordal graph.
(ii) $H(G)$ is a chordal bipartite graph.
(iii) $\mathbf{M}(G)$ is a totally balanced matrix.
(iv) $B(G)$ is a chordal bipartite graph.

We mention another interesting equivalence which follows essentially from our proof of Theorem 12.1, and was stated explicitly later in Brandstädt [1991].

[6] When I introduced the term "chordal bipartite graph" in 1978, my thinking was that the word "chordal" in this context should be interpreted as an adjective permitting cycles of the smallest possible length (length 3 for graphs in general, or length 4 for bipartite graphs) but requiring that any larger cycle must have a chord. Thus, the meaning of "chordal" was context dependent, admittedly a somewhat confusing matter. Indeed, this criticism is voiced in Brandstädt, Le and Spinrad [1999, page 41]. Clearly, *a graph G is both a chordal graph and bipartite if and only if G is a forest of trees.*

Theorem 13.16. An undirected graph G is a chordal graph if and only if $B(G)$ is a perfect elimination bipartite graph.

Our Corollary 12.11 (page 263) related chordal bipartite graphs to perfect elimination bipartite graphs by adding the hereditary property to the latter. We now present two additional characterizations of chordal bipartite graphs.

Let $H = (X, Y, E)$ be a bipartite graph, and let $\sigma = [e_1, \ldots, e_m]$ be an ordering of the edges. Define $H_i = (X, Y, E_i)$ where $E_i = \{e_i, \ldots, e_m\}$, that is, H_i is the graph obtained by erasing the edges $e_1, \ldots, e_{i-1}$, but not their endpoints. We call σ a *perfect "edge-without-vertex" erasing* ordering if e_i is bisimplicial in H_i.

Denote by $\text{Split}_Y(H)$ the split graph obtained by completing Y into a clique.

In the following theorem, the equivalence (i) $\Longleftrightarrow$ (ii) is due to Dalhaus [1991], and several researchers, according to Brandstädt, Le and Spinrad [1999], have observed (i) $\Longleftrightarrow$ (iii).

Theorem 13.17. The following conditions are equivalent for a bipartite graph H:

(i) H is a chordal bipartite graph.
(ii) $\text{Split}_Y(H)$ is a strongly chordal graph.
(iii) H has an edge-without-vertex erasing order.

Further results on chordal bipartite graphs, relating them to vertex orderings, totally balanced matrices, matrices having a Γ-free ordering, and other classes, can be found throughout Brandstädt, Le and Spinrad [1999]. Minimum triangulations of chordal bipartite graphs are studied in Kloks [1994].

A Final Note on Terminology

Samuel Eilenberg, one of the leading mathematicians of the twentieth century, and my doctoral thesis advisor, objected strenuously against the use of mathematical terms such as "partial function". In his view, an adjective modifying a noun should specialize the mathematical concept represented by the noun and *not* generalize it. Since a partial function is not a function at all, but rather a mapping which is defined on only part of its domain, i.e., a partially defined function, he regarded such terms as an imprecise abuse of language. For him, semigroups were always monoids (with identity), and he would have disliked weakly chordal graphs.

Thus, quite correctly, a subset is a set, a recursive function is a function, an alligator purse is a purse, and a strongly chordal graph is a chordal graph, which in turn is a graph. The "is-a" hierarchy is a partial order (whoops!) is a partially ordered set (whoops again!) is simply an order.

The terminology we used in this book, and subsequently, has tried to follow this principle. For example, we prefer to use the terms interval tolerance graphs

and neighborhood subtree tolerance graphs when referring to these classes, but never tolerance interval graphs since the concept of an interval graph is so firmly established in the literature. Similarly, we insist on using the term interval probe graphs. The use of chordal bipartite graph is consistent with this approach, but has caused confusion. It is never possible to get it right every time, but we hope that there will be enough tolerance on both sides of the interval to keep the conflict graph very sparse.

Bibliography

Allen, James F.
[1983] Maintaining knowledge about temporal intervals, *Commun. ACM* **26**, 832–843.

Alon, Noga, and Scheinerman, Edward R.
[1988] Degrees of freedom versus dimension for containment orders, *Order* **5**, 11–16.

Andreae, T.
[1985] On superperfect non-comparability graphs, *J. Graph Theory* **9**, 523–532.

Anstee, Richaed P., and Farber, Martin
[1984] Characterizations of totally balanced matrices, *J. of Algorithms* **5**, 215–230.

Apostolico, A., and Hambrus, S.E.
[1987] Finding maximum cliques on circular-arc graphs, *Infor. Process. Lett.* **26**, 209–215.

Atkins, J.E. and Middendorf, M.
[1996] On physical mapping and the consecutive ones property for sparce matrices, *Discrete Applied Math.* **71**, 23–40.

Avriel, M., Penn, M. and Shpirer, N.
[2000] Container ship stowage problem: complexity and connection to the coloring of circle graphs, *Discrete Applied Math.* **103**, 271–279.

Berge, Claude, and Chvátal, V., eds.
[1984] "*Perfect Graphs*", *Annals of Discrete Math., vol.* **21**, North-Holland, Amsterdam.

Berry, A., and Bordat, J. P.
[2001] Asteroidal triples of moplexes, *Discrete Applied Math.* **111**, 219–229.

Bibelnieks, E., and Dearing, P. M.
[1993] Neighborhood subtree tolerance graphs, *Discrete Applied Math.* **43**, 107–144.

Boros, Endre
[1991] On shift stable hypergraphs, *Discrete Math.* **87**, 81–84.

Bouchet, A.
[1987] Reducing prime graphs and recognizing circle graphs, *Combinatorica* **7**, 243–254.
[1994] Circle graph obstructions, *J. Combin. Theory B* **60**, 8–22.

Brandstädt, A.
[1991] Classes of bipartite graphs related to chordal graphs, *Discrete Applied Math.* **32**, 51–60.
[1997] LexBFS and powers of chordal graphs, *Discrete Math.* **171**, 27–42.

Brandstädt, A., Le, V. B., and Spinrad, J. P.
[1999] "*Graph Classes: A Survey*", SIAM, Philadelphia.

Cheah, M., and Corneil, D. G.
[1996] On the structure of trapeziod graphs, *Discrete Applied Math.* **66**, 109–133.

Chudnovsky, M., Cornuéjols, G., Liu, X., Seymour, P., and Vušković, K.
[2002] Cleaning for Bergeness, submitted for publication.

Chudnovsky, M., Robertson, N., Seymour, P. and Thomas, R.
[2002] The Strong Perfect Graph Theorem, submitted for publication.

Chudnovsky, M., and Seymour, P.
[2002] Recognizing Berge graphs, submitted for publication.

Chvátal, V.
[1985] Star-cutsets and perfect graphs, *J. Combin. Theory B* **39**, 189–199.

Cook, W. J., Cunningham, W. H., Pulleyblank, W. R., and Schrijver, A.
[1998] "*Combinatorial Optimization*", John Wiley & Sons, New York.

Corman, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C.
[2001] "*Introduction to Algorithms*", second edition, McGraw-Hill and MIT Press, Cambridge, Mass.

Corneil, D. G., Kim, H., Natarajan, S., Olariu, S., and Sprague, A.
[1995] Simple linear time recognition of unit interval graphs, *Infor. Process. Letters* **55**, 99–104.

Corneil, D. G., Lerchs, H., and Burlingham, L.
[1981] Complement reducible graphs, *Discrete Applied Math.* **3**, 163–174.

Corneil, D. G., Olariu, S., and Stewart, L.
[1997] Asteroidal triple-free graphs, *SIAM J. Discrete Math.* **10**, 399–430.
[1999] A linear algorithms for dominating pairs in asteroidal triple-free graphs, *SIAM J. Computing* **28**, 1284–1297.

Corneil, D. G., Perl, Y., and Stewart, L.
[1985] A linear recognition algorithm for cographs, *SIAM J. Computing* **14**, 926–934.

Cornuéjols, G., Liu, X., and Vušković, K.
[2002] A polynomial algorithm for recognizing Berge graphs, submitted for publication.

Dagan, Ido, Golumbic, Martin Charles, and Pinter, Ron.Y.
[1988] Trapezoid graphs and their coloring, *Discrete Applied Math.* **21**, 35–46.

Dahlhaus, E.
[1991] Chordale Graphen im besonderen Hinblick auf parallele Algorithmen, Habilitation thesis, Univ. Bonn.

Eschen, Elaine M., and Spinrad, J. P.
[1993] An $O(n^2)$ algorithm for circular-arc graph recognition, *Proc. 4th ACM-SIAM Symp. on Discrete Algorithms,* pp. 128–137.

Farber, M.
[1983] Characterizations of strongly chordal graphs, *Discrete Math.* **43**, 173–189.

Felsner, S., Müller, R., and Wernisch, L.
[1997] Trapezoid graphs and generalizations, geometry and algorithms, *Discrete Applied Math.* **74**, 13–32.

Fishburn, Peter C.
[1985] "*Interval Orders and Interval Graphs: A Study of Partially Ordered Sets*", John Wiley & Sons, New York.
[1988] Interval orders and circle orders, *Order* **5**, 225–234.

Gabor, C., Supowit, K., and Hsu, W.
[1989] Recognizing circle graphs in polynomial time, *J. Assoc. for Comput. Mach.* **36**, 435–473.

Garey, M. R., Johnson, D. S., Miller, G. L., and Papadimitriou, C. H.
[1980] The complexity of coloring circular arcs and chords, *SIAM J. Alg. Discrete Meth.* **1**, 216–227.

Gattass, E. A. and Nemhauser, G. L.
[1981] An application of vertex packing to data analysis in the evaluation of pavement deterioration, *Operations Research Letters* **1**, 13–17.

Goldberg, A. V.
[1998] Recent developments in maximum flow algorithms, Technical Report 98-045, NEC Research Institute.

Goldberg, A. V., and Tarjan, R. E.
[1988] A new approach to the maximum flow problem, *J. Assoc. for Comput. Mach.* **35**, 921–940.

Goldberg, P. W., Golumbic, M. C., Kaplan, H., and Shamir, R.
[1995] Four strikes against physical mapping of DNA, *J. Comput. Biology* **3**, 139–152.

Golumbic, Martin Charles
[1984] Algorithmic aspects of perfect graphs, *Annals of Discrete Math.* **21**, 301–323.
[1988] Algorithmic aspects of intersection graphs and representation hypergraphs, *Graphs and Combinatorics* **4**, 307–321.
[1998] Reasoning about time, in "*Mathematical Aspects of Artificial Intelligence*", F. Hoffman, ed., American Mathematical Society, *Proc. Symposia in Applied Math.*, vol 55, pp. 19–53.

Golumbic, Martin Charles, and Hammer, Peter L.
[1988] Stability in circular arc graphs, *J. of Algorithms* **9**, 314–320.

Golumbic, Martin Charles, and Jamison, Robert E.
[1985a] Edge and vertex intersection of paths in a tree, *Discrete Math.* **55**, 151–159.
[1985b] Edge intersection graphs of paths in a tree, *J. Combin. Theory B* **38**, 8–22.

Golumbic, Martin Charles, and Kaplan, Haim
[1998] Cell flipping in permutation diagrams, *Lecture Notes in Comput. Sci.* **1373**, Springer-Verlag, 577–586.

Golumbic, Martin Charles, Kaplan, Haim, and Shamir, Ron
[1994] On the complexity of DNA physical mapping, *Advances in Applied Math.* **15**, 251–261.
[1995] Graph sandwich problems, *J. of Algorithms* **19**, 449–473.

Golumbic, Martin Charles, and Laskar, Renu
[1993] Irredundancy in circular arc graphs, *Discrete Applied Math.* **44**, 79–89.

Golumbic, Martin Charles, and Mintz, Aviad
[1999] Factoring logic functions using graph partitioning, *Proc. IEEE/ACM Int. Conf. Computer Aided Design*, pp. 195–198.

Golumbic, Martin Charles, Mintz, Aviad, and Rotics, Udi
[2001] Factoring and recognition of read-once functions using cographs and normality, *Proc. 38th ACM Design Automation Conference*, pp. 109–114.

Golumbic, Martin Charles, and Monma, Clyde L.
[1982] A generalization of interval graphs with tolerances, *Proc. 13th Southeastern Conf. on Combinatorics, Graph Theory and Computing, Congressus Numerantium* **35**, Utilitas Math., Winnipeg, Canada, pp. 321–331.

Golumbic, Martin Charles, Monma, Clyde L., and Trotter, W. T.
[1984] Tolerance graphs, *Discrete Applied Math.* **9**, 157–170.

Golumbic, Martin Charles, Rotem, Doron, and Urrutia, Jorge
[1983] Comparability graphs and intersection graphs, *Discrete Math.* **43**, 37–46.

Golumbic, Martin Charles, and Scheinerman, Edward R.
[1989] Containment graphs, posets, and related classes of graphs, *Ann. N.Y. Acad. Sci.* **555**, 192–204.

Golumbic, Martin Charles, and Shamir, Ron
[1993] Complexity and algorithms for reasoning about time, *J. Assoc. for Comput. Mach.* **40**, 1108–1133.

Golumbic, Martin Charles, and Siani, Assaf
[2002] Coloring algorithms for tolerance graphs: reasoning and scheduling with interval constraints, *Lecture Notes in Comput. Sci.* **2385**, Springer-Verlag, pp. 196–207.

Golumbic, Martin Charles, and Trenk, Ann N.
[2004] "*Tolerance Graphs*", Cambridge University Press.

Grötschel, M., Lovász , L., and Schrijver, A.
[1981] The ellipsoid method and its consequences in combinatorial optimization, *Combinatorica* **1**, 169–197.

Hayward, Ryan B.
[1985] Weakly triangulated graphs, *J. Combin. Theory B* **39**, 200–209.

Hayward, R. B., Hoàng, C. T., and Maffray, F.
[1990] Optimizing weakly triangulated graphs, *Graphs and Combinatorics* **6**, 33–35. Erratum to *ibid.* **5**, 339–349.

Hertz, Alain
[1995] Polynomially solvable cases for the maximum stable set problem, *Discrete Applied Math.* **60**, 195–210.

Hoffman, A. J., Kolen, A. W. J., and Sakarovitch, M.
[1985] Totally-balanced and greedy matrices, *SIAM J. Alg. Discrete Meth.* **6**, 721–730.

Hougardy, S.
[1998] Inclusions between classes of perfect graphs, Technical Report, Humbolt University, Berlin. Available at website http://www.informatik.hu-berlin.de/~hougardy/paper/classes.html (last checked April 2003).

Hsu, W. L.
[1985] Maximum weight clique algorithms for circular-arc graphs and circle graphs, *SIAM J. Comput.* **14**, 224–231.
[1995] $O(mn)$ algorithms for the recognition and isomorphism problems on circular-arc graphs, *SIAM J. Comput.* **24**, 411–439.

Hsu, W. L., and Spinrad, J. P.
[1995] Independent sets in circular-arc graphs, *J. of Algorithms* **19**, 154–160.

Johnson, D. S. and McGeoch, C. C., eds.
[1993] "*Network Flows and Matching: First DIMACS Implementation Challenge*", American Mathematical Society, *DIMACS Series on Discrete Mathematics and Theoretical Computer Science*, vol. 12.

Johnson, D. S. and Trick, M. A., eds.
[1996] "*Cliques, Coloring and Satisfiability: Second DIMACS Implementation Challenge*", American Mathematical Society, *DIMACS Series on Discrete Mathematics and Theoretical Computer Science*, vol. 26.

King, V., Rao, S., and Tarjan, R. E.
[1994] A faster deterministic maximum flow algorithm, *J. of Algorithms* **17**, 447–474.

Kloks, T.
[1994] "*Treewidth. Computations and Approximations*", *Lecture Notes in Comput. Sci.* **842**, Springer-Verlag.

Kloks, T., and Bodlaender, H.
[1992] Testing superperfection of k-trees, *Lecture Notes in Comput. Sci.* **621**, Springer-Verlag, pp. 282–293.

Köhler, E.
[2000] Recognizing graphs without asteroidal triples, *Lecture Notes in Comput. Sci.* **1928**, Springer-Verlag, pp. 255–266.

Langley, L.
[1995] Recognition of orders of interval dimension 2, *Discrete Applied Math.* **60**, 257–266.
Lengauer, T.
[1990] "*Combinatorial Algorithms for Integrated Circuit Layout*", John Wiley, New York.
Lovász, L.
[1994] Stable sets and polynomials, *Discrete Math.* **124**, 137–153.
Lubiw, A.
[1987] Doubly lexical orderings of matrices, *SIAM J. Comput.* **16**, 854–879.
Ma, T. H., and Spinrad, J. P.
[1994] On the 2-chain subgraph cover and related problems, *J. of Algorithms* **17**, 251–268.
Mahadev, N. V. R., and Peled, U. N.
[1995] "*Threshold Graphs and Related Topics*", North-Holland, Amsterdam.
McConnell, Ross M.
[2001] Linear time recognition of circular-arc graphs, *Proc. 42nd IEEE Symp. on Foundations of Computer Science,* pp. 386–394.
[2003] Linear time recognition of circular-arc graphs, *Algorithmica,* to appear.
McConnell, R. M., and Spinrad, J. P.
[1999] Modular decomposition and transitive orientation, *Discrete Math.* **201**, 189–241.
McKee, Terry A., and McMorris, Fred R.
[1999] "*Topics in Intersection Graph Theory*", SIAM, Philadelphia.
Monma, C. L., and Wei, V. K.
[1986] Intersection graphs of paths in a tree, *J. Combin. Theory B* **41**, 141–181.
Narasimhan, G., and Manber, R.
[1992] Stability number and chromatic number of tolerance graphs, *Discrete Applied Math.* **36**, 47–56.
Natanson, A., Shamir, R., and Sharan, R.
[2001] Complexity classification of some edge modification problems, *Discrete Applied Math.* **113**, 109–128.
Pevzner, P.
[2000] "*Computational Molecular Biology*", MIT Press, Cambridge, Mass.
Phillips, S., and Westbrook, J.
[1993] On-line load balancing and network flows, *Proc. 25th ACM Symp. on Theory of Computing,* pp. 402–411.
Prisner, E.
[1999] A journey through intersection graph county. Available at website http://www.math.uni-hamburg.de/spag/gd/mitarbeiter/prisner/Pris/Rahmen.html (last checked April 2003).
Ramirez-Alfonsin, J. L., and Reed, B., eds.
[2001] "*Perfect Graphs*", Wiley, New York.
Reiterman, J., Rödl, V., Šiňajová, E., and Tůma, M.
[1985] Threshold hypergraphs, *Discrete Math.* **54**, 193–200.
Rival, I., ed.
[1985] "*Graphs and Order: The Role of Graphs in the Theory of Ordered Sets and Its Applications*", Proc. NATO Advanced Institute on Graphs and Orders (Banff, Canada, May 18–31, 1984), D. Reidel Publishing, Dordrecht, Holland.
Scheinerman, E. R.
[1992] The many faces of circle orders, *Order* **9**, 343–348.
Scheinerman, E.R., and Wierman, J.
[1988] On circle containment orders, *Order* **4**, 315–318.

Shrwani, N.
[1995] "*Algorithms for VLSI Physical Design Automation*", Kluwer Academic Publishers, Boston.
Spinrad, J. P.
[1985] On comparability and permutation graphs, *SIAM J. Comput.* **14**, 658–670.
[1994] Recognition of circle graphs, *J. of Algorithms* **16**, 264–282.
Spinrad, J. P., Brandstädt, A., and Stewart, L.
[1987] Bipartite permutation graphs, *Discrete Applied Math.* **18**, 279–292.
Spinrad, J. P., and Sritharan, R.
[1995] Algorithms for weakly chordal graphs, *Discrete Applied Math.* **59**, 181–191.
Sysło, M. M.
[1985] Triangulated edge intersection of paths in a tree, *Discrete Math.* **55**, 217–220.
Tarjan, R. E., and Yannakakis, M.
[1984] Simple linear-time algorithms to test chordality of graphs, test acyclicity of hypergraphs and selectively reduce acyclic hypergraphs, *SIAM J. Comput.* **13**, 566–579.
Trotter, William T.
[1992] "*Combinatorics and Partially Ordered Sets*", Johns Hopkins, Baltimore.
Verbin, E.
[2002] personal communication.
Waterman, Michael S.
[1995] "*Introduction to Computational Biology*", Chapman Hall, London.
Yannakakis, M.
[1981] Computing the minimum fill-in is NP-complete, *SIAM J. Alg. Discrete Methods* **2**, 77–79.
[1982] The complexity of the partial order dimension problem, *SIAM J. Alg. Discrete Methods* **3**, 351–358.

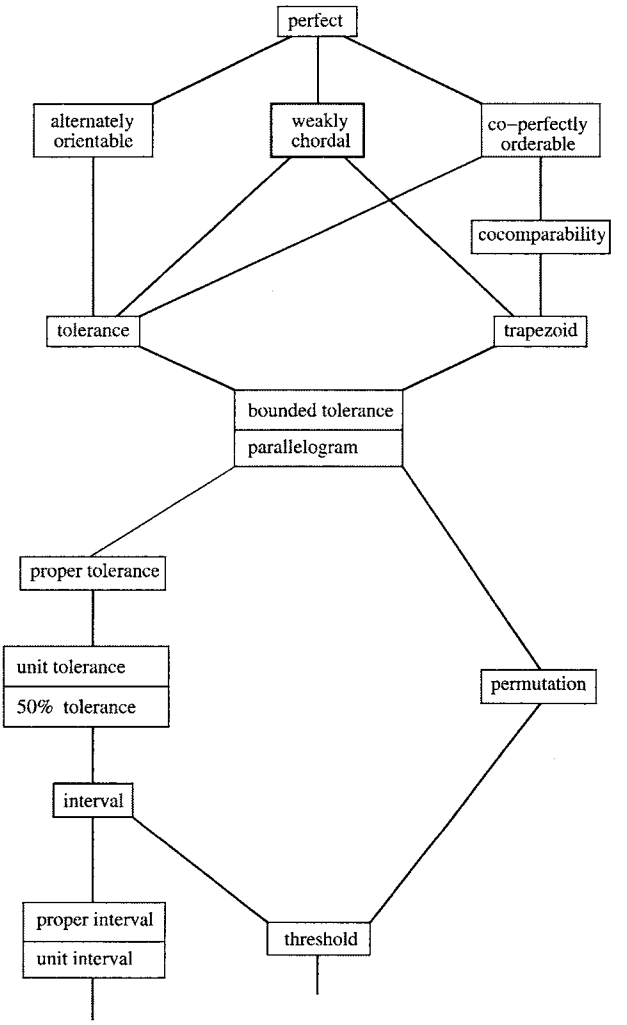

Fig.13.3. A complete hierarchy of classes of perfect graphs ordered by inclusion. Reprinted from Golumbic and Trenk [2004].