Chapter 2

Chordal Graphs

A graph is a *chordal graph* if it has no induced cycles larger than triangles. A *chord of a cycle* is an edge between nonconsecutive vertices of the cycle; thus a graph is chordal if and only if every cycle large enough to have a chord does have a chord. The study of chordal graphs goes back to [Hajnal & Surányi, 1958], frequently under the names *rigid-circuit graphs* or *triangulated graphs*. Chapter 4 of [Golumbic, 1980] is the standard reference for chordal graphs. [Blair & Peyton, 1993] is more up to date and more in the style presented here.

In spite of there having been considerable activity during the 1960s, it was not until the 1970s that chordal graphs were characterized in terms of intersection graphs. Many of the most sophisticated applications of chordal graphs, which we sketch in section 2.4, came later and involved the rediscovery of chordal graph theory in statistics and matrix analysis. The recent dates on many of our references show that chordal graphs are still being intensively studied today.

Contrary to history, we begin with the intersection graph approach to chordal graphs.

2.1 Chordal Graphs as Intersection Graphs

For the purpose of this section only, we define a graph to be a *subtree graph* if it is the intersection graph of a family of subtrees of a tree. But you should keep in mind that Theorem 2.4 at the end of this section will show that *the subtree graphs are precisely the chordal graphs*! The tree and family of subtrees in the definition are called a *tree representation* of the subtree graph and, while a tree is a topological object, it is clear that it can always

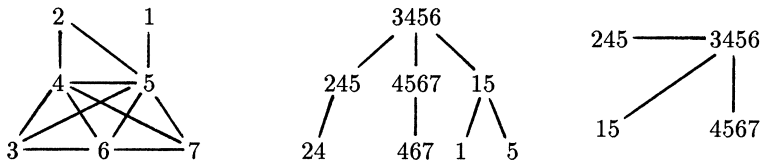

Figure 2.1: *A chordal graph and two tree representations.*

be taken to be a tree in the graph-theoretic sense.

Example 2.1 The graph G shown on the left in Figure 2.1 is a subtree graph isomorphic to $\Omega(\{T_1, \ldots, T_7\})$ where each T_i is the subtree of the tree in the middle induced by those vertices that contain i. For instance, $V(T_5) = \{15,\ 245,\ 3456,\ 4567,\ 5\}$. There are, of course, many such tree representations of G. For instance, the tree shown on the right is a tree representation for G, but now the vertex set is precisely the set of maxcliques of G.

It is easy to see that G is a subtree graph if and only if it has an edge clique cover $\mathcal{E}$ whose members can be associated with vertices of a tree T such that, for every $v \in V(G)$, $\{Q : v \in Q \in \mathcal{E}\}$ induces a subtree T_v of T. This is a very transparent translation of being a subtree graph into a condition on an edge clique cover. Theorem 2.1 shows that the edge clique cover can always be taken to be the set of the maxcliques of G. Theorem 2.3 then shows how to test whether the maxcliques of G can be arranged into a tree as just described.

When a tree representation exists whose vertex set is the set of maxcliques of G, then it is called a *clique tree representation* (or a *clique tree* for) G. Equivalently, a clique tree is a spanning tree of the clique graph $K(G)$ such that, for each $v \in V(G)$, T_v is connected. (Lemma 2.2 will give an alternative condition to check.) Given any clique tree T for G and any two maxcliques Q_i and Q_j of G, let $T(Q_i, Q_j)$ denote the path in T connecting Q_i and Q_j.

Exercise 2.1 Show that in any clique tree T for a chordal graph G, the family $\{T_v : v \in V(G)\}$ of subtrees of T, with each subtree viewed as a set of vertices of T, satisfies the Helly condition.

Theorem 2.1 *A graph is a subtree graph if and only if it has a clique tree representation.* □

Exercise 2.2 Use Lemma 1.11 to prove Theorem 2.1.

Exercise 2.3 Show that every subtree graph is the intersection graph of a family of *distinct* subtrees of a tree. Is every subtree graph the intersection graph of a family of distinct subtrees of a *clique* tree?

The following lemma essentially appears in [Acharya & Las Vergnas, 1982] (see also [Levin, 1983]) modulo knowing other results that we prove in this section and the next; the lemma seems to first appear in this simple "clique tree check" form in [McKee, 1993].

Lemma 2.2 *A spanning subtree T of $K(G)$ is a clique tree for a connected graph G if and only if*

$$|V(G)| = \sum_{Q \in V(T)} |Q| - \sum_{Q_i Q_j \in E(T)} |Q_i \cap Q_j|. \tag{2.1}$$

Proof. Suppose T is a spanning tree of $K(G)$. For each $v \in V(G)$, the subgraph T_v satisfies $1 \leq |V(T_v)| - |E(T_v)|$, with equality if and only if T_v is connected and thus is a subtree. Summing over all $v \in V(G)$ proves equality (2.1). □

Example 2.2 The cycle C_4 is not a subtree graph: each of the four spanning trees of $K(C_4)$ leaves one T_v disconnected, and $4 < 8 - 3$ in equality (2.1).

Exercise 2.4 Show that a subtree graph of order n can have at most n maxcliques.

Theorem 2.3 will show how easy it is to find clique tree representations of subtree graphs. It first appeared in [Bernstein & Goodman, 1981] in the computer science context we discuss in section 2.4, and it has been rediscovered many times. [Gavril, 1987] and [Shibata, 1988] give nice treatments.

It is important to realize that the approach in Theorem 2.3 requires knowing all the maxcliques of G, a computationally hard problem in general—the number of maxcliques of G can grow exponentially in the number of vertices of G—yet one that can be done efficiently for subtree graphs because of Exercise 2.4. In certain applications, for instance the one in subsection 2.4.4 below, G is given at the start as the set of its maxcliques.

Exercise 2.5 Show that the order-$2p$ complete p-partite graph $K_{2,\ldots,2}$ has 2^p maxcliques.

For any graph G, define the *weighted clique graph* $K^w(G)$ to be the clique graph $K(G)$ with each edge Q_iQ_j given weight $|Q_i \cap Q_j|$. Theorem 2.3 will involve maximum spanning trees of $K^w(G)$, which can be found efficiently by using Kruskal's well-known greedy algorithm. Recall that the usual *minimum* spanning tree version of Kruskal's algorithm finds a(ll) minimum spanning tree(s) of a connected weighted graph by repeatedly choosing an edge of smallest weight that does not form a cycle with previously chosen edges. The *maximum* spanning tree version that we use is the same, except that we now always choose an edge with *largest* weight that does not form a cycle with previously chosen edges.

Example 2.3 For the graph G on the left in Figure 2.1, a maximum spanning tree of $K^w(G)$ must contain the weight-three edge joining vertex 3456 to 4567, one of the two weight-two edges incident with 245, and one of the three weight-one edges incident with 15; one maximum spanning tree is shown on the right in the figure. Checking that such a tree is a clique tree requires either checking that each of the seven T_i's is connected or checking that $7 = 21 - 14$ in equality (2.1).

Theorem 2.3 *A connected graph G is a subtree graph if and only if some maximum spanning tree of $K^w(G)$ is a clique tree for G. Moreover, this is equivalent to* every *maximum spanning tree of $K^w(G)$ being a clique tree for G, and every clique tree of G is such a maximum spanning tree.*

Proof. If some maximum spanning tree of $K^w(G)$ is a clique tree for G, then by definition G is a connected subtree graph.

Conversely, suppose G is a connected subtree graph with clique tree T. Thus T is a spanning tree of $K^w(G)$, but suppose, arguing toward a contradiction, that T is not a *maximum* spanning tree of $K^w(G)$. Among all maximum spanning trees of $K^w(G)$, choose T' to have a maximum number of edges in common with T. Pick any edge $e = Q_iQ_j \in E(T') \setminus E(T)$ having weight $|Q_i \cap Q_j|$ as large as possible. Since T is a tree representation of G, each $v \in V(G)$ that is in $Q_i \cap Q_j$ must also be in every vertex of the path $T(Q_i, Q_j)$ in T_v, and so each edge of this path must have weight at least $|Q_i \cap Q_j|$. There must be some edge f of this path that is not in $E(T')$. But the spanning tree $T'' = T' - e + f$ then has total weight at least as large as the weight of T'. Thus T'' is a maximum spanning tree of $K^w(G)$ having

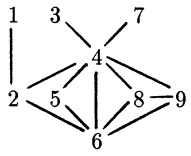

Figure 2.2: *A graph having three minimal vertex separators.*

one more edge in common with T than does T', contradicting the choice of T'.

Therefore, being a clique tree implies some maximum spanning tree of $K^w(G)$ is a clique tree. The rest of the theorem follows from Lemma 2.2 since every maximum spanning tree T of $K^w(G)$ will have the same total weight $\sum_{QQ' \in E(T)} |Q \cap Q'|$. □

A set S of vertices of G is a *minimal vertex separator* of G whenever there exist $u, v \in V(G)$ such that every path connecting u and v contains a vertex in S and no proper subset of S has this same property.

Example 2.4 In the graph shown in Figure 2.2, the minimal vertex separators are $\{2\}$, $\{4\}$, and $\{4, 6\}$.

The following two exercises show how Kruskal's algorithm locates the minimal vertex separators of a subtree graph and that, even though a subtree graph can have many clique trees T, the multiset $\{Q_i \cap Q_j : Q_iQ_j \in E(T)\}$ is uniquely determined.

Exercise 2.6 (see [Barrett, Johnson, & Lindquist, 1989] and [Ho & Lee, 1989]) Suppose G is a connected subtree graph with clique tree T and $S \subseteq V(G)$. Show that S is a minimal vertex separator of G if and only if there exists $Q_iQ_j \in E(T)$ such that $S = Q_i \cap Q_j$.

Exercise 2.7 For a subtree graph G with clique tree T, show that the multiplicity of each $Q_i \cap Q_j$ in the multiset $\{Q_i \cap Q_j : Q_iQ_j \in E(T)\}$ equals one fewer than the number of components in the subgraph of G induced by those vertices that are adjacent to every vertex in $Q_i \cap Q_j$.

Exercise 2.8 Construct several clique trees for the chordal graph in Figure 2.2 and then use them to illustrate Exercise 2.7.

Exercise 2.9 For any graph G, define $S \subseteq V(G)$ to be a *minimal vertex weak separator* of G if there exist two vertices in a common component of the subgraph of G induced by $V(G) \setminus S$ such that the distance between the two vertices is greater in that subgraph than in G. Call an edge Q_iQ_j of $K^\omega(G)$ a "dominated chord" of a clique tree T of G if $Q_iQ_j \notin E(T)$ and $|Q_i \cap Q_j| < |Q \cap Q'|$ for every $QQ' \in E(T(Q_i, Q_j))$.

Show that S is a minimal vertex weak separator of G if and only if there exists a dominated chord Q_iQ_j of T such that $S = Q_i \cap Q_j$.

The next theorem is from [Buneman, 1974], [Gavril, 1974a], and [Walter, 1978]. Our argument follows [Shibata, 1988].

Theorem 2.4 (Buneman, Gavril, and Walter) *A graph is a subtree graph if and only if it is a chordal graph.*

Proof. First, suppose G is a subtree graph with clique tree T. Arguing toward a contradiction, suppose that G contains an *induced* cycle C whose vertices are, in order, $v_1, \ldots, v_k, v_1$ where $k \geq 4$. Putting $v_0 = v_k$ and $v_{k+1} = v_1$, we know that, for $i \in \{1, \ldots, k\}$, $T_{v_i} \cap T_{v_{i-1}} \neq \emptyset \neq T_{v_i} \cap T_{v_{i+1}}$, but, since C is induced, $T_{v_i} \cap T_{v_j} = \emptyset$ for all other vertices v_j of C. Thus there exists a path Π through T connecting some vertex of T_{v_1} with some vertex of T_{v_k} and containing along the way vertices from each T_{v_j} with $1 < j < k$. But v_1 is also adjacent to v_k, so $T_{v_1} \cap T_{v_k} \neq \emptyset$ with $T_{v_1} \cap T_{v_k} \cap Q = \emptyset$ for every vertex Q of Π in T_{v_i} where $1 < i < k$. This contradicts T being a tree.

Conversely, suppose G contains no induced cycle larger than a triangle and that T is any maximum spanning tree of $K^w(G)$. Arguing toward a contradiction, suppose that there are nonadjacent vertices Q and Q' of T such that (i) there is some vertex in $T(Q, Q')$ that does not contain $Q \cap Q'$ and, among all such, that (ii) $|Q \cap Q'| = k$ is as large as possible. Moreover, among all such Q, Q', suppose that (iii) $T(Q, Q')$ is as short a path as possible. Say $T(Q, Q')$ is $Q = Q_1, Q_2, \ldots, Q_{p-1}, Q_p = Q'$, where $p \geq 3$.

For each $i \in \{1, \ldots, p-1\}$, define $R_i = (Q_i \cap Q_{i+1}) \setminus (Q \cap Q') \subseteq V(G)$. Let each $|Q_i \cap Q_{i+1}| = k_i$ $(1 \leq i < p)$. Since T is a maximum spanning tree for $K^w(G)$ and $QQ' \notin E(T)$, each $k_i \geq k$. By (iii), $Q \cap Q' \not\subseteq Q_i$ for each $i \in \{2, \ldots, p-1\}$. Therefore, each $R_i \neq \emptyset$. Since $R_i \cap R_{i+1} \subseteq Q_{i+1}$, the subgraph of G induced by $\cup R_i$ is connected and we can pick a shortest path $x_1, x_2, \ldots, x_q$ therein such that $x_1 \in R_1$ and $x_q \in R_{p-1}$. For each $v \in Q \cap Q'$, v will be adjacent to x_1 and x_q and so $v, x_1, \ldots, x_q, v$ will be a cycle in G. Since the path $x_1, x_2, \ldots, x_q$ was chosen to be shortest and since G has no induced cycles larger than triangles, each x_i must be adjacent to v. Since v

was chosen arbitrarily from $Q \cap Q'$, it must be that, for each $i \in \{1, \ldots, q-1\}$, there is a maxclique S_i of G containing $\{x_i, x_{i+1}\} \cup (Q \cap Q')$. Set $S_0 = Q$ and $S_q = Q'$, and note that each $S_i \cap S_{i+1} \supseteq (Q \cap Q') \cup \{x_{i+1}\}$, so $|S_i \cap S_{i+1}| > k$. So by (ii), $S_i \cap S_{i+1}$ is contained in each vertex along $T(S_i, S_{i+1})$ for each $i \in \{0, \ldots, q-1\}$. Thus $Q \cap Q' \subseteq S_i \cap S_{i+1}$ is contained in each vertex along $T(Q, Q')$, contradicting (i). □

See [Hsu & Ma, 1991] for a linear-time algorithm for finding a clique tree of a chordal graph. Other authors pay attention to what sorts of clique trees a chordal graph can have. For instance, [Blair & Peyton, 1994] gives a linear-time algorithm for finding minimum diameter clique trees of a chordal graph, while [Lih, 1993] investigates finding clique trees that have paths to which all vertices are close. [Lin, McKee, & West, to appear] investigates clique trees having a minimum number of leaves, and [Prisner, 1992] studies chordal graphs that have clique trees with only three leaves. Chapter 3 is devoted to chordal graphs that have clique trees with only two leaves.

[Chen & Lih, 1990] and [Bandelt & Prisner, 1991] characterize chordal graphs whose clique graph is not chordal and show that if G is chordal then $K(K(G))$ is chordal. Section 7.5 is devoted to the clique graphs of chordal graphs.

Exercise 2.10 (Chen & Lih and Bandelt & Prisner) Give an example of a chordal graph of order eight whose clique graph is not chordal.

[Raychaudhuri, 1988] gives a polynomial algorithm for finding the intersection number of a chordal graph.

2.2 Other Characterizations

One measure of the richness of chordal graph theory is the large number of different characterizations of chordal graphs in the literature; see Theorem 7.47, [Benzaken, Crama, Duchet, Hammer, & Maffray, 1990], and [Bakonyi & Johnson, 1996] for just a few examples. This section considers several standard characterizations, but because of our focus on clique trees and intersection graphs our proofs are not necessarily the standard ones.

Exercise 2.11 (see [Dirac, 1961]) Show that a graph is chordal if and only if every minimal vertex separator is complete.

We need two standard definitions for Theorem 2.5, from [Fulkerson & Gross, 1965] and [Rose, 1970]. A vertex is a *simplicial* vertex of a graph if

its neighbors induce a complete graph (which, remember, includes the case of the null graph). Equivalently, a vertex is simplicial if it is in a unique maxclique. An ordering $\langle v_1, \ldots, v_n \rangle$ of all the vertices of G is a *perfect elimination ordering* of G if, for each $i \in \{1, \ldots, n\}$, v_i is a simplicial vertex of the subgraph induced by $\{v_i, \ldots, v_n\}$.

Example 2.5 In the graph on the left in Figure 2.1, vertices 1, 2, 3, and 7 are the simplicial vertices. The vertices have been labeled so that their numerical ordering is one possible perfect elimination ordering.

Theorem 2.5 (Fulkerson & Gross and Rose) *A graph is chordal if and only if it has a perfect elimination ordering.*

Proof. First, suppose G is a subtree graph with clique tree T. We argue by induction on the order of T with the result trivial when the order is one. Suppose Q is any maxclique of G corresponding to a leaf of T. Since no maxclique can be contained in any other, Q must contain some $v \in V(G)$ that occurs in only that one maxclique, and so v is simplicial. Let G^- result from G by removing v, and let T^- result from T by removing v from each vertex of T. Then G^- is still a chordal graph, since it has tree representation T^-. By inductive hypothesis, Q^- has a perfect elimination ordering that, when v is inserted at the beginning, makes a perfect elimination ordering for G.

Conversely, suppose $\langle v_1, \ldots, v_n \rangle$ is a perfect elimination ordering for G. We argue by induction on n with the result trivial when $n = 1$. Suppose Q is the maxclique of G consisting of v_1 and all its neighbors. Let G^- be the subgraph of G induced by $\{v_2, \ldots, v_n\}$. Since $\langle v_2, \ldots, v_n \rangle$ is a perfect elimination ordering for G^-, the inductive hypothesis implies that there is a clique tree T^- for G^-. Notice that $Q^- = Q \backslash \{v_1\}$ will be contained in some vertex R of T^-. If $Q^- = R$, then let T result by simply inserting v_1 into R; if Q^- is properly contained in R, then let T result by creating a new vertex Q and making it adjacent to R. In either case, T is a tree representation for G. □

Exercise 2.12 Show that a graph is chordal if and only if every induced subgraph has a simplicial vertex.

Exercise 2.13 Show that finding perfect elimination orderings is "foolproof" in the sense that, if G has a perfect elimination ordering, then taking *any* simplicial vertex v of G as a first vertex, then *any* simplicial vertex of the subgraph induced by $V(G) \backslash \{v\}$ as the second, and so on, will always result in a perfect elimination ordering of G.

Exercise 2.14 Show how a perfect elimination ordering for G can be used to give a direct construction of a clique tree for G.

We conclude this section with a characterization from [Tarjan & Yannakakis, 1984] that can be implemented in $\mathcal{O}(|V(G)|+|E(G)|)$ time; see also [Golumbic, 1984] and [Shier, 1984]. A *maximum cardinality search* "marks" the vertices of G as follows: First mark an arbitrary vertex; then repeatedly mark any previously unmarked vertex having as many marked neighbors as possible. Stop when all vertices have been marked.

Example 2.5 (continued) In the graph in Figure 2.1, taking the vertices in the *opposite* of their numerical order shows one possible order in which they might be marked by a maximum cardinality search. If vertices 5, 6, and 7 (in any order) are the first three marked, then the remaining vertices must be marked in the order $4, 3, 2, 1$.

Theorem 2.6 (Tarjan & Yannakakis) *A graph G is chordal if and only if in some maximum cardinality search of G, as each vertex becomes marked, its previously marked neighbors are pairwise adjacent in G. Moreover, this is equivalent to, in* every *maximum cardinality search of G, as each vertex becomes marked, its previously marked neighbors are pairwise adjacent in G.*

Proof. If some maximum cardinality search marks the vertices of G in the order $v_1, \ldots, v_n$ such that the neighbors of v_i among $v_1, \ldots, v_{i-1}$ are pairwise adjacent in G, then $\langle v_n, \ldots, v_1 \rangle$ is automatically a perfect elimination ordering the G, and so G is chordal by Theorem 2.5.

Conversely, suppose G is connected and chordal with clique tree T. Suppose a maximum cardinality search marks the vertices of G in the order $v_1, \ldots, v_n$. (We show how maximum cardinality search locates maxcliques of G.) No matter which v_1 was chosen, vertices $v_1, \ldots, v_k$ (for some $k \leq n$) will form a maxclique Q of G, because of always marking a vertex that is adjacent to as many previously marked vertices as possible, and so $\{v_1, \ldots, v_k\} = Q \in V(T)$; for the purpose of this proof, call such a vertex Q a "saturated vertex" of T. Since T is a maximum spanning tree of $K^w(G)$ by Theorem 2.3, the next vertex v marked in G will occur in some neighbor Q' of Q in T for which $Q \cap Q'$ (the previously marked vertices that v is adjacent to) is as large as possible. Any unmarked vertices occurring in Q' will now be adjacent to more than $|Q \cap Q'|$ previously marked vertices, and so these will be marked next, making Q' saturated. This process continues

to make saturated vertices of T one at a time, with the vertices saturated at any time always forming a subtree of T. Since each newly marked vertex of G is always in the same maxclique as its previously marked neighbors, these neighbors will be pairwise adjacent. □

Exercise 2.15 Suppose G is chordal. The first paragraph of the proof of Theorem 2.6 shows that every maximum cardinality search of G corresponds to a reversed perfect elimination ordering of G. Show by example that the converse fails—that a perfect elimination ordering of a chordal graph need not correspond to a reversed maximum cardinality search marking.

Exercise 2.16 (Blair, England, & Thomason) Prim's algorithm constructs a(ll) maximum spanning tree(s) of a weighted graph by starting at an arbitrary vertex and repeatedly choosing an edge of largest weight that joins a vertex already in the tree with a vertex not yet in the tree. ([Tarjan, 1983] and [Graham & Hell, 1985] contain detailed analysis of both the Kruskal and Prim algorithms.) Discuss how the second paragraph of the proof of Theorem 2.6 illustrates the central theme of [Blair & Peyton, 1993]: that "the maximum cardinality search algorithm is just Prim's algorithm in disguise."

See [Panda, 1996] for deeper discussion of maximum cardinality-type algorithms, and [Simon, 1995] for the role of minimal vertex separators in maximum cardinality-type search algorithms on chordal graphs. [Galinier, Habib, & Paul, 1995] contains more information on clique trees and their role in algorithms. [Kumar & Veni Madhavan, 1989] presents a simple linear-time algorithm for testing the planarity of a chordal graph based on a chordal graph being planar if and only if it is K_5-free and each 3-vertex minimal vertex separator has multiplicity one.

2.3 Tree Hypergraphs

Continuing the discussion of section 1.6, a hypergraph $(X, \mathcal{E})$ is a *tree hypergraph* if there is a tree T with $X = V(T)$ such that, for each $S_i \in \mathcal{E}$, there is a subtree T_i of T with $V(T_i) = S_i$.

Exercise 2.17 Show that the hypergraph $(\{a, b, c, d\}, \mathcal{E})$ with $\mathcal{E} = \{\{a\}, \{c\}, \{b, d\}, \{a, b, d\}, \{a, b, c, d\}\}$ is a tree hypergraph, and that the tree T in the definition can be any tree with vertex set $\{a, b, c, d\}$ so long as it contains the edge bd and one of the edges ab, ad.

Clearly, the line graph $\Omega(\mathcal{E})$ of a tree hypergraph $(X, \mathcal{E})$ is a subtree graph, and so is a chordal graph by Theorem 2.4. The next example, however, shows that being a tree hypergraph requires more than just having a chordal line graph.

Example 2.6 The hypergraph $(\{1,2,3\}, \mathcal{E})$ having $\mathcal{E} = \{\{1,2\}, \{1,3\}, \{2,3\}\}$ has a chordal line graph ($\cong K_3$), yet is not a tree hypergraph; the following exercise shows that (at least part of) the problem is that $\mathcal{E}$ does not satisfy the Helly condition.

Exercise 2.18 Show that every tree hypergraph is a Helly hypergraph.

The following theorem appeared independently in [Duchet, 1978], [Flament, 1978], and [Slater, 1978]; our argument follows Slater's.

Theorem 2.7 (Duchet, Flament, and Slater) *A hypergraph is a tree hypergraph if and only if it is a Helly hypergraph with a chordal line graph.*

Proof. We have already observed the implication one way. For the converse, suppose $(X, \mathcal{E})$ is a Helly hypergraph and its line graph $G = \Omega(\mathcal{E})$ is chordal. Say $\mathcal{E} = \{S_1, \ldots, S_m\}$. We argue by induction on m. For the $m = 1$ basis, $(X, \mathcal{E})$ is a tree hypergraph for which T can be *any* tree with vertex set S_1. Suppose $m > 1$. Since G is chordal, Theorem 2.5 allows us to reorder the S_i's as necessary so that S_1 is a simplicial vertex of G and $\{S_1, \ldots, S_k\}$ induces the unique maxclique of G that contains S_1. We can assume $k \geq 2$ since if $k = 1$, meaning that S_1 is an isolated vertex in G, then the remainder of the argument becomes trivial. By the Helly condition, there is some $x \in S_1 \cap \cdots \cap S_k$. Put $S_1' = S_1 \setminus \{x\}$ and $S_i' = S_i \setminus S_1'$ when $i \geq 2$. Note that $k < j \leq m$ implies $S_1 \cap S_j = \emptyset$ and $S_j' = S_j$. Suppose i and j are such that $2 \leq i < j \leq m$. If $S_i' \cap S_j' \neq \emptyset$, then $S_i \cap S_j \supseteq S_i' \cap S_j' \neq \emptyset$. If $S_i \cap S_j \neq \emptyset$, then either $j \leq k$ and $x \in S_i' \cap S_j' \neq \emptyset$, or $j > k$ and $S_i' \cap S_j' = (S_i \setminus S_1') \cap S_j = S_i \cap S_j \neq \emptyset$ since $S_1 \cap S_j = \emptyset$. Thus $S_i \cap S_j \neq \emptyset$ if and only if $S_i' \cap S_j' \neq \emptyset$. In this way, $\{S_2', \ldots, S_m'\}$ satisfies the Helly condition and $\Omega(\{S_2', \ldots, S_m'\}) \cong \Omega(\{S_2, \ldots, S_m\})$ is chordal. So by the induction hypothesis, $(X \setminus S_1', \{S_2', \ldots, S_m'\})$ is a tree hypergraph with respect to some tree T'. Form T from T' by adding, for each element of S_1', a new vertex of degree one adjacent to x. Then each S_i is the vertex set of a subtree of T and $V(T) = X$. □

Exercise 2.19 Show that a graph is chordal if and only if it is the line graph of a tree hypergraph.

Let $H = (X, \mathcal{E})$ be a hypergraph. A *partial hypergraph* of H is a hypergraph $H' = (X', \mathcal{E}')$ where $\mathcal{E}' \subseteq \mathcal{E}$ and $X' = \cup_{S' \in \mathcal{E}'} S'$. If $A \subseteq X$, the *subhypergraph of X induced by A* is the hypergraph $H_A = (A, \mathcal{E}_A)$ where $\mathcal{E}_A = \{S \cap A : S \in \mathcal{E}\}$.

While sections 2.1 and 2.2 show that every induced subgraph of a subtree graph is itself a subtree graph, the following example shows that this hereditary property fails for tree hypergraphs.

Example 2.7 Consider the tree hypergraph $(\{1,2,3,4\}, \mathcal{E})$ in which $\mathcal{E} = \{\{1,2,3\}, \{1,2,4\}, \{2,3,4\}\}$. If $A = \{1,3,4\}$, then Theorem 2.7 shows that H_A is not a tree hypergraph.

The *dual hypergraph* $H^* = (X^*, \mathcal{E}^*)$ of a hypergraph $H = (X, \mathcal{E})$ has $X^* = \mathcal{E}$ with $\mathcal{E}^* = \{S_x^* : x \in X\}$ where each $S_x^* = \{S \in \mathcal{E} : x \in S\}$. Note that $H^{**} \cong H$. Given a graph G, the *clique hypergraph* of G is the hypergraph $(V(G), \mathcal{E})$ where $\mathcal{E}$ is the set of all maxcliques of G.

Exercise 2.20 Show that a graph G is chordal if and only if H^* is a tree hypergraph where H is the clique hypergraph of G.

Exercise 2.21 Show that the dual of a subhypergraph of the hypergraph H is isomorphic to a partial hypergraph of H^*.

Section 2.4.2 will sketch an application of tree hypergraphs in database theory. See [Naiman & Wynn, 1992] for an application in probability theory of duals of tree hypergraphs (called "generalized simple tubes" there).

A *cycle of length k* in the hypergraph $H = (X, \mathcal{E})$ is a sequence $v_1, S_1, v_2, S_2, \ldots, S_k, v_1$ where $S_1, \ldots, S_k$ are distinct edges, $v_1, \ldots, v_k$ are distinct vertices, $v_i, v_{i+1} \in S_i$ for all $i = 1, \ldots, k-1$, and $v_k, v_1 \in S_k$. A *totally balanced hypergraph* is a hypergraph in which every cycle of length greater than two contains an edge S_i that contains at least three of the vertices $v_1, \ldots, v_k$ of the cycle.

Exercise 2.22 Suppose H is any totally balanced hypergraph. Show that H^* and all the partial hypergraphs and subhypergraphs of H are also totally balanced and that H must be a Helly hypergraph.

The following theorem can be found in [Lehel, 1983, 1985] and [Ryser, 1969].

Theorem 2.8 *A hypergraph is totally balanced if and only if each of its subhypergraphs is a tree hypergraph.*

Proof. Suppose H is a totally balanced hypergraph. Exercise 2.22 shows that every subhypergraph of H is a totally balanced Helly hypergraph that, by the definition of totally balanced, has a chordal line graph. Theorem 2.7 then implies that every subhypergraph of H is a tree hypergraph.

Conversely, suppose every subhypergraph of H is a tree hypergraph, yet suppose H has a cycle of length three with none of its edges containing three vertices of the cycle. If this cycle has length three, then those three vertices would induce a subhypergraph of H that is not a Helly hypergraph; if it has length greater than three, then its vertices would induce a subhypergraph of H whose line graph is not chordal. Either case contradicts Theorem 2.7. □

A hypergraph is a *strong Helly hypergraph* if each of its subhypergraphs is a Helly hypergraph. Compare the following with the Gilmore criterion in Exercise 1.23.

Theorem 2.9 (Lehel) *A hypergraph $H = (X, \mathcal{E})$ is a strong Helly hypergraph if and only if, for all $u, v, w \in X$, there exists $x \in \{u, v, w\}$ such that every edge in $\mathcal{E}$ that contains at least two of u, v, w also contains x.*

Proof. This follows from applying Exercise 1.23 to all the subhypergraphs induced by distinct $u, v, w \in X$. □

Corollary 2.10 (Lehel) *If a hypergraph is totally balanced, then it is both a tree hypergraph and a strong Helly hypergraph.*

Proof. Suppose H is totally balanced. Theorem 2.8 implies H is a tree hypergraph. Since every cycle of H of length three has an edge containing at least three vertices of the cycle, Theorem 2.9 can be used to show that H is strong Helly. □

Exercise 2.23 Use the hypergraph $H = (\{0, 1, 2, 3, 4\}, \{S_1, S_2, S_3, S_4\})$ with $S_1 = \{0, 2, 3\}$, $S_2 = \{0, 3, 4\}$, $S_3 = \{0, 1, 4\}$, and $S_4 = \{0, 1, 2\}$ to show that the converse to Corollary 2.10 fails.

Totally balanced hypergraphs also play an important role with respect to "strongly chordal graphs," as discussed in section 7.12, as do strong Helly hypergraphs with respect to "hereditary clique-Helly graphs" in section 7.5.

2.4 Some Applications of Chordal Graphs

Each of the following subsections is merely a brief sketch of one application of chordal graphs. As an example of a type of application that is not represented in our collection, [Chandrasekaran & Tamir, 1982] studies the location of "supply centers" on a network having a tree structure. Section 3.4 consists of additional examples when the tree representations are required to be paths.

2.4.1 Applications to Biology

Let S denote a given set of molecular sequences, where each sequence corresponds to a *taxon* (an organism). For simplification, assume each sequence in S has length k and is built from the four letter alphabet $B = \{\mathsf{A},\mathsf{C},\mathsf{G},\mathsf{T}\}$; thus each corresponds to a DNA sequence on the four *bases* A, C, G, and T. (Protein sequences are similarly built from a 20 letter alphabet.) In the language of numerical taxonomy, the taxa (organisms) are described by k *characters*, each having one of four possible *states*. These characters can be represented as functions $f_1, \ldots, f_k$ where $f_i : S \to B$ with $f_i(x)$ the base at position i for taxon $x \in S$. Note that each character f_i induces a partition of the set S of taxa into at most four nonempty equivalence classes, the preimages of the bases in B.

Compatibility analysis seeks to find collections of characters from among $f_1, \ldots, f_k$ that are *compatible* (consistent) in that there exists a tree T with $S \subseteq V(T)$ on which, for each f_i in the collection, each equivalence class of f_i corresponds to a subtree of T. If a collection of characters is not compatible—if there is no such tree—then insofar as the evolutionary history for S is a tree, the true evolutionary history for S is not reflected in those characters. Thus compatibility analysis, considered as a consistency test, is a valuable method in that it can tell us something definite (albeit negative) about evolutionary aspects of certain characters.

Now suppose B is any finite set of states and each character f_i corresponds to a partition $P_i = \{X_1^{(i)}, \ldots, X_{m_i}^{(i)}\}$ of S, with each $m_i \leq |B|$ and $X_s^{(i)} \neq \emptyset$. Note that it is possible to have $X_s^{(i)}$ and $X_t^{(j)}$ equal as sets, even though $i \neq j$. Define the *partition intersection graph* $\Omega(P_1, \ldots, P_k)$, $k \geq 2$, to have vertices $\{X_1^{(1)}, \ldots, X_{m_1}^{(1)}, \ldots, X_1^{(k)}, \ldots, X_{m_k}^{(k)}\}$, with $X_s^{(i)}$ adjacent to $X_t^{(j)}$ if and only if $i \neq j$ and $X_s^{(i)} \cap X_t^{(j)} \neq \emptyset$. Notice that $\Omega(P_1, \ldots, P_k)$ is a k-partite graph with chromatic number k, and each taxon $x \in S$ corresponds to a maxclique of order k. [McMorris & Meacham, 1983] characterizes all graphs that arise this way.

Theorem 2.11 (McMorris & Meacham) *A graph with chromatic number $k > 1$ is a partition intersection graph if and only if it has no isolated vertices and has an edge clique cover, each member of which has order k.*

Proof. Assume G has chromatic number $k > 1$.

First suppose $G \cong \Omega(P_1, \ldots, P_k)$. Check that (1) each $X_s^{(i)} \in V(G)$ has at least $k - 1 \geq 1$ neighbors; (2) for each taxon $x \in S$ and each $\mathsf{B} \in B$, $\{X_s^{(i)} : f_i(x) = \mathsf{B}\}$ induces a complete subgraph of order k in G; and (3) the family of all such induced subgraphs is an edge clique cover of G.

Conversely, suppose $\mathcal{E} = \{Q_1, \ldots, Q_m\}$ is an edge clique cover for G where each $|Q_j| = k$, G has been *properly k-colored* (meaning that no two adjacent vertices have the same color), and no vertex of G is isolated. For each $v \in V(G)$, set $\mathcal{E}_v = \{Q_j : v \in Q_j\}$. Since each $v \in V(G)$ is on at least one edge of G and that edge is in at least one member of the edge clique cover $\mathcal{E}$, each $\mathcal{E}_v \neq \emptyset$. Suppose $\{v_1, \ldots, v_\ell\}$ is any one of the k color classes. Then each $Q_i \in \mathcal{E}$ will contain exactly one of $v_1, \ldots, v_\ell$ and so will be contained in exactly one of $\mathcal{E}_{v_1}, \ldots, \mathcal{E}_{v_\ell}$. Since $\mathcal{E}_{v_1}, \ldots, \mathcal{E}_{v_\ell}$ are disjoint subsets of $\mathcal{E}$, they partition $\mathcal{E}$. So each of the k color classes corresponds to one of k partitions $P_1, \ldots, P_k$ of $\mathcal{E}$. Moreover, $uv \in E(G)$ if and only if, for some i, both $u, v \in Q_i$ where u and v are in different color classes. But this is equivalent to $Q_i \in \mathcal{E}_u \cap \mathcal{E}_v$ with $\mathcal{E}_u, \mathcal{E}_v$ in different partitions among $P_1, \ldots, P_k$, which in turn is equivalent to $\mathcal{E}_u\mathcal{E}_v$ being an edge of $\Omega(\{P_1, \ldots, P_k\})$. Hence $G \cong \Omega(\{P_1, \ldots, P_k\})$. □

In the case of just two characters f_i and f_j, being compatible is equivalent to the bipartite graph $\Omega(P_i, P_j)$ being acyclic. Pairwise compatibility can be used to construct a *compatibility graph*, using the characters as vertices with adjacency corresponding to pairwise compatibility. Compatibility analysis seeks the largest collections of compatible characters. In the special case where every character has only two possible states, [McMorris, 1977] shows that maxcliques of the compatibility graph correspond to maximal compatible collections of characters. See also [Gusfield, 1991]. However, this fails in general, as shown in [Fitch, 1977] and by an infinite family of examples in [Meacham, 1983].

By assigning each character f_i (and so each corresponding partition P_i of S) a color i and coloring each vertex $X_s^{(i)}$ of $\Omega(P_1, \ldots, P_k)$ with color i, we have a *chromatic chordal completion problem*, as in [Buneman, 1974]: Given a graph whose vertices are properly k-colored, determine whether edges can be added between vertices of different colors in order to make the graph

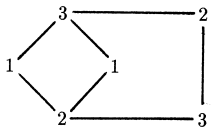

Figure 2.3: *A graph with no chromatic chordal completion.*

chordal. Edges can be added in this way if and only if the collection of characters is compatible.

Example 2.8 The graph shown in Figure 2.3, 3-colored with "colors" 1, 2, and 3, cannot be made chordal by adding edges between vertices of different colors: The second 2-3 edge would have to be added to eliminate the length-four "1,2,1,3" cycle, creating a new length-four "2,3,2,3" cycle that could not be eliminated.

Finding the complexity of the chromatic chordal completion problem was posed in [McMorris & Meacham, 1983]. Note that the chromatic restriction is important since an arbitrary uncolored graph can obviously be made into a chordal graph; thus the only problem in the single color case is to find minimal and minimum such sets of edges. See [Rose & Tarjan, 1975] and [Rose, Tarjan, & Lueker, 1976] for the complexity of these problems.

Recently, there has been a lot of activity in assessing the computational complexity of all the variations on the chromatic chordal completion problem. See [Bodlaender & Kloks, 1993], [McMorris, Warnow, & Wimer, 1994], [Agarwala & Fernández-Baca, to appear], [Kannan & Warnow, 1992, 1994], [Indury & Schaeffer, 1993], and [Bodlaender, Fellows, & Warnow, 1992].

There is still a lot of theoretical work to do before these results can be useful for compatibility analysis. For example, how can a largest set of compatible characters be selected when chromatic chordal completion is impossible?

2.4.2 Applications to Computing

The application discussed in subsection 2.4.1 is an example of a general "filiation" problem, determining whether certain data or objects are compatible with arrangement in a tree pattern. Corresponding "seriation" problems, with arrangement in a linear pattern, will be discussed in Chapter 3.

Another filiation application occurs in computer science. A *database scheme* can be thought of as a collection of tables—for instance, a manager's table with columns for "employee name," "social security number," "position," "job skills," etc.; a payroll table with columns for "social security number," "date of employment," "salary," etc.; a receptionist's table with columns for "employee name," "telephone extension," "hours," etc.; and so on—with the tables called *relations* and their columns called *attributes*.

Suppose a database scheme consists of a family $\mathcal{R}$ of relations and a set X of attributes (so $(X, \mathcal{R})$ is a hypergraph). This is an *acyclic database scheme* if the relations in $\mathcal{R}$ can be arranged as the vertices of a tree, commonly called a *join tree*, such that the vertices containing any given attribute induce a subtree. Join trees are like tree representations, and paths within the join tree constitute unique retrieval paths for data. Having a join tree representation is one of a large number of desirable properties of database schemes—matters of consistency, efficiency, and compatibility—that are shown to be equivalent to each other in [Beeri, Fagin, Maier, & Yannakakis, 1983], which also cites evidence that database schemes that possess these desirable properties are "sufficiently general to encompass most 'real-world' situations." [Golumbic, 1988] provides a simple introduction.

In terms of graphs, define $G = G(\mathcal{R})$ to have $V(G) = X$ with $E(G) = \{xy : x, y \in R \in \mathcal{R}\}$ (so $\mathcal{R}$ is an edge clique cover of $G(\mathcal{R})$).

Proposition 2.12 *A database scheme $\mathcal{R}$ is an acyclic database scheme if and only if each complete subgraph of $G(\mathcal{R})$ is contained in a common member of $\mathcal{R}$ and $G(\mathcal{R})$ is a chordal graph.* □

In terms of hypergraphs, $\mathcal{R}$ is an acyclic database scheme if and only if the dual of the hypergraph $(X, \mathcal{R})$ is a tree hypergraph; Proposition 2.12 then corresponds to Theorem 2.7. (Warning: There are many different notions of "cycle" and "acyclic" in use for hypergraphs, and being "acyclic" very often means something different from not having a "cycle"; acyclic database schemes correspond to what are often called "α-acyclic" hypergraphs.)

Subsection 2.4.4 will mention a somewhat related role of chordal graphs connected with expert systems.

Here is another, completely different application in computing: Many problems that are NP-complete in general become tractable, sometimes even solvable in linear time, when restricted to chordal graphs. While for many people this is the most important application of chordal graphs, it often

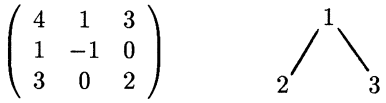

Figure 2.4: *A matrix M and its graph $G(M)$.*

involves little of the specific nature of chordal graphs as intersection graphs—other highly structured families can do as well. [Klein, 1996] discusses similar computational concerns for parallel computing.

[Chung & Mumford, 1994] is an example of such computational concerns, dealing with problems arising in computer vision. The specific problem faced is to determine bounds on the number of edges needed to be added to make a nonchordal graph chordal—the *minimum fill-in problem*—and so susceptible to more efficient algorithms. [Kloks, Bodlaender, Müller, & Kratsch, 1993], [Kloks & Kratsch, 1994], and [Parra & Scheffler, 1995, 1997], for instance, discuss how to use the minimal vertex separators of the nonchordal graph to determine how to add a minimal set of edges. Similar concerns also arise with sparse matrix computation in subsection 2.4.3 and maximum likelihood estimation in subsection 2.4.4.

2.4.3 Applications to Matrices

Gaussian elimination on an $n \times n$ matrix $M = (m_{ij})$ involves the choice of a nonzero *pivot* entry m_{ij}, then using elementary row and column operations to change m_{ij} into 1 and all other ith row and jth column entries into 0. An *elimination scheme* is a sequence of n pivots used to reduce a matrix to the identity matrix, and a *perfect elimination scheme* has the further property that no zero entry is ever made nonzero along the way. Perfect elimination schemes minimize both computation and data storage.

The *graph of* M, denoted $G(M)$, has vertex set $\{1, \ldots, n\}$, with vertices $i \neq j$ adjacent if and only if either $m_{ij} \neq 0$ or $m_{ji} \neq 0$.

Example 2.9 Figure 2.4 shows a matrix and its graph.

Proposition 2.13 is from [Rose, 1970] and has led to much further work; see [Rose, 1972], Chapter 12 of [Golumbic, 1980], [Golumbic, 1984], and various papers in [George, Gilbert, & Liu, 1993].

Proposition 2.13 (Rose) *A symmetric matrix M with nonzero diagonal entries has a perfect elimination scheme using the diagonal entries as pivots if and only if $G(M)$ is chordal.*

Proof sketch. Pivoting on m_{ii} results in removing all the edges incident to vertex i in $G(M)$ and simultaneously creating a new edge hj whenever $m_{hi} \neq 0 \neq m_{ij}$ but $m_{hj} = 0$, as below:

$$\begin{pmatrix} & \vdots & & \vdots & \\ \dots & m_{ii} & \dots & m_{ij} & \dots \\ & \vdots & & \vdots & \\ \dots & m_{hi} & \dots & 0 & \dots \\ & \vdots & & \vdots & \end{pmatrix}.$$

(Other entries might also inadvertently become zero in M, and so other edges disappear from $G(M)$.) Hence no zero entry is made nonzero in M precisely when every two neighbors of i (h and j above) are adjacent in $G(M)$; equivalently, when i is a simplicial vertex. (For instance, in the matrix in Example 2.9, you could pivot on either of the entries -1 or 2 but not on 4.) A perfect elimination scheme on the diagonal entries of M thus corresponds to a perfect elimination ordering for $G(M)$, and such a perfect elimination scheme exists if and only if G is chordal. □

The remainder of this subsection will describe a less practical, but more surprising, appearance of chordal graphs in matrix analysis.

It is trivial to compute the determinant of A when A^{-1} is a diagonal matrix: $\det A = \prod_{i=1}^{n} a_{ij}$. There are also simple methods to compute $\det A$ whenever A^{-1} is known to be "tridiagonal," meaning that the entries b_{ij} of A^{-1} are zero whenever $|i - j| > 1$. Observe that $G(A^{-1})$ is then a union of paths. In the early 1980s, this was generalized to larger families of matrices, including, in [Klein, 1982], when A^{-1} is "treediagonal" (meaning that $G(A^{-1})$ is a tree). This work culminated in Proposition 2.14 from [Barrett & Johnson, 1984] ("reinventing" chordal graphs). See [Barrett, Johnson, & Lundquist, 1989] and [Johnson, 1990] for more recent surveys of where this led next. For any $n \times n$ matrix M and any set $S \subseteq \{1, \dots, n\}$ (or any subgraph S of $G(M)$), let $M[S]$ denote the submatrix determined by those rows and columns of M indexed by the elements of S.

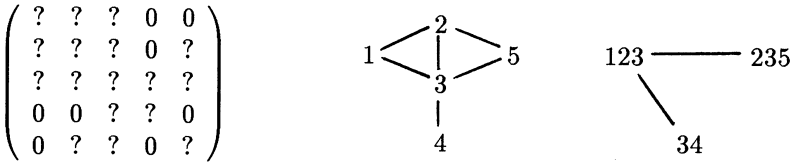

Figure 2.5: *A family of matrices having the same chordal graph G, with one clique tree for G.*

Proposition 2.14 (Barrett & Johnson) *If $G(A^{-1})$ is chordal with clique tree T, then*

$$\det A = \frac{\prod_{Q \in V(T)} \det A[Q]}{\prod_{Q_i Q_j \in E(T)} \det A[Q_i \cap Q_j]}, \tag{2.2}$$

provided the denominator is nonzero.

Example 2.10 Suppose A is any matrix whose inverse A^{-1} is as shown in Figure 2.5, with the ? entries unspecified (possibly zero). The graph $G(A^{-1})$ is shown in the middle, and one of the two possible clique trees T for $G(A^{-1})$ is shown at the right; for either clique tree, $E(T) = \{\{2,3\},\{3\}\}$. Formula (2.2) becomes

$$\begin{aligned}\det A &= \frac{\det A[\{1,2,3\}] \cdot \det A[\{2,3,5\}] \cdot \det A[\{3,4\}]}{\det A[\{2,3\}] \cdot \det A[\{3\}]} \\ &= \frac{\det \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} \cdot \det \begin{pmatrix} a_{22} & a_{23} & a_{25} \\ a_{32} & a_{33} & a_{35} \\ a_{52} & a_{53} & a_{55} \end{pmatrix} \cdot \det \begin{pmatrix} a_{33} & a_{34} \\ a_{43} & a_{44} \end{pmatrix}}{\det \begin{pmatrix} a_{22} & a_{23} \\ a_{32} & a_{33} \end{pmatrix} \cdot a_{34}}.\end{aligned}$$

Proof sketch. Suppose $G = G(A^{-1})$ is chordal with clique tree T. Suppose L is any leaf vertex of T and R is the set of all entries that are in vertices of T other than L. Thus $L \cap R$ corresponds to the edge joining L to the rest of T. We show that

$$\det A = \frac{\det A[L] \cdot \det A[R]}{\det A[L \cap R]}, \tag{2.3}$$

from which Proposition 2.14 follows inductively.

Suppose, for convenience, that the elements of $L \setminus R$ come first in the matrices A and A^{-1}, with those in $L \cap R$ coming next and those not in L coming last. Thus the matrices consist of nonempty blocks as shown below, with two blocks of A^{-1} consisting entirely of zero entries, reflecting that no vertex in $L \setminus R$ is adjacent to any vertex not in L:

$$A = \left(\begin{array}{c|c|c} A_{11} & A_{12} & A_{13} \\ \hline A_{21} & A_{22} & A_{23} \\ \hline A_{31} & A_{32} & A_{33} \end{array}\right), \quad A^{-1} = \left(\begin{array}{c|c|c} B_{11} & B_{12} & 0 \\ \hline B_{21} & B_{22} & B_{23} \\ \hline 0 & B_{32} & B_{33} \end{array}\right).$$

It is easy to verify that

$$\det B_{11} \cdot \det B_{33} = \det \left(\begin{array}{c|c} B_{11} & 0 \\ \hline 0 & B_{33} \end{array}\right). \tag{2.4}$$

By a result of Jacobi from 1834 relating minors of A and A^{-1},

$$\det B_{11} \cdot \det A = \det \left(\begin{array}{c|c} A_{22} & A_{23} \\ \hline A_{32} & A_{33} \end{array}\right),$$

$$\det B_{33} \cdot \det A = \det \left(\begin{array}{c|c} A_{11} & A_{12} \\ \hline A_{21} & A_{22} \end{array}\right),$$

$$\text{and } \det \left(\begin{array}{c|c} B_{11} & 0 \\ \hline 0 & B_{33} \end{array}\right) \cdot \det A = \det A_{22}.$$

Multiplying both sides of (2.4) by $(\det A)^2$ and then using these three equalities gives

$$\det \left(\begin{array}{c|c} A_{22} & A_{23} \\ \hline A_{32} & A_{33} \end{array}\right) \cdot \det \left(\begin{array}{c|c} A_{11} & A_{12} \\ \hline A_{21} & A_{22} \end{array}\right) = \det A_{22} \cdot \det A,$$

from which (2.3) follows by dividing through by $\det A_{22}$. □

Perfect Gaussian elimination and determinantal formulas, the two topics of this subsection, can be interrelated as in [Bakonyi, 1992].

Exercise 2.24 (see [Grone, Johnson, Sá, & Wolkowicz, 1984]) Show that, in any chordal graph, new edges can be inserted one at a time, always maintaining a chordal graph, all the way up to a complete graph.

This exercise is important in another broad topic—"matrix completion problems." See [Johnson, 1990], [Bakonyi & Johnson, 1995], and [Johnson, Jones, & Kroschel, 1995] for examples. [Bakonyi & Johnson, 1996] deals directly with several algebraic characterizations of chordal graphs.

2.4.4 Applications to Statistics

The application of chordal graphs to statistics dates from the seminal paper [Darroch, Lauritzen, & Speed, 1980] (another "reinvention" of chordal graphs). Recent textbook introductions to this use of graphs include [Santner & Duffy, 1989], [Wickens, 1989], [Christensen, 1990], [Whittaker, 1990], and [Lauritzen, 1996]. [Khamis & McKee, 1997] is a guide to this literature written for graph theorists, while [McKee & Khamis, 1996] and [Khamis, 1996] present a multigraph approach to some of the same topics. [Lauritzen & Spiegelhalter, 1988], [Pearl, 1988], and [Neapolitan, 1990] move on into the propagation of probabilistic evidence in expert systems.

For a set $\{1, \ldots, d\}$ of *factors*, a *level* ℓ_i of factor i is an allowable value of factor i. We denote the common occurrence of levels $\ell_1, \ldots, \ell_d$ by the conjunction $\bigwedge\{\ell_i : 1 \leq i \leq d\}$. A ("$d$-dimensional hierarchical loglinear") *model* M consists of a set of *generators*: incomparable subsets of $\{1, \ldots, d\}$. Generators correspond to inclusion-maximal sets of factors having interrelationships taken to be significant within the model. The *interaction graph of* M, denoted $G(M)$, has the factors as vertices, with two adjacent whenever the factors are in a common generator; thus the generators of M form an edge clique cover of $G(M)$. If the generators of M are precisely the maxcliques of $G(M)$, then M is called a *graphical model*.

Example 2.11 Suppose $d = 7$ where factor 1 corresponds to "sex" with $\ell_1 \in$ {male, female}, factor 2 is "age" with $\ell_2 \in$ {under 30, 30–45, 46–60, over 60}, and so on. Suppose the generators are $\{1, 5\}$, $\{2, 4, 5\}$, $\{3, 4, 5, 6\}$, and $\{4, 5, 6, 7\}$, so $G(M)$ is the graph shown on the left in Figure 2.1. In this model, the interaction of factor 1 ("sex") is not taken as important with factor 2 ("age"), but only with factor 5 (perhaps "occupation").

Choosing which model to apply to observed data involves various statistical techniques that are not of concern here. But not all models are equally easy to make inferences from. Those with particularly desirable properties, usually called *decomposable models*, were shown in [Darroch, Lauritzen, & Speed, 1980] to be precisely those that have chordal interaction graphs.

Suppose a large population is sampled and, for that sample, the number of individuals having each possible combination of levels of factors is determined. The principal advantage of a decomposable model with g generators for these data is that the predictive value of all the data is contained in knowing these numbers for a small number of special sets of factors: the g generators and a certain $g - 1$ intersections of pairs of generators. Proposition 2.15, from [Darroch, Lauritzen, & Speed, 1980], shows how this is

done. In this proposition, for any $S \subseteq \{1, \ldots, d\}$, $\hat{p}(\bigwedge\{\ell_i : i \in S\})$ denotes the number of individuals in the sample who have the combination $\bigwedge\{\ell_i : i \in S\}$ of levels of factors in S, divided by the total number of individuals in the sample. This proposition actually characterizes decomposable models, and so chordal interaction graphs, and replaces iterative methods that are needed in the general case.

Proposition 2.15 (Darroch, Lauritzen, & Speed) *If $G(M)$ is chordal with clique tree T, then for every choice $\ell_1, \ldots, \ell_d$ of levels of the factors,*

$$\hat{p}\left(\bigwedge\{\ell_i : i \in V(G)\}\right) = \frac{\prod_{Q \in V(T)} \hat{p}(\bigwedge\{\ell_i : i \in Q\})}{\prod_{Q_i Q_j \in E(T)} \hat{p}(\bigwedge\{\ell_i : i \in Q_i \cap Q_j\})}, \tag{2.5}$$

provided the denominator is nonzero.

Proof sketch. Suppose $G = G(M)$ is chordal with clique tree T. Suppose L is any leaf vertex of T and R is the set of all factors that are in vertices of T other than L. Thus $L \cap R$ corresponds to the edge joining L to the rest of T. We show that

$$\hat{p}\left(\bigwedge\{\ell_i : i \in V(G)\}\right) = \frac{\hat{p}(\bigwedge\{\ell_i : i \in L\}) \cdot \hat{p}(\bigwedge\{\ell_i : i \in R\})}{\hat{p}(\bigwedge\{\ell_i : i \in L \cap R\})}, \tag{2.6}$$

from which Proposition 2.15 follows inductively.

For any set S of factors, let $\bigwedge S$ abbreviate the compound event of each factor $i \in S$ having level ℓ_i. Then $\hat{p}(\bigwedge\{\ell_i : i \in S\})$ approximates, for the entire population sampled, the joint probability of the compound event $\bigwedge S$; abbreviate this probability by $\Pr[\bigwedge S]$. Then (2.6) corresponds to

$$\Pr\left[\bigwedge V(G)\right] = \frac{\Pr[\bigwedge L] \cdot \Pr[\bigwedge R]}{\Pr[\bigwedge(L \cap R)]}. \tag{2.7}$$

Since $L \cap R$ corresponds to the edge of T joining L to the rest of T, every path from a vertex of $L \backslash R$ to a vertex of $R \backslash L$ passes through a vertex of $L \cap R$. This means that the compound events $\bigwedge(L \backslash R)$ and $\bigwedge(R \backslash L)$ are conditionally independent, conditioning on $\bigwedge(L \cap R)$; in symbols,

$$\begin{aligned} \Pr\left[\bigwedge(L \backslash R) \mid \bigwedge(L \cap R)\right] \cdot \Pr[\bigwedge(R \backslash L) \mid \bigwedge(L \cap R)] \\ = \Pr[\bigwedge(L \backslash R \cup R \backslash L) \mid \bigwedge(L \cap R)]. \end{aligned} \tag{2.8}$$

By the definition of conditional probability,

$$\Pr\left[\bigwedge(L \backslash R) \mid \bigwedge(L \cap R)\right] = \frac{\Pr[\bigwedge L]}{\Pr[\bigwedge(L \cap R)]},$$

$$\Pr\left[\bigwedge(R\backslash L) \mid \bigwedge(L\cap R)\right] = \frac{\Pr[\bigwedge R]}{\Pr[\bigwedge(L\cap R)]},$$

and

$$\Pr[\bigwedge(L\backslash R \cup R\backslash L) \mid \bigwedge(L\cap R)] = \frac{\Pr[\bigwedge(L\backslash R \cup R\backslash L \cup (L\cap R))]}{\Pr[\bigwedge(L\cap R)]} = \frac{\Pr[\bigwedge V(G)]}{\Pr[\bigwedge(L\cap R)]}.$$

Using these three equalities in (2.8) and then multiplying through by $\Pr[\bigwedge(L\cap R)]^2$ gives

$$\Pr\left[\bigwedge L\right]\cdot\Pr\left[\bigwedge R\right] = \Pr\left[\bigwedge V(G)\right]\cdot\Pr\left[\bigwedge(L\cap R)\right],$$

from which (2.7) follows by dividing through by $\Pr[\bigwedge(L\cap R)]$ and using $\Pr[\bigwedge S]\approx\hat{p}(\bigwedge\{\ell_i : i\in S\})$ again. □

Propositions 2.14 and 2.15 are obviously similar in form, each using a product over $V(T)$ divided by a product over $E(T)$. This similarity is no coincidence, as may be sensed from the proof sketches, but a more abstract viewpoint is needed in order to be precise; [Speed & Kiiveri, 1986] and [McKee, 1993] present such viewpoints.

2.5 Split Graphs

A graph G is a *split graph* if $V(G)$ can be partitioned into $Q\cup I$, where Q induces a complete graph and I induces an edgeless graph; thus G has $|Q|(|Q|-1)/2$ edges within Q and anywhere from zero to $|Q|\cdot|I|$ other edges, between Q and I. Split graphs were introduced in [Földes & Hammer, 1977a]; also see Chapter 6 of [Golumbic, 1980]. (Warning: [Földes & Hammer, 1977b] gives a different meaning to "split.") They were independently studied in [Tyškevič & Černjak, 1978a, 1978b, 1979]. While split graphs may seem too special to be of interest, the theorem and corollary in this section guarantee the place of split graphs in intersection graph theory.

Exercise 2.25 Show that the definition of split graphs could have equivalently required that Q be a maxclique of G.

Theorem 2.16 (Földes & Hammer) *A graph G is a split graph if and only if both G and its complement $\overline{G}$ are chordal.*

Proof. First suppose G is a split graph. Then it is easy to see that $\overline{G}$ is also split and that both G and $\overline{G}$ are chordal.

Conversely, suppose that G and $\overline{G}$ are both chordal. Suppose T is a minimum-diameter clique tree of G and, arguing toward a contradiction, that the diameter of T is at least three, and so T is not a star. Then there exist vertices Q_1 and Q_2 that are the middle two vertices in an induced P_4 in T. Among the other neighbors of Q_1 in T there must be a Q_0 for which there exists a vertex $v \in (Q_0 \cap Q_1) \backslash Q_2$, since otherwise all the neighbors (other than Q_2) of Q_1 in T could be made adjacent to Q_2 instead of Q_1 and so create a clique tree for G with smaller diameter than T. Let u be a vertex in $Q_0 \backslash Q_1$. Similarly, there is a neighbor $Q_3 \neq Q_1$ of Q_2 and vertices $w \in (Q_2 \cap Q_3) \backslash Q_1$, and $x \in Q_3 \backslash Q_2$. Then $\{u, v, w, x\}$ induces a subgraph of G with edge set $\{uv, wx\}$. But that would force an induced C_4 in $\overline{G}$, contradicting that $\overline{G}$ is chordal. □

Exercise 2.26 (Földes & Hammer) Show that a graph is a split graph if and only if none of its induced subgraphs is isomorphic to $2K_2$, C_4, or C_5.

The following corollary, from [McMorris & Shier, 1983], characterizes split graphs as intersection graphs. (Compare it with Exercise 2.3.) Recall that a *star* is a graph isomorphic to $K_{1,n}$ ($n \geq 0$); in other words, a tree T with diameter at most two. A *substar* of a star is then simply a subtree of the star.

Corollary 2.17 (McMorris & Shier) *A graph is a split graph if and only if it is the intersection graph of a set of* distinct *substars of a star.*

Proof. First suppose G is a split graph. Then G is chordal by Theorem 2.16, and by its proof any minimum-diameter clique tree T for G has diameter less than three and so is a star. Then G is the intersection graph of the substars $\{T_v : v \in V(G)\}$ of that star. If the diameter of T is two, then these subtrees are distinct; if the diameter is one, then leaves can be added for each $v \in V(G)$ to produce distinct subtrees.

Conversely, suppose G is the intersection graph of a set of distinct substars of a star T. We can suppose that T is a clique tree for G and that $Q \in V(T)$ is adjacent to all other vertices of T. Then G is split, as shown by the partition of $V(G)$ into the complete subgraph Q and the independent subset $I = V(G) \backslash Q$. □

Exercise 2.27 Find an example of a graph that is not a split graph, yet is the intersection graph of a family of (not necessarily distinct) substars of a star. (What does Theorem 1.5 have to do with this?)