University of Alberta

Polygon Graph Recognition

by

E.S. Elmallah and L.K. Stewart

Technical Report TR 95–03
February 1995

DEPARTMENT OF COMPUTING SCIENCE
The University of Alberta
Edmonton, Alberta, Canada

Polygon Graph Recognition

E.S. Elmallah and L.K. Stewart
Department of Computing Science
University of Alberta
Edmonton, Alberta, T6G 2H1
CANADA

Abstract

For any fixed integer $k \geq 2$, define the class of k-polygon graphs as the intersection graphs of chords inside a convex k-polygon, where the endpoints of each chord lie on two different sides. The case where $k = 2$ is degenerate; for our purpose, we view any pair of parallel lines as a 2-polygon. Hence, polygon graphs are all circle graphs. Interest in such graphs arises since a number of intractable problems on circle graphs can be solved in polynomial time on k-polygon graphs, for any fixed k, given a polygon representation of the input graph. In this paper we show that determining whether a given circle graph is a k-polygon graph, for any fixed k, can be solved in $O(4^k n^2)$ time. The algorithm exploits the structure of a decomposition tree of the input graph and produces a k-polygon representation, if one exists. In contrast, we show that determining the minimum value of k is NP-complete. [1]

Keywords: graph algorithms, graph theory

[1]This research is partially supported by NSERC Operating Grants

1. Introduction

This paper investigates the complexity of the following class of recognition problems: given an undirected graph $G = (V, E)$, and an integer k, $k \geq 2$, is G a k-polygon graph? That is, does there exist a one-to-one mapping between V and chords of a k-polygon such that $(v_i, v_j) \in E$ if and only if their corresponding chords intersect? The case where $k = 2$ is degenerate but important; for our purpose we view any pair of parallel lines as a 2-polygon. If the answer is positive, then we are interested in generating the corresponding intersection diagram. We henceforth let $|V| = n$ and $|E| = m$.

Two important cases of the above problem are well-solved. At one extreme, the problem for $k = 2$ calls for recognizing permutation graphs, defined also as follows: a graph $G = (V, E)$ on n vertices is a permutation graph if there is a labelling $\{v_1, v_2, \cdots, v_n\}$ of the vertices and a permutation π of $\{1, 2, \cdots, n\}$ such that for every possible $i < j$, $(v_i, v_j) \in E$ if j appears before i in π. Equivalently, [6] shows that G is a permutation graph if and only if both G and its complement are comparability graphs. Using the above characterization, [6] devised a recognition algorithm based on computing transitive orientations. The fastest algorithm, however, for recognizing such graphs, and constructing associated permutation diagrams, is due to Spinrad [12] and runs in $O(n^2)$ time.

At the other extreme, the problem for $k \geq n$ is essentially that of recognizing circle graphs (overlap graphs). The first known polynomial time recognition algorithms for this class are due to Bouchet [1], and independently Naji [11]. Subsequent improvements of the running time to $O(nm)$ and $O(n^2)$ are due to Gabor, Supowit and Hsu [7] and Spinrad [13], respectively. The interested reader may find several other algorithmic aspects of permutation graphs and circle graphs in Golumbic [9].

Our interest in solving the above problem for any arbitrary k arises since a number of combinatorial optimization and enumeration problems that appear to be intractable on circle graphs admit polynomial time solutions on k-polygon graphs, given a polygon diagram of the input graph (see for example [4] and [5]). Some such combinatorial problems arise in real-world applications, with VLSI routing problems being a good case in point.

Our contribution in resolving the above problem is two-fold. We first devise an $O(4^k n^2)$ time algorithm for solving the problem, for any fixed k. The algorithm takes as input a special tree representation of the input circle graph, as will be mentioned shortly, and an integer k; it produces a k-polygon representation of G, if one exists (Theorem 5.1). Second, we show that computing the minimum value $k(G)$ for which G is a $k(G)$-polygon graph is NP-complete (Theorem 7.1). The former result is the first known asymptotic upper bound for solving the above general problem. The running time depends exponentially on k, and hence the algorithm appears to be practical only for small k. Nevertheless, the algorithm admits speedup using parallelization techniques. The NP-completeness result, on the other hand, shows that such asymptotic behaviour is probably unavoidable. Finally, we conclude by showing that if $G = \bigcup_{i=1}^{r} G_i$ is a disconnected circle graph with r components then $k(G) = (\sum_{i=1}^{r} k(G_i)) - 2(r-1)$.

The recognition algorithm has two main ingredients (mentioned below): Theorem C82 due to Cunningham [2], and Theorem GSH89 due to Gabor, Supowit, and Hsu [7]. To start, we need to reproduce from [2] some basic definitions and results related to the following notion of graph decomposition: let $G = (V, E)$ be a finite undirected graph, and let $\{V_1, V_2\}$ be a partition of V. Call $\{V_1, V_2\}$ a split of V if:

(i) $|V_1|, |V_2| \geq 2$, and

(ii) there exist $W_1 \subseteq V_1$ and $W_2 \subseteq V_2$ such that the subset of edges $\{(v_1, v_2) \in E | v_1 \in V_1, v_2 \in V_2\}$ is precisely the set of all pairs $\{(v_1, v_2)| v_1 \in W_1, v_2 \in W_2\}$. We henceforth call W_1 and W_2 the interface vertices of the partition $\{V_1, V_2\}$.

Graphs that do not admit such a split are called prime. If $\{V_1, V_2\}$ is a split, then let $\ell \in V$ be a new vertex (called a marker). The simple $*$-decomposition of G associated with the split $\{V_1, V_2\}$ is a set $\{G_1, G_2\}$ of graphs where, for $i = 1, 2$, G_i is obtained from the induced subgraph $G[V_i]$ by adding the marker ℓ and making it adjacent to every vertex in W_i. A decomposition of G is defined inductively to be either $\{G\}$, or a set of graphs obtained from a decomposition $\mathcal{M}$ of G by replacing a member G_1 of $\mathcal{M}$ by the members of a simple $*$-decomposition of G_1, where the marker of this simple decomposition is not an element of any member of $\mathcal{M}$. As mentioned in [2], it is then possible to associate a $*$-decomposition tree T with any decomposition $\mathcal{M}$. The vertices of T are the members of $\mathcal{M}$, and the edges correspond to the markers of $\mathcal{M}$; each edge joins in T the two graphs of $\mathcal{M}$ of which the corresponding marker is a vertex.

Two partitions $\{X, \overline{X}\}$ and $\{Y, \overline{Y}\}$ of V cross if each of the four possible intersections between the X's and the Y's is non-empty. A split is said to be good if it is crossed by no other split of G. Using the theory of decomposition frames developed in [3], Cunningham showed in [2] that the set of all good splits of a diconnected graph G (note: G is diconnected if for every $\phi \subset A \subseteq V$ there exists an arc (v_1, v_2) with $v_1 \in A$ and $v_2 \notin A$) generate a unique decomposition (the standard $*$-decomposition) each of whose members has no good split. In addition, [2] characterizes diconnected graphs that have no good splits. As a special case, the following characterization applies to undirected graphs (a similar characterization applies to symmetric diconnected graphs):

Theorem C82 [2]: Each connected undirected graph has a unique standard $*$-decomposition, each of whose members is prime, complete, or a star.

Several algorithms for computing such a standard $*$-decomposition exist in the literature. The first algorithm, due to Cunningham, runs in $O(n^3)$ time and applies to undirected and symmetric diconnected graphs. In the special case of undirected graphs, the bound has been improved in [7] to $O(nm)$, and subsequently reduced to $O(n^2)$ in [10]. Our recognition algorithm requires the standard $*$-decomposition tree T of the input graph, and hence, we may assume that T is computed using the latter algorithm. Note that each node in T corresponds to a circle graph, since G is a circle graph.

To present the second ingredient, we recall the following notion of uniquely representable circle graphs from [7]. Let D be a circle diagram of $G = (V, E)$, $|V| = n$, described by a sequence $\pi = (a_1, a_2, \cdots, a_{2n})$ of the $2n$ endpoints of chords in a clockwise traversal of the circle (starting at an arbitrary point). A <u>shift</u> of π results in the new sequence $(a_{2n}, a_1, a_2, \cdots, a_{2n-1})$, and a <u>reversal</u> of π results in the sequence $(a_{2n}, \cdots, a_2, a_1)$. A circle graph G is <u>uniquely representable</u> if for any two distinct diagrams of G one can be obtained from the other by a sequence of shifts and reversals. Thus, a uniquely representable graph has essentially one diagram that can be described by sequences that are equivalent under the order-preserving rewriting rules mentioned above. In our present context, it is important to note that if G is uniquely representable then the parameter $k(G)$ can be computed from such a unique diagram of G (as described, for example, in Section 6.2). Gabor et al. [7] proved the following elegant result:

Theorem GSH89 [7]: Let G be a circle graph with at least five vertices. Then G is prime (with respect to the $*$-decomposition) if and only if it is uniquely representable.

2. The Basic Model

Throughout the paper, let T be the standard $*$-decomposition tree of a connected input circle graph $G = (V, E)$. Each node x of T is a circle diagram, denoted D_x, that corresponds to either a prime graph, a star, or a clique, and each edge (x, y) is identified with a marker ℓ_{xy} that corresponds to a split of V. Hence, ℓ_{xy} appears in both D_x and D_y. Denote by $T(x)$ the subtree containing x in the forest $T \setminus (x, y)$. Let $G_{T(x)} = (V_{T(x)}, E_{T(x)})$ denote the graph whose $*$-decomposition tree is $T(x)$, and let $D_{T(x)}$ be any circle representation of $G_{T(x)}$. Define $T(y)$, $G_{T(y)}$, and $D_{T(y)}$ in a similar way, with respect to the other node y. Finally, if ℓ is a chord in a circle diagram D then ℓ splits the circle into two closed regions, called <u>semicircles</u> henceforth, each region contains ℓ and an arc of the circle bounded by the two endpoints of ℓ.

3. Noncrossing Splits and Circle Layouts

The main result in this section (Theorem 3.1, below) shows that if (x, y) is an edge in T, and $D_{T(x)}$ and $D_{T(y)}$ are as defined above, then the chords in $D_{T(x)}$ and $D_{T(y)}$ can only intersect in a limited way in any possible circle diagram D of G. The result is described using the following colouring scheme. First, set $B = V_{T(x)} \backslash \ell_{xy}$ and assume that each chord in B is coloured blue. Similarly, set $R = V_{T(y)} \backslash \ell_{xy}$ and assume that each chord in R is coloured red. Now, the marker ℓ_{xy} is associated with the split $\{B, R\}$ of V. In D, call a maximal arc that contains only red (or only blue) endpoints a zone of type red (or blue). Also, let $B_I \subseteq B$ (and $R_I \subseteq R$) be the set of interface chords for the split $\{B, R\}$ of V. Since $|B_I|, |R_I| \geq 1$, it follows that D has at least 2 blue zones, say β_0 and β_1, and at least two red zones, denoted ρ_0 and ρ_1, such that the sequence $(\beta_0, \rho_0, \beta_1, \rho_1)$ appears in a circular listing of D. An important fact then is:

Theorem 3.1: Let G and D be as defined above, then the above colouring scheme induces exactly 4 zones.

The following definitions will be used. If α_i and α_j are arcs in D, then $\alpha_{i,j}$ denotes the set of chords with one endpoint in α_i and the other in α_j. If α_i and α_j are two disjoint arcs (or zones) in D then let $\alpha_{\{i,j\}^2} = \alpha_{i,i} \bigcup \alpha_{i,j} \bigcup \alpha_{j,j}$. For any arc α, first(α) and last(α) denote the counterclockwise and clockwise endpoints, respectively, of α. Two distinct monocoloured zones, α_i and α_j, are said to be conjugates if every chord that has exactly one endpoint in one of these two zones has its other endpoint in the other zone.

The proof is broken into lemmas 3.1 - 3.4 below. To start, we derive a simple sufficient condition for the existence of an interface chord in certain arcs of the diagram.

Lemma 3.1: Let α_0 be a zone of colour $C \in \{B, R\}$ and let α'_0 be any arc that contains α_0 but does not contain all chord endpoints of D. Furthermore, suppose that no chord of colour C has exactly one endpoint in α'_0. Then $C_I \cap \alpha'_{0,0} \neq \phi$, where C_I denotes the set of interface chords of colour C.

Proof:

For simplicity, we may assume that α_0 is a red zone (then α_0 and α'_0 correspond to ρ_0 and ρ'_0, respectively, in Figure 3.1). Consider the red chords of $\alpha'_{0,0}$ and their connections to the rest of the chords of D. In particular, since G is connected, there must be a path connecting a red chord $r \in \alpha'_{0,0}$ to any chord ℓ having one or both endpoints outside of α'_0. Such a path must contain a blue chord having exactly one endpoint in α'_0. Consider traversing this path from r to ℓ, and let b be the first blue chord encountered. Since b is the first blue chord encountered, it must have at least one endpoint in α'_0 and it must cross a red chord, r', its predecessor on the path under consideration. All chords preceding b on the path are in $\alpha'_{0,0}$, since no red chord has exactly one endpoint in $\alpha'_{0,0}$. Therefore, $r' \in C_I \cap \alpha'_{0,0}$. This completes the proof of Lemma 3.1.

■

Lemma 3.2: Let $\ell' \in \alpha_{0,1}$ be a chord, where α_0 and α_1 are two distinct zones. Then α_0 and α_1 are conjugates.

Proof:

To prove the lemma, we need to show that if ℓ'' ($\ell'' \neq \ell'$) is a chord that has exactly one endpoint in α_0 (respectively α_1), then it has the other endpoint in α_1 (respectively α_0).

We may assume, without loss of generality, that $\alpha_0 = \beta_0$ and $\alpha_1 = \beta_1$ are two blue zones, as in Figure 3.1. To derive a contradiction, let $\ell'' \in \beta_{0,2}$ where $\beta_0 \neq \beta_2$, $\beta_1 \neq \beta_2$. Then there exist at least 3 red zones $\{\rho_i | i = 0,1,2\}$ such that the sequence $(\beta_i, \rho_i | \ i = 0,1,2)$ appears in a circular listing of the diagram D. For each zone ρ_i, $i \leq 2$, let ρ'_i be the arc $(\mathrm{last}(\beta_\mathrm{i}), \mathrm{first}(\beta_{\mathrm{i+1}}))$ (modulo 3) that includes ρ_i as a proper subinterval, as shown in Figure 3.1.

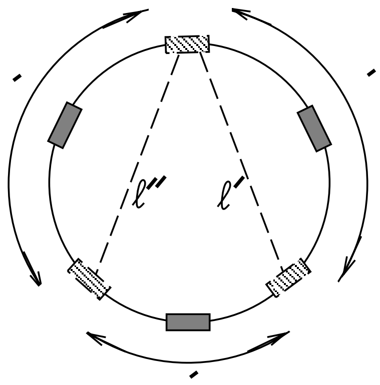


Figure 3.1 An impossible configuration

Observe that all of the following must hold, by the definition of interface chords:

(a) $R \cap \rho'_{0,1} = \phi$ or ($\ell' \in B_I$ and $\ell'' \notin B_I$),

(b) $R \cap \rho'_{1,2} = \phi$ or ($\ell' \notin B_I$ and $\ell'' \in B_I$),

(c) $R \cap \rho'_{0,2} = \phi$ or ($\ell' \in B_I$ and $\ell'' \in B_I$).

We consider all possible configurations of ℓ' and ℓ'', with respect to B_I.

Case (i): $\ell', \ell'' \in B_I$.

Then $R \cap \rho'_{0,1} = \phi$ and $R \cap \rho'_{1,2} = \phi$ by (a) and (b). That is, there is no red chord with exactly one of its endpoints in ρ'_1. Therefore, applying Lemma 3.1 to ρ'_1 leads to a contradiction, since any interface chord of $\rho'_{1,1}$ must cross $\ell', \ell'' \in B_I$.

Case (ii): neither ℓ' nor ℓ'' is in B_I.
Then all of $R \cap \rho'_{0,1}$, $R \cap \rho'_{1,2}$, and $R \cap \rho'_{0,2}$ are empty. By Lemma 3.1, we find a red interface chord in each of $\rho'_{0,0}$, $\rho'_{1,1}$, and $\rho'_{2,2}$. But no blue chord can cross three such red interface chords, contradicting their existence.

Case (iii): exactly one of ℓ', ℓ'' is in B_I.
Suppose without loss of generality that $\ell' \in B_I$ and $\ell'' \notin B_I$. Then $R \cap \rho'_{1,2} = \phi$ and $R \cap \rho'_{0,2} = \phi$ by (b) and (c), and therefore, by Lemma 3.1, $R_I \cap \rho'_{2,2} \neq \phi$. But any red interface chord in $\rho'_{2,2}$ does not cross $\ell' \in B_I$, a contradiction.

This completes the proof of Lemma 3.2.

■

Roughly speaking, the above lemma implies that if $\ell' \in \alpha_{0,1}$ is a chord then all possible "clusters" of chords in $\alpha_{0,0}$ and $\alpha_{1,1}$ are linked to each other and the rest of the graph by a "bundle" of chords in the set $\alpha_{0,1}$. Moreover, removing such a bundle disconnects each possible cluster from the remaining graph. It also follows that conjugation partitions the set of zones into equivalence classes, where each class has exactly two zones.

To complete the proof of the theorem, consider any two monocoloured chords, say $\{\ell', \ell''\} \subseteq B$. Then either ℓ' and ℓ'' have their endpoints in one zone, in exactly two zones, or in four zones (ℓ' and ℓ'' having their endpoints in exactly three zones is ruled out by Lemma 3.2); in the latter case, one of the following configurations occurs:

1. an <u>X-configuration</u> where ℓ' and ℓ'' cross but have no zone in common, or

2. a <u>||-configuration</u> where ℓ' and ℓ'' do not cross and have no zone in common.

We now show that neither of the above configurations occur.

Lemma 3.3: X-configurations are impossible.
Proof:
Assume to the contrary that $\ell' \in B$ induces an X-configuration with some other chord $\ell'' \in B$. Let $\ell' \in \beta_{0,2}$ and $\ell'' \in \beta_{1,3}$, as in Figure 3.2. Then there exist at least 4 red zones $\{\rho_i | \, i = 0, \cdots, 3\}$ such that $(\beta_i, \rho_i | \, i = 0, \cdots, 3)$ appear in a circular listing of D. For $0 \leq i \leq 3$, let ρ'_i be the arc $(\mathrm{last}(\beta_\mathrm{i}), \mathrm{first}(\beta_{\mathrm{i+1}}))$ (modulo 4), that includes ρ_i as a proper subinterval.

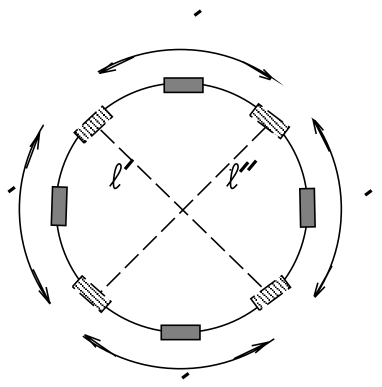


Figure 3.2 An X-configuration

We observe that all of the following must hold:
(a) ($R \cap \rho'_{0,1} = \phi$ and $R \cap \rho'_{2,3} = \phi$) or ($\ell' \notin B_I$ and $\ell'' \in B_I$),
(b) ($R \cap \rho'_{0,3} = \phi$ and $R \cap \rho'_{1,2} = \phi$) or ($\ell' \in B_I$ and $\ell'' \notin B_I$),
(c) ($R \cap \rho'_{0,2} = \phi$ and $R \cap \rho'_{1,3} = \phi$) or ($\ell' \in B_I$ and $\ell'' \in B_I$).

We consider all possible configurations of ℓ' and ℓ'', with respect to B_I.

Case (i): $\ell', \ell'' \in B_I$.
For $0 \leq i \leq 3$, each of the four sets $R \cap \rho'_{i,i+1}$ (modulo 4) is empty, by (a) and (b). Therefore, it must be that $|R \cap \rho'_{0,2}|, |R \cap \rho'_{1,3}| \geq 1$; otherwise, by Lemma 3.1, we find red interface chords in $\rho'_{0,0}$ and $\rho'_{2,2}$, or in $\rho'_{1,1}$, and $\rho'_{3,3}$, that do not cross ℓ' and ℓ'', a contradiction. Thus, we may assume that the set $\{\rho_i | \; i = 0, \cdots, 3\}$ is selected so that $|\rho_{0,2}|, |\rho_{1,3}| \geq 1$.

Suppose that the two sets $\beta_{\{0,2\}^2}$ and $\rho_{\{0,2\}^2}$ interchange their colours, and let

$$\begin{aligned} B' &= B \setminus \beta_{\{0,2\}^2} \cup \rho_{\{0,2\}^2} \\ R' &= R \setminus \rho_{\{0,2\}^2} \cup \beta_{\{0,2\}^2} \end{aligned}$$

Claim: $\{B', R'\}$ is a split of V with interface chords $B'_I = B_I \setminus \beta_{0,2} \cup \rho_{0,2}$ and $R'_I = R_I \setminus \rho_{0,2} \cup \beta_{0,2}$.
Proof: The claim follows from the following observations.

1. $\{B', R'\}$ is a partition of V and $|B'|, |R'| \geq 2$.
2. For all $b' \in B'$, $r' \in R'$, b' crosses r' if and only if $b' \in B'_I$ and $r' \in R'_I$.

The "only if" part is easy to verify. For the "if" part, we show that if $b' \in B_I \setminus \beta_{\{0,2\}^2}$ and $r' \in \beta_{\{0,2\}^2}$ then they cross. (The proof is similar for the case where $b' \in \rho_{\{0,2\}^2}$ and $r' \in R_I \setminus \rho_{\{0,2\}^2}$, and all other cases are easy to verify.)

Suppose that b' and r' do not cross. Then both endpoints of b' are on the arc $(\mathrm{last}(\beta_0), \mathrm{first}(\beta_2))$ or both are on the arc $(\mathrm{last}(\beta_2), \mathrm{first}(\beta_0))$. Suppose, without loss of generality, that $b' \in \beta_{i,j}$ and that the zones under consideration appear in a clockwise circular listing of D in the order: β_0, β_i, ρ_0, β_1, ρ_1, β_j, β_2. Consider the arc $\beta_{0,i} = (\mathrm{last}(\beta_0), \mathrm{first}(\beta_{\mathrm{i}}))$. There must be a red zone contained in $\beta_{0,i}$, and $\beta_{0,i}$ cannot contain exactly one endpoint of any red chord (since such a chord would cross some but not all of the blue interface chords, a contradiction), so by Lemma 3.1, there must be a red interface chord contained in $\beta_{0,i}$. However, such a chord does not cross ℓ' or ℓ'' and thus it existence leads to a contradiction.

The above observations complete the proof of the claim. □

The two splits $\{B', R'\}$ and $\{B, R\}$ cross each other, contradicting the goodness of the latter split. Hence, Case (i) is impossible.

Case (ii): neither ℓ' nor ℓ'' is in B_I.
Then, from (a), (b), and (c) above, and by Lemma 3.1, we find a red interface chord in each of $\rho'_{0,0}$, $\rho'_{1,1}$, $\rho'_{2,2}$, and $\rho'_{3,3}$. But no blue chord can cross four such red interface chords, contradicting their existence. Thus, Case (ii) is impossible.

Case (iii): exactly one of ℓ', ℓ'' is in B_I.
Suppose without loss of generality that $\ell' \in B_I$ and $\ell'' \notin B_I$. Then $R \cap \rho'_{0,1} = \phi$, $R \cap \rho'_{2,3} = \phi$, $R \cap \rho'_{0,2} = \phi$, and $R \cap \rho'_{1,3} = \phi$, by (a) and (c). Therefore, $R \cap \rho'_{0,3} \neq \phi$ and $R \cap \rho'_{1,2} \neq \phi$, else, by Lemma 3.1 we find red interface chords in $\rho'_{0,0}$ and $\rho'_{3,3}$, or in $\rho'_{1,1}$, and $\rho'_{2,2}$, that do not cross $\ell' \in B_I$, a contradiction. Thus, we may assume that the set $\{\rho_i | \; i = 0, \cdots, 3\}$ is selected so that $|\rho_{0,3}|, |\rho_{1,2}| \geq 1$.

Suppose that the two sets $\beta_{\{1,3\}^2}$ and $\rho_{\{0,3\}^2}$ exchange colours, and let

$$\begin{aligned} B' &= B \setminus \beta_{\{1,3\}^2} \cup \rho_{\{0,3\}^2} \\ R' &= R \setminus \rho_{\{0,3\}^2} \cup \beta_{\{1,3\}^2} \end{aligned}$$

Claim: $\{B', R'\}$ is a split of V with interface chords $B'_I = B_I$ and $R'_I = R_I \setminus \rho_{0,3} \cup \beta_{1,3}$.
Proof: The claim follows from the following observations.

1. $\{B', R'\}$ is a partition of V and $|B'|, |R'| \geq 2$.

2. For all $b' \in B'$, $r' \in R'$, b' crosses r' if and only if $b' \in B'_I$ and $r' \in R'_I$.

   It is easy to verify that, for all $b' \in B_I$, $r' \in R_I \setminus \rho_{0,3} \cup \beta_{1,3}$, b' and r' cross. Now, let $b' \in B'$ and $r' \in R'$ such that b' and r' cross. We will show that $b' \in B_I$ and $r' \in R_I \setminus \rho_{0,3} \cup \beta_{1,3}$.

   If $b' \in B$ and $r' \in R$ then $b' \in B_I$ and $r' \in R_I \cap R' = R_I \setminus \rho_{0,3} \cup \beta_{1,3}$. If $b' \in B$ and $r' \notin R$ then $r' \in R' \setminus R = \beta_{\{1,3\}^2}$. Now, since r' crosses some blue chord, it must be that $r' \in \beta_{1,3}$. Now, $b' \in B_I$; otherwise, $b' \in B \setminus B_I$ and $r' \in B \setminus B_I$ form an X-configuration in B that is ruled out by Case (ii). If $b' \notin B$ then $b' \in B' \setminus B = \rho_{\{0,3\}^2}$. Since b' crosses $r' \in R'$, $b' \in \rho_{0,3} \subseteq R_I$ and $r' \in R$. If $r' \in R_I$ then Case (i) forbids this configuration. Thus, we may suppose that $r' \notin R_I$ and therefore, $r' \in \rho'_{0,0}$ or $r' \in \rho'_{3,3}$. Suppose, without loss of generality, that $r' \in \rho_{i,j} \subseteq \rho'_{0,0}$. Now, $i \neq 0$, $j \neq 0$, and $i \neq j$, since $\rho_{0,0} \cap R' = \phi$. Then there must be two blue zones, β_i and β_j such that the zones under consideration appear in clockwise order: β_0, ρ_i, β_i, ρ_0, β_j, ρ_j, β_1. Let $\beta'_{i,j}$ be the arc $(\mathrm{last}(\rho_\mathrm{i}), \mathrm{first}(\rho_\mathrm{j}))$ that includes β_i, ρ_0, and β_j as subintervals. Now, $\beta'_{i,j}$ cannot contain exactly one endpoint of any blue chord, since r' is not an interface chord. Therefore, by Lemma 3.1, $\beta'_{i,j}$ must contain both endpoints of a blue interface chord. But such a chord cannot cross the red interface chord that we know exists in $\rho_{1,2}$. Thus, we have a contradiction.

This completes the proof of the claim. □

The two splits $\{B', R'\}$ and $\{B, R\}$ cross each other, contradicting the goodness of the split $\{B, R\}$. Hence, Case (iii) is impossible.

This completes the proof of Lemma 3.3.

■

Lemma 3.4: $||$-configurations are impossible.
Proof:
Again, to derive a contradiction suppose that ℓ' and ℓ'' induce a $||$-configuration, where $\{\ell', \ell''\} \subseteq B_I$. Let $\ell' \in \beta_{0,3}$ and $\ell'' \in \beta_{1,2}$, as in Figure 3.3. Then there exist at least 4 red zones $\{\rho_i | \ i = 0, \cdots, 3\}$ such that $(\beta_i, \rho_i | \ i = 0, \cdots, 3)$ appear in a circular listing of D.

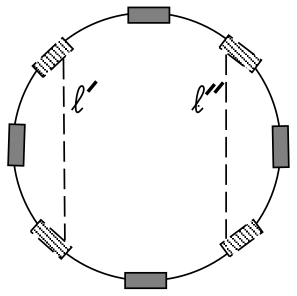


Figure 3.3 A $||$-configuration

We observe that all of the following must hold:
(a) ($R \cap \rho'_{0,1} = \phi$ and $R \cap \rho'_{1,2} = \phi$) or ($\ell' \notin B_I$ and $\ell'' \in B_I$),
(b) ($R \cap \rho'_{2,3} = \phi$ and $R \cap \rho'_{3,0} = \phi$) or ($\ell' \in B_I$ and $\ell'' \notin B_I$),
(c) $R \cap \rho'_{1,3} = \phi$ or ($\ell' \in B_I$ and $\ell'' \in B_I$).

As in Lemma 3.3, we consider all cases.

Case (i): $\ell', \ell'' \in B_I$.
For $0 \leq i \leq 3$, each of the four sets $R \cap \rho'_{i,i+1}$ (modulo 4) is empty, by (a) and (b). As in Lemma 3.3, Lemma 3.1 implies that the set $\{\rho_i| \; i = 0, \cdots, 3\}$ can be selected so that $|\rho_{0,2}|$, $|\rho_{1,3}| \geq 1$. But then any chord of $\rho_{0,2}$ and any chord of $\rho_{1,3}$ form an X-configuration, which is impossible by Lemma 3.3. Thus, Case (i) cannot occur.

Case (ii): neither ℓ' nor ℓ'' is in B_I.
Then, from (a), (b), and (c) above, and by Lemma 3.1, we find a red interface chord in each of $\rho'_{1,1}$ and $\rho'_{3,3}$. Therefore, all blue interface chords must cross both ℓ' and ℓ'', forming X-configurations, contradicting Lemma 3.3. Thus, Case (ii) is impossible.

Case (iii): exactly one of ℓ', ℓ'' is in B_I.
Suppose without loss of generality that $\ell' \in B_I$ and $\ell'' \notin B_I$. Then $R \cap \rho'_{0,1} = \phi$, $R \cap \rho'_{1,2} = \phi$, and $R \cap \rho'_{1,3} = \phi$, by (a) and (c). Now, Lemma 3.1 implies the existence of a red interface chord in $\rho'_{1,1}$. But such a chord does not cross $\ell' \in B_I$, a contradiction. Therefore, Case (iii) cannot occur.

This completes the proof of Lemma 3.4 and the proof of Theorem 3.1.

■

■

4. Basic Operations on Circle Diagrams

Let T be the standard $*$-decomposition tree of the connected circle graph G. Three types of geometric operations on circle diagrams of T are now introduced. First, consider the circle diagram in Figure 4.1(a), where the shaded area A corresponds to a set of chords, and the dotted line (p, p') corresponds to a marker vertex associated with some hypothetical split of V. In this diagram, the area A can be rotated in two basic ways, while keeping the two points p and p' fixed on the page. Namely, after a WE-rotation we obtain Figure 4.1(b), and after a NS-rotation we obtain Figure 4.1(c).

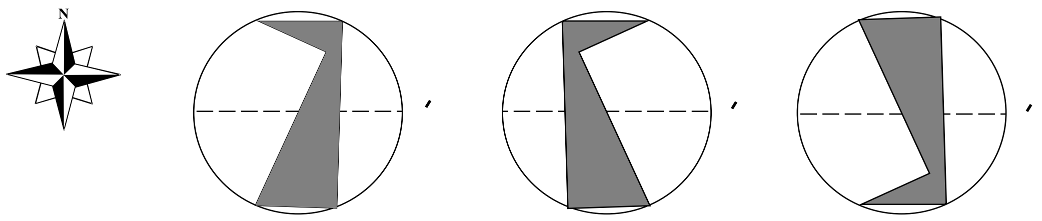


Figure 4.1 Rotations of polygon diagrams

For a circle diagram D_x representing a clique node x, a permutation of D_x is any diagram obtained from arbitrarily permuting its chords. Similarly, if D_x represents a star then a permutation is any diagram obtained by permuting its leaves.

In addition to the above geometric operations, we need the following concepts. Let D be a circle diagram of G. A partial polygon diagram of G can be obtained by adding a set K of corners to the circumference of the circle. In the resulting diagram (D, K) each section of the circle between two consecutive corners corresponds to a side of a polygon. The resulting partial diagram constitutes a (complete) polygon representation of G if no side of (D, K) contains the two endpoints of one chord. As such, permutation graphs are exactly those circle graphs which have a circle diagram to which the addition of just two corners results in a diagram with the endpoints of each chord on different sides. Note that a circle diagram with two corners has only two sides and thus does not form a polygon; nevertheless, we refer to this degenerate case as a 2-polygon.

A kernel of a circle diagram D is a minimum cardinality set of corners K such that the diagram (D, K) is a polygon representation of G. Recall that each line segment ℓ of D splits the circle into two closed regions called the ℓ-semicircles of D (cf. Section 2). We say that ℓ is deficient in D if at least one of the ℓ-semicircles does not contain a corner. In the same vein, define the deficiency of a diagram (D, K), denoted $\delta(D, K)$, to be the minimum number of corners that must be added to obtain a complete polygon representation.

Next, let (x, y) be an edge incident to a leaf node y in the standard $*$-decomposition tree T of G, and consider the neighbours of the vertex corresponding to the marker ℓ_{xy} in G_y. We say that y is active if there exists at least one vertex not adjacent to ℓ_{xy} in G_y. That is, there exists at least one chord not intersecting ℓ_{xy} in any diagram D_y of G_y. This particular chord forces any recognition algorithm to add a corner, regardless of the structure of other nodes in T.

Roughly speaking, the recognition algorithm processes each node x by computing a partial diagram $(D_{T(x)}, K_{T(x)})$ of $G_{T(x)}$, where $K_{T(x)}$ is a minimum set of *required* corners. The computed diagram $(D_{T(x)}, K_{T(x)})$ may not be a complete polygon diagram (i.e. more corners may be added), nevertheless, it provides a basis for solving the problem. Applying the same argument to the complementary graph $G_{T \setminus T(x)}$, we obtain a second partial diagram $(D_{T \setminus T(x)}, K_{T \setminus T(x)})$. A straightforward but useful observation then is: the deficiency of one diagram, say $D_{T(x)}$, can be reduced by the corners $K_{T \setminus T(x)}$ of the other diagram in any partial polygon diagram $(D_{T(x)} * D_{T \setminus T(x)}, K_{T(x)} \bigcup K_{T \setminus T(x)})$ of G.

Now, suppose that T is rooted at an arbitrary vertex r. Then the above complementary support situation may occur if $x \neq r$. In this case, let w be x's parent in T, and let $\ell_{wx} \in D_x$ be the marker corresponding to the edge (w, x) of T. Here, we use the "conditional" optimality definition given below to capture any possible support to $(D_{T(x)}, K_{T(x)})$ from $(D_{T \setminus T(x)}, K_{T \setminus T(x)})$. The definition is formalized with the help of two (imaginary) corners α and α', placed immediately next to the two endpoints of ℓ_{wx} in $D_{T(x)}$, such that the hypothetical chord (α, α') intersects exactly the set $\ell_{wx} \bigcup N(\ell_{wx})$, where $N(\ell_{wx})$ denotes the set of chords that intersect ℓ_{wx}. More specifically, we define 1-optimal diagrams as follows:

1. Call a diagram $(D_{T(x)}, K_{T(x)})$ 1-supported (relative to the distinguished marker ℓ_{wx}) if $(D_{T(x)}, K_{T(x)} \cup \{\alpha, \alpha'\})$ is a complete polygon representation of $G_{T(x)}$.

2. If $(D_{T(x)}, K_{T(x)})$ is a 1-supported diagram with a smallest possible set of corners $K_{T(x)}$, then $K_{T(x)}$ is called a 1-supported kernel. Note that, $\delta(D_{T(x)}, K_{T(x)}) \leq |\{\alpha, \alpha'\}| = 2$.

3. If $K_{T(x)}$ is a 1-supported kernel such that $\delta(D_{T(x)}, K_{T(x)})$ is minimum among all possible diagrams using 1-supported kernels then $(D_{T(x)}, K_{T(x)})$ is called a 1-optimal diagram.

As an example, Figure 5.1(a) illustrates a circle diagram of a node y whose parent is denoted x. A 1-optimal diagram for y (relative to the distinguished marker ℓ_{xy}) has a 1-supported kernel of size 2, and deficiency equals 2.

In the special case where $x = r$, we have $T(x) = T$, and there is no complementary support to account for. For uniformity, however, we call a (complete) polygon diagram of G a 0-supported diagram. Likewise, we call a kernel of G a 0-supported kernel, and call an optimal solution of G a 0-optimal solution. Using the above convention, and the following simple 0/1 indicator

$$\eta(x) = \begin{cases} 0 & \text{if } x = r \text{ (the root)} \\ 1 & \text{otherwise} \end{cases}$$

we can refer to $\eta(x)$-supported and $\eta(x)$-optimal diagrams, for any node x.

5. The Main Algorithm

We now show a simple upper bound on recognizing k-polygon graphs, for any fixed k.

Theorem 5.1: Given a standard $*$-decomposition tree T of a circle graph $G = (V, E)$, $|V| = n$, and an integer k, there exists an $O(4^k n^2)$ algorithm for deciding whether G is a k-polygon graph, and producing a corresponding polygon diagram, if one exists.

In proving the above theorem we may assume that all degree-1 nodes of T are active. If not, then pruning any inactive leaf from T results in a smaller problem instance that has the same kernel size. Moreover, any polygon diagram for the reduced problem can be easily extended to one for the original problem. The above pruning step can be repeatedly applied until the remaining tree satisfies the above assumption. This process can be done in $O(n)$ time, and hence, we may proceed further with the above assumption in mind.

In the recognition algorithm, described below, an arbitrary node r is designated as the root node. Nodes of T are then stored in a stack $\mathcal{S}$ in *postorder* ; that is, if y is a descendant of x then y appears somewhere on top of x in $\mathcal{S}$. Processing of T proceeds next from the leaves to the root r, according to the ordering in $\mathcal{S}$. Each node is processed by computing a $\eta(x)$-optimal diagram $(D_{T(x)}, K_{T(x)})$ for the graph $G_{T(x)}$. To maintain a polynomial running time, the algorithm terminates with failure if $|K_{T(x)}| > k$, for any node x. Otherwise, it returns an optimal solution $(D_{T(r)}, K_{T(r)})$.

Processing a node x is carried out by executing one of three merge functions, depending on x's type. Each function takes as input pointers to a target node x, and the set Y of x's children, and returns an optimal diagram for $G_{T(x)}$. These elements are combined in the following function:

```
function main ($T, r, k$) {
#     input:   a standard $*$-decomposition tree $T$ of $G$ in which each degree-1
#              node is active, a root node $r$, and an integer $k$
#     output:  a polygon diagram $(D_{T(r)}, K_{T(r)})$ of $G$ with the smallest possible
#              number of sides, if $|K_{T(r)}| \le k$
    $\mathcal{S}$ = the nodes of $T$ stored in postorder              #$r$ is at the bottom of $\mathcal{S}$
    while ($\mathcal{S}$ is not empty) {
        $x$ = pop($\mathcal{S}$)
        #let $Y$ be the set of $x$'s children
        $(D_{T(x)}, K_{T(x)}) = \begin{cases} \texttt{primeMerge(x,Y)} & \text{if } x \text{ is a prime node} \\ \texttt{cliqueMerge(x,Y)} & \text{if } x \text{ is a clique node} \\ \texttt{starMerge(x,Y)} & \text{otherwise} \end{cases}$
        if ($|K_{T(x)}| > k$) { return(nil,$\phi$) }
    }
    return $(D_{T(r)}, K_{T(r)})$
}
```

To prove the main theorem it suffices to show that

Theorem 5.2: Function `main()` computes an $\eta(x)$-optimal diagram $(D_{T(x)}, K_{T(x)})$ for any node x in $O(4^k n^2_{T(x)})$ time.

The proof is broken into lemmas 5.1-5.3 below. First, we show in Lemma 5.1 the sufficiency of using 1-optimal diagrams $\{(D_{T(y)}, K_{T(y)})|\ y \in Y\}$ to compute that of $G_{T(x)}$. To shorten and simplify the notation, we abbreviate any partial polygon diagram, say $(D_{T(x)}, K_{T(x)})$, with $P_{T(x)}$, and let $\delta P_{T(x)} = \delta(D_{T(x)}, K_{T(x)})$. Furthermore, we use $D_x * \sum_{y\in Y} D_{T(y)}$ to denote the $*$-composition of D_x with all diagrams in $\{D_{T(y)}|\ y \in Y\}$.

The main strategy is to show that if $P'_{T(x)}$ is a $\eta(x)$-optimal diagram of $G_{T(x)}$ then one can factor the corners of $K'_{T(x)}$ into disjoint subsets, denoted $\{K'_{T(y)}|\ y \in Y\}$ and K_{new}, such that for any arbitrary node $y \in Y$, replacing $K'_{T(y)}$ by a set of corners $K_{T(y)}$ of a 1-optimal diagram $P_{T(y)}$ (as computed by function `main()`) results in an <u>extendible</u> diagram; that is, one that can be extended to an $\eta(x)$-optimal diagram for $G_{T(x)}$ by possibly adding some new corners. More specifically, we factor $P'_{T(x)}$ such that:

$$\begin{aligned} D'_{T(x)} &= D_x * \sum_{y_i \in Y} D'_{T(y_i)}, \text{ and} \\ K'_{T(x)} &= K_{new} \bigcup_{y_i \in Y} K'_{T(y_i)}, \end{aligned}$$

where D_x is a circle diagram of G_x, and for each $y_i \in Y$, $D'_{T(y_i)}$ is a circle diagram of $G_{T(y_i)}$ obtained by adding a suitable marker ℓ_{x,y_i}. Factoring $K'_{T(x)}$ is done as follows: first, assume that chords in each diagram $D'_{T(y_i)}$ are assigned a unique colour. By Theorem 3.1, each colour induces exactly two zones in any possible diagram of $G_{T(x)}$. We next assign a corner of $K'_{T(x)}$ to the set $K'_{T(y_i)}$ if it lies inside a zone of colour $D'_{T(y_i)}$ (hence, each such corner lies on an arc between two monocoloured endpoints). The remaining corners, if any, are assigned to K_{new}. A simple but important fact then is: for each $y_i \in Y$, $P'_{T(y_i)} = (D'_{T(y_i)}, K'_{T(y_i)})$ is a 1-supported (but not necessarily 1-optimal) diagram of $G_{T(y_i)}$.

Lemma 5.1: Let $P'_{T(x)} = (D'_{T(x)}, K'_{T(x)})$ be an $\eta(x)$-optimal diagram for $G_{T(x)}$ factored as above. In addition, let y be any arbitrary selected node in Y, for which function `main()` has computed a 1-optimal diagram $P_{T(y)} = (D_{T(y)}, K_{T(y)})$. Then the new diagram $P_{T(x)}$ defined by

$$\begin{aligned} D_{T(x)} &= D_x \; * D_{T(y)} * \sum_{y_i \in Y \setminus \{y\}} D'_{T(y_i)}, \\ K_{T(x)} &= K_{new} \bigcup K_{T(y)} \bigcup_{y_i \in Y \setminus \{y\}} K'_{T(y_i)} \end{aligned}$$

is extendible.

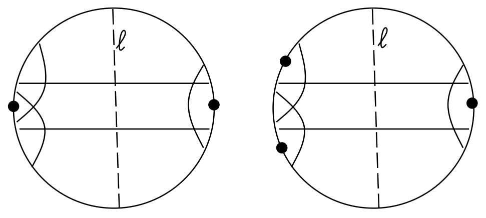
Proof: We consider the following disjoint cases:
Case 1: $\delta P_{T(y)} \leq \delta P'_{T(y)}$. Since $P_{T(y)}$ is 1-optimal, it follows that $|K_{T(y)}| \leq |K'_{T(y)}|$. So, the new diagram $P_{T(x)}$ (with $P_{T(y)}$ oriented so that if it is deficient on a side then $P'_{T(y)}$ is deficient on that side in $P'_{T(x)}$) certainly satisfies $|K_{T(x)}| \leq |K'_{T(x)}|$, and $\delta P_{T(x)} \leq \delta P'_{T(x)}$. Hence, $P_{T(x)}$ is a $\eta(x)$-optimal diagram.

Case 2: $\delta P_{T(y)} > \delta P'_{T(y)}$. By assumption, $P_{T(y)}$ is 1-optimal, hence, it must be the case that $|K_{T(y)}| < |K'_{T(y)}|$ (that is, equality does not hold, otherwise $P_{T(y)}$ violates optimality in that $\delta P_{T(y)}$ is not the smallest possible). Let

$$\begin{aligned} \delta_{diff} &= \delta P_{T(y)} - \delta P'_{T(y)} \quad (> 0), \text{ and} \\ k_{diff} &= |K'_{T(y)}| - |K_{T(y)}| \;\; (> 0). \end{aligned}$$

Note that $\delta_{diff} \leq \delta P_{T(y)} \leq 2$. The claim follows easily if $\delta_{diff} \leq k_{diff}$ since the 1-optimal diagram $P_{T(y)}$ can be augmented with δ_{diff} new corners to reduce its deficiency to $\delta P'_{T(y)}$, while keeping the total number of corners in the augmented diagram $\leq |K'_{T(y)}|$. The above augmentation step can also take place in $P_{T(x)}$ (instead of $P_{T(y)}$), and hence $P_{T(x)}$ is extendible.

The remaining case where $2 \geq \delta_{diff} > k_{diff}$ occurs when: $\delta P_{T(y)} = 2$, $\delta P'_{T(y)} = 0$, and $k_{diff} = 1$. Here, it is sometimes impossible to nullify $\delta P_{T(y)}$ by just adding a new corner, as illustrated by the example of Figure 5.1. We therefore consider the effect of $P_{T(y)}$ in the target diagram $P_{T(x)}$. To this end, define the restriction of $P_{T(x)}$ to $P_{T(y)}$ to be the subdiagram of $P_{T(x)}$ obtained by keeping all corners in $K_{T(x)}$, and deleting all chords not in $P_{T(y)}$.



(a) $P_{T(y)}$ (b) $P'_{T(y)}$

Figure 5.1 An example where $\delta P_{T(y)} = 2$, $\delta P'_{T(y)} = 0$, and $k_{diff} = 1$.

We now distinguish the following cases:

Case 2.A: $P_{T(y)}$ receives complementary support from $P_{T(x)}$. That is, the restriction of $P_{T(x)}$ to $P_{T(y)}$ contains a corner not in $K_{T(y)}$. Then one can add a new corner to $P_{T(x)}$ so as to nullify the deficiencies of all chords of $P_{T(y)}$ in the modified diagram of $P_{T(x)}$. Hence, $P_{T(x)}$ is extendible.

Case 2.B: Else, $P_{T(y)}$ receives complementary support only from some ancestor of x in T. This implies the following properties of node x:

P1. $x \neq r$, and by definition $\eta(x) = 1$. So, let w denote x's parent in T.

P2. $K_{new} = \phi$ (recall that K_{new} contributes to $P_{T(y)}$'s support). Hence, in the diagram D_x any chord ℓ, $\ell_{wx} \neq \ell \neq \ell_{xy}$, intersects at least one of the two markers ℓ_{wx} or ℓ_{xy}.

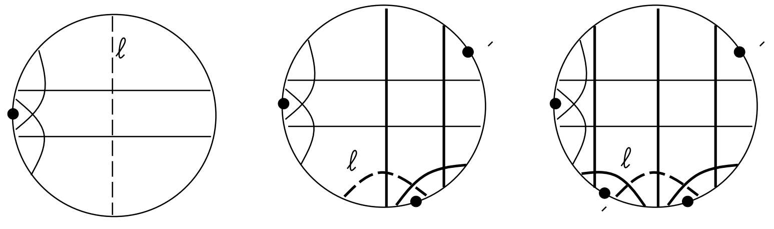
P3. $Y = \{y\}$ (since all possible children of x are active, and any active child contributes at least one corner). That is, $P_{T(x)}$ is simply $((D_x {*} D_{T(y)}), K_{T(y)})$.

We next distinguish the following subcases:

Case 2.B.1: ℓ_{wx} does not intersect ℓ_{xy} in D_x (see Figure 5.2(b) and (c) for two examples). To show that $P_{T(x)}$ is extendible, it suffices to show that a new corner can be added to it so as to make the modified diagram 1-supported with deficiency $\leq \delta P'_{T(x)}$. To this end, recall the following facts about $\delta P'_{T(x)}$:

1. $\delta P'_{T(x)} \geq 1$: to see this, recall that property **P2** implies that exactly one of the ℓ_{wx}-semicircles of D_x does not contain the two endpoints of any chord. Consequently, ℓ_{wx} appears deficient in any 1-optimal diagram of $D_{T(x)}$.
2. By assumption, $P'_{T(x)}$ is 1-optimal. Hence, its deficiency can be nullified by adding one or two corners from a set $\{\alpha, \alpha'\}$, where α and α' are placed immedi-

ately next to the two endpoints of the marker ℓ_{wx} so that the hypothetical chord (α, α') intersects ℓ_{wx}.



(a) $P_{T(y)}$ (b) $P_{T(x)}$ (c) another $P_{T(x)}$

Figure 5.2 Examples of case 2.B.1

To show the above sufficient condition, let c be any corner in $P_{T(y)}$ (c exists since all leaves of $T(y)$ are active), and note that one can add a new corner c' to $P_{T(x)}$ such that

1. the hypothetical chord (c, c') crosses all chords that intersect ℓ_{xy} in D_x, and some of the interface chords intersected by ℓ_{xy} in $P_{T(y)}$, and
2. if α nullifies $\delta P'_{T(x)}$ then the hypothetical chord (c, α) crosses the remaining interface chords intersected by ℓ_{xy} in $P_{T(y)}$, and the marker ℓ_{wx} (as in Figure 5.2b),
3. else it must be the case that $\{\alpha, \alpha'\}$ nullifies $\delta P'_{T(x)}$. Then the hypothetical chords (c, α) and (c, α') cross the remaining interface chords intersected by ℓ_{xy} in $P_{T(y)}$, and the marker ℓ_{wx} (as in Figure 5.2c).

Thus, $P_{T(x)}$ with the added corner c' is 1-supported with deficiency $\leq \delta P'_{T(x)}$. This proves the sufficient condition.

Case 2.B.2: Else, ℓ_{wx} intersects ℓ_{xy} in D_x. For convenience, we say that the diagram $P_{T(y)}$ (or $P'_{T(y)}$) is of <u>type $(a|a)$</u>, abbreviated $P_{T(y)} \in (a|a)$, if each of the ℓ_{xy}-semicircles of $P_{T(y)}$ contains at least one corner. On the other hand, if exactly one of the two semicircles is corner-free then we write $P_{T(y)} \in (a|-)$. Note that $P'_{T(y)} \in (a|a)$, since $\delta P'_{T(y)} = 0$. We now deal with the following cases.

Cases 2.B.2$'$: $P_{T(y)} \in (a|a)$. Then $P_{T(x)}$ is a 1-supported diagram with a smaller kernel than $P'_{T(x)}$, contradicting the assumption that $P'_{T(x)}$ is 1-optimal.

Cases 2.B.2$''$: $P_{T(y)} \in (a|-)$. Then the following distribution of the $|K'_{T(y)}| = |K_{T(y)}| + 1$ corners in $P'_{T(y)} \in (a|a)$ must hold true:

1. Exactly $|K_{T(y)}|$ corners appear in one of the ℓ_{xy}-semicircles of $P'_{T(y)}$. This follows since $P'_{T(y)}$ is a 1-supported diagram (but not necessarily 1-optimal, as described in factoring $K'_{T(x)}$), and in addition, all of the $|K_{T(y)}|$ corners are required (since $P_{T(y)}$ is 1-optimal).
2. Exactly one non-required corner exists in the other semicircle of $P'_{T(y)}$. That is, this particular semicircle does not contain the two endpoints of any chord $\ell \neq \ell_{xy}$. Call this corner c.

Deleting c from $P'_{T(y)}$ results in a 1-supported diagram with $|K_{T(y)}|$ corners and deficiency = 1, contradicting the assumption that $P_{T(y)}$ is 1-optimal (since $\delta P_{T(y)} = 2$).

The above exhausts all possible cases, and completes the proof of Lemma 5.1 ■

Second, we show the sufficiency of using the geometric operations, discussed in the last section, to generate 0/1-optimal diagrams:

Lemma 5.2: There exists an assignment of orientations to 1-optimal diagrams in $\{D_{T(y)}|\ y \in Y\}$, and permutation of chords in D_x (if x is a clique or a star node), such that $D_x * \sum_{y \in Y} D_{T(y)}$ is extendible to an $\eta(x)$-optimal diagram by possibly adding some new corners.
Proof:
This follows easily from Lemmas 3.1 and 5.1 above, by taking into account all possible degrees of freedom for the markers in D_x and the 1-optimal diagrams of the nodes in Y.
■

Third, we provide performance guarantees for the functions called in the while-loop:

Lemma 5.3: Given a set $\{(D_{T(y)}, K_{T(y)})|\ y \in Y\}$ of precomputed 1-optimal diagrams, then the above merge functions compute an $\eta(x)$-optimal diagram of $G_{T(x)}$ in

$$t \in \begin{cases} O(n^2_{T(x)}) & \text{if } x \text{ is a prime leaf node} \\ O(4^k n_{T(x)}) & \text{if } x \text{ is an internal prime node} \\ O(n_{T(x)}) & \text{if } x \text{ is a clique node or a star node} \end{cases}$$

where $n_{T(x)}$ is the number of nodes in $G_{T(x)}$.
Proof:
See the details in the next section. ■

To complete the proof of Theorem 5.2 above, we note that:
The timing mentioned in Lemma 5.3 assumes that 1-optimal diagrams for nodes in Y have been precomputed. Summing over all nodes in $T(x)$, it is easy to verify that the total

time required to process all prime leaves in $T(x)$ is $O(n_{T(x)}^2)$, and the time required to process the remaining nodes in $T(x)$ is $O(4^k n_{T(x)}^2)$. Adding the two terms together gives the stated bound. This completes the proof. ■

6. Optimal Diagrams

We now focus on computing a 0/1-optimal diagram for any arbitrary node x in the pruned tree T. The following few notations will be used throughout the section. Let r denote the root node of T, w denote x's parent (if $x \neq r$), and Y denote the set of x's possible children. In addition, let n_x and $n_{T(x)}$ denote the number of nodes in G_x and $G_{T(x)}$, respectively.

It is also worthwhile recalling the following facts. First, the algorithm considers processing a node x only after computing a set of 1-optimal diagrams $\{D_{T(y)} | \ y \in Y\}$. Second, in the pruned tree T, every child $y \in Y$ is active with respect to the marker ℓ_{xy}. Third, the definition of a 1-optimal diagram of $G_{T(x)}$ uses two hypothetical corners α and α', placed immediately next to the two endpoints of the distinguished marker ℓ_{wx}, such that the hypothetical chord (α, α') intersects $\ell_{wx} \bigcup N(\ell_{wx})$ (cf. Section 4), where $N(\ell_{wx})$ is the set of chords intersecting ℓ_{wx}.

6.1 Function Fill-In

A core function that will be used subsequently calls for solving the following restricted-corners assignment problem: given a partial polygon diagram (D, K), where K is a required set of corners, find a complete polygon diagram $(D, K \cup K_{new})$, such that K_{new} is as small as possible. That is, K_{new} equals the deficiency $\delta(D, K)$. In the context of evaluating a node x with a set Y of children, the diagram D stands for any arbitrary diagram $D_x * \sum_{y \in Y} D_{T(y)}$ obtained by permuting and orienting the markers in D_x. The required set K stands for the set $\bigcup_{y \in Y} K_{T(y)}$ of precomputed 1-supported kernels.

In this context, the problem admits a simple $O(1)$ time solution if x is a leaf node corresponding to a clique or a star. We now devise a solution when x is an internal node of T; here $K \neq \phi$ and there exists a corner β at which one can start traversing the diagram.

```
function fillIn $(D, K)$ {
#     input:     a partial polygon diagram $(D, K)$
#     output:    a set $K_{new}$ of new corners such that $(D, K \cup K_{new})$ is a
#                complete polygon diagram, and $K_{new}$ is as small as possible
#     notation:  let $Q$ be the set of endpoints of chords in $D$
    $K_{new} = \phi$
    $\beta$= any corner in $K$
    for ($e$ in $Q$) { $side(e) = nil$ }
    $currentSide = 1$
    for ($e$ in $Q \cup K$) {    #$e$ is encountered in a clockwise traversal of $D$
                                 #beginning just after the selected corner $\beta$ and
                                 #ending just before $\beta$
        if ($e$ in $Q$) {
            let $e'$ be the other endpoint of the chord with endpoint $e$
            if ($side(e') == currentSide$) {
                $currentSide = currentSide + 1$
                $K_{new} = K_{new} \cup$ { a corner just counterclockwise of $e$ }
            }
            $side(e) = currentSide$
        }
        elseif ($e$ in $K$) { $currentSide = currentSide + 1$ }
    }
    return ($K_{new}$)
}
```

The above algorithm requires $O(n_{T(x)})$ time to process the graph $G_{T(x)}$ corresponding to the internal node x. Correctness of the above function is straightforward and deserves no further comment.

The remaining case occurs if x is a leaf node corresponding to a prime graph G_x. Here, $K = \phi$ and there is no corner β at which to start the above process. Nevertheless, one can find a chord ℓ that splits D_x into two semicircles $\{W, \overline{W}\}$ such that no other chord lies completely in W. Clearly, there exists a solution in which W contains exactly one corner, denoted β hereafter. If W contains n' endpoints (where $n' \leq n_x - 2$, if x is a leaf that does not correspond to a clique or a star) then β can be placed in any of $n' + 1$ possible arcs in W. Function `fillIn()` can then be executed $n' + 1$ times to compute $K_{new} = \min \{$`fillIn(D,`$\{\beta\}$`)`$|\ \beta$ in $W\}$. The set $K_{new} \cup \{\beta\}$ is a solution to `fillIn(`D, ϕ`)`. This special case requires $O(n_x^2)$ to process a prime leaf node x.

6.2 Prime Nodes

We now consider computing a 0/1-optimal diagram for a node x that corresponds to a prime graph G_x. This can be achieved using a brute-force algorithm that examines all possible orientations of 1-optimal diagrams in $\{D_{T(y)}|\ y \in Y\}$, as described below.

```
function primeMerge(x,Y) {
```

```
#     input:     $x$, $Y$, $D_x$, and $\{D_{T(y)} | y \in Y\}$, as described above
#     output:    a $\eta(x)$-optimal diagram $(D, K)$ of $G_{T(x)}$, and
#                a minimum fill-in $F$; that is, $|F| = \delta(D, K)$
#     notation:  $D$ and $K$ stand for $D_{T(x)}$, $K_{T(x)}$, respectively
# step 1:  initialize the set of all external-support corners
        $K_{support} = \begin{cases} \{\alpha, \alpha'\} & \text{if } x \neq r \text{ (the root)} \\ \phi & \text{otherwise} \end{cases}$
# step 2:  initialize $D$, $|K|$ and $|F|$ to some out-of-range values
    $D = nil$;  $|K| = |F| = \infty$
# step 3:  search for a 0/1-optimal solution
    for ($K_{sup} \subseteq K_{support}$) {                                   #iterate $\leq 4$ times
       for (every possible orientation of orientable diagrams in $\{D_{T(y)} | y \in Y\}$) {
          $D_{tmp} = D_x * \sum_{y \in Y} D_{T(y)}$
          $K_{tmp} = K_{sup} \bigcup_{y \in Y} K_{T(y)}$
          $K_{new}$ = fillIn ($D_{tmp}, K_{tmp}$)
          $K' = (K_{tmp} \cup K_{new}) \setminus K_{sup}$                  #now, $(D_{tmp}, K')$ is a partial
          $F' = K_{sup}$                                            #diagram of deficiency $|F'|$
          if (($|K'| < |K|$) or ($|K'| == |K|$ and $|F'| < |F|$))
             { $D = D_{tmp}$;  $K = K'$;  $F = F'$ }
       }
    }

      return $\begin{cases} (D, K, F) & \text{if } x \neq r \\ (D, K \cup F, \phi) & \text{otherwise} \end{cases}$    #return a 1-optimal diagram / #return a 0-optimal diagram
}
```

Besides its role in processing prime nodes, function `primeMerge()` will be called by `cliqueMerge()` and `starMerge()` in a context where a small distinguished subset of nodes in Y will be marked *orientable* (for the rotations done in the for-loop in step 3); the remaining nodes will be treated as *fixed* points. In this respect, function `primeMerge()` will act as a "rotation manager" for the other two functions.

Timing: Suppose that `primeMerge()` is called with a subset $Y' \subseteq Y$ of orientable nodes ($Y' = Y$ if x is a prime node), then function `fillIn()` will be called $4 \times 4^{|Y'|}$ times. Several straightforward observations can be used to speed-up the above function. For instance, if $D_{T(y)}$, $y \in Y$, is a 1-optimal diagram that has at least one corner in each of its two possible ℓ_{xy}-semicircles then y need not be subjected to a NS-rotation. Likewise, if $\delta(D_{T(y)}, K_{T(y)}) = 0$ then $D_{T(y)}$ need not be subjected to a WE-rotation when processing x. Nevertheless, the worst case running time remains the same, and hence no further effort will be expended on improving the algorithm.

6.3 Clique Nodes

The underlying problem in processing a clique node x lies in considering all possible permutations and orientations of the markers in D_x. Fortunately, it is possible to take advantage of the symmetry of D_x to quickly identify an optimal permutation for each problem instance, as will be shown next.

To introduce the algorithm, however, we need to examine a new level of detail in polygon diagrams. Specifically, for a child node $y \in Y$, let I_{xy} be the set of interface chords in the precomputed 1-optimal diagram $(D_{T(y)}, K_{T(y)})$; that is the set of chords that intersect the marker ℓ_{xy}. In addition, let W be any one of the two semicircles induced by the marker ℓ_{xy} in $D_{T(y)}$, and let $WK_{T(y)} \subseteq K_{T(y)}$ be the subset of corners that lie in W. We focus on the distribution of $WK_{T(y)}$ around the interface chords I_{xy} in the diagram. To this end, we define the *semicircle* deficiency of I_{xy} in W, denoted $\delta(D_{T(y)}, WK_{T(y)}, I_{xy})$, to be the deficiency of I_{xy} in the partial diagram $(D_{T(y)}, WK_{T(y)})$, assuming that all corners not in $WK_{T(y)}$ have been erased.

Furthermore, we say that W is a *big-end* of $D_{T(y)}$ (or simply, a big-end of the node y) if $WK_{T(y)} \neq \phi$ and $\delta(D_{T(y)}, WK_{T(y)}, I_{xy}) \leq \delta(D_{T(y)}, \overline{W}K_{T(y)}, I_{xy})$, where $\overline{W}K_{T(y)}$ is the subset of corners in the other semicircle. We now introduce a (redundant) notation that may help in a better visualization of the distribution of corners in y's big-end. Namely, we say that the diagram $D_{T(y)}$ (or, the node y itself) is of type $[1,1]$, $[0,1]$, or $[0,0]$ if the semicircle deficiency of y's big-end is 0, 1, or 2, respectively. Figure 6.1 gives an example layout for each of the three types mentioned above.

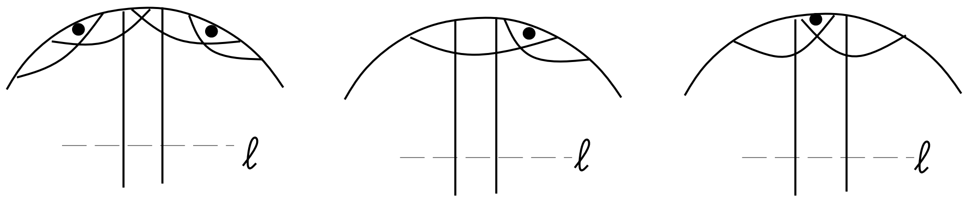


Figure 6.1 Examples of semicircle deficiences of I_{xy}

Step 1 of function `cliqueMerge(x,Y)` operates on (at most) three key elements of Y (equivalently, markers in D_x), denoted y_0, y_1, and y_2. To introduce the definitions, we need the following concept: for any subset $Y' \subseteq Y$, call an element $y \in Y'$ a *leader* of Y' if y's big-end has the smallest possible semicircle deficiency among all nodes in Y'. The first key element y_0 is chosen to be a leader of Y. In addition, if $|Y| \geq 2$ then let y_1 be a leader of

$Y \setminus \{y_0\}$, else $y_1 = nil$. Similarly, if $|Y| \geq 3$ then y_2 is a leader of $Y \setminus \{y_0, y_1\}$, else $y_2 = nil$. In Figure 6.2, the big-end of each key element is marked with a filled circle. To present the algorithm, we start with the case where $x \neq r$ (hence, $\eta(x) = 1$).

```
function cliqueMerge(x,Y) {
#      input:    $x$, $Y$, $D_x$, and $\{D_{T(y)} | y \in Y\}$, as described above
#      output:  a 1-optimal diagram $(D,K)$ of $G_{T(x)}$, and
                 a minimum fill-in $F$; that is, $|F| = \delta(D,K)$
# step 1:  identify an extendible diagram of $G_x$
    if ($|Y| == 1$) { $D_x$ = any possible diagram of $G_x$ }
    elseif ($|Y| == 2$) {
        $D_x$ = a diagram where $y_0$ and $y_1$ are placed next
             to each other as in Figure 6.2(a)
    }
    elseif ($|Y| \geq 3$) {
        $D_x$ = a diagram where the markers corresponding to
             $y_0$, $y_1$, and $y_2$ are ordered as in Figure 6.2(b),
             and all other chords are ordered arbitrarily
    }
# step 2:  if $|Y| \neq 2$ mark all nodes in $Y$ *fixed*; else if $|Y| == 2$ mark $y_0$ and $y_1$
#          *orientable* and mark the remaining nodes in $Y \setminus \{y_0, y_1\}$ *fixed*; extend $D_x$
#          to a 1-optimal solution
    $(D,K,F)$ = primeMerge($x,Y$)
    return $(D,K,F)$
}
```

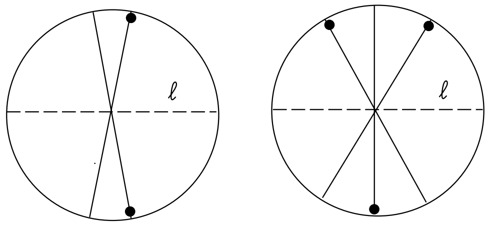


Figure 6.2 Processing a clique node x

Lemma 6.1: Let D_x and (D,K) be as computed above. Then (D,K) is a 1-optimal solution.

Proof:

That (D, K) is a 1-supported diagram of $G_{T(x)}$, with respect to the marker ℓ_{wx}, is immediate. To see that K is a 1-supported kernel, note that the call to `primeMerge(x,Y)` does not introduce any new corner. Hence, the computed set K is exactly the required set $\bigcup_{y\in Y} K_{T(y)}$. It then remains to show that the computed partial diagram (D, K) has the smallest possible deficiency $\delta(D, K)$ among all possible diagrams having exactly $|K|$ corners. This is done by exhausting all possible cases, as follows:

1. For $|Y| = 1$, the claim follows by symmetry of D_x.
2. For $|Y| = 2$, the arrangement of Figure 6.2(a) implies that every possible marker in $D_x \setminus \{y_0, y_1\}$ has a zero deficiency. One may then verify that y_0 and y_1 admit WE-rotations that yield

$$\delta(D,K) = \begin{cases} 0 & \text{if } y_0, y_1 \text{ have type} \in \{[1,1],[0,1]\} \\ 1 & \text{otherwise} \end{cases}$$

3. For $|Y| = 3$, the claim follows easily since $\delta(D, K) = 0$.

Moreover, in all of the above cases, $\delta(D, K)$ has the smallest possible value. ■

The above algorithm does not require any modification to compute a 0-optimal diagram $(D, K \bigcup F)$ if $x = r$. In this case, however, the marker ℓ_{wx} illustrated in Figure 6.2 does not exist.

Timing: the above function calls `primeMerge()` once with no orientable markers; this results in at most 4 calls to `fillIn()`, and hence the $O(n_{T(x)})$ time bound.

6.4 Star Nodes

Similar to the situation for clique nodes, it is possible to identify a winning permutation for each input instance if x is a star node. The process is equally straightforward, however, the analysis requires more cases to be considered. On middle ground, we present key ingredients of a less efficient, but shorter, algorithm that has the same worst-case order of running time as the core function `fillIn()`. The devised function is a refinement of the following brute-force algorithm:

```
function starMerge(x,Y) {                  #inputs and outputs are as in cliqueMerge()
    $D = nil$;  $|K| = |F| = \infty$                                #some out-of-range values
    for (each permutation of the leaves in $D_x$) {
        $(D', K', F')$ = primeMerge($x, Y$)
        if ((|$K'$| < |$K$|) or (|$K'$| == |$K$| and |$F'$| < |$F$|)) { $D = D'$; $K = K'$; $F = F'$ }
    }
    return $(D, K, F)$
}
```

The refined algorithm operates on (at most) two key elements of Y, denoted y_0 and y_1. As in the previous section, the key elements are leaders of the set Y. Specifically, if the distinguished marker ℓ_{wx} is the center c of the star then y_0 and y_1 are as defined in the last section. Else (if $\ell_{wx} \neq c$), then y_0 and y_1 are defined as leaders of the two sets $Y \setminus \{c\}$ and $Y \setminus \{c, y_0\}$, respectively.

Revising the function to to achieve $O(n_{T(x)})$ running time for $|Y| = 1$ is straightforward. It then remains to discuss the case when $|Y| \geq 2$. In this case, let $(D_{T(x)}, K_{T(x)})$ be any $\eta(x)$-optimal diagram for $G_{T(x)}$. The diagram $D_{T(x)}$ then corresponds to some permutation of the markers in D_x. Traversing this permutation from one end to the other, let y_f and y_ℓ be the first and last elements, respectively. We now observe that the size of the $\eta(x)$-supported kernel $|K_{T(x)}|$, and the deficiency $\delta(D_{T(x)}, K_{T(x)})$ are entirely determined by the following factors:

1. if ℓ_{wx} exists and $\ell_{wx} \neq c$: the semicircle deficiences of the two ends of the center chord c, and the location of ℓ_{wx} relative to the extreme chords y_f and y_ℓ, and
2. if $y_f \neq \ell_{wx}$ (or, by symmetry, $y_\ell \neq \ell_{wx}$): the semicircle deficiency of y_f's big-end (respectively, y_ℓ's big-end).

The structure of the remaining chords in the set $Y \setminus \{c, y_f, y_\ell, \ell_{wx}\}$ is irrelevant. Since y_0 and y_1 are the best elements with respect to (2) above, it is easy to see that there exists a winning permutation in which y_0 and y_1 appear in place of y_f and y_ℓ.

The above observations can be put in force by incorporating the following changes: (i) the for-loop is changed to deal with the existing elements of the set $\{y_0, y_1, c, \ell_{wx}\}$ as the only *permutable elements*, all other elements will be declared *fixed* and will be placed contiguously surrounded by some two elements in the above set, and (ii) function `primeMerge()` is signaled that the above four elements are the only *orientable* elements.

Timing: To show that the revised function executes function `fillIn()` a constant number of times note first the for-loop iterates at most 3! times. In each iteration, function `primeMerge()` generates at most 4×4^3 diagrams, and calls `fillIn()` for each of the generated diagrams. It is possible to further refine the function so that `fillIn()` is called at most 4 times for each input instance, as in `cliqueMerge()`; however, we do not describe the refinement as it does not affect the order of the worst-case running time.

7. Complexity of the k-Polygon Problem

We now show that

Theorem 7.1: Given a circle graph G, determining the minimum integer k such that G is a k-polygon graph is NP-complete.

The proof reduces the 3-satisfiability problem where each variable occurs at most 5 times (see, for example, [8] problem [LO2]) to the k-polygon problem. Let $F = \bigwedge_{i=1}^{m} C_i$ be an instance of the above restricted satisfiability problem with m clauses $\{C_1, C_2, \cdots, C_m\}$ and n variables $\{x_1, x_2, \cdots, x_n\}$. Without loss of generality, we may assume that every variable appears in both positive (uncomplemented) and negative (complemented) forms. In addition, we may assume that F is indecomposable; that is F can not be written as $F_1 \bigwedge F_2$ with no common variables between F_1 and F_2. The proof constructs a circle diagram D for a circle graph G having a standard $*$-decomposition tree T isomorphic to a star with a central node r, and $1 + 3m + \Delta(F)$ leaves, where $\Delta(F)$ is a number derived from the formula F.

More precisely, T has a distinguished leaf z corresponding to the circle diagram illustrated in Figure 7.1(a), called the equator gadget. (The figure shows also two corners that will be explained shortly.) In addition, T has $3m$ leaves $\{u_1, u_2, \cdots, u_{3m}\}$, each corresponding to a circle diagram, called a literal gadget, as illustrated in Figure 7.1(b) for an arbitrary leaf u. The remaining $\Delta(F)$ leaves are called link structures. Each such structure has a circle diagram similar to that of Figure 7.1(b) and will be explained shortly.

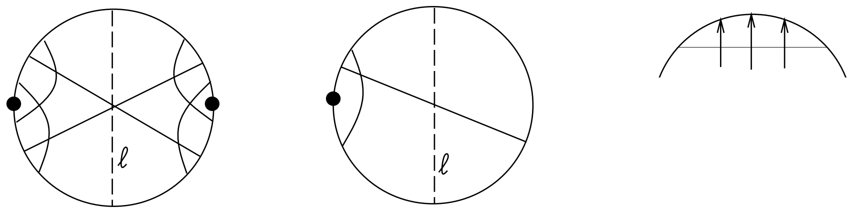


Figure 7.1

At this point, it is important to observe that each of the abovementioned gadgets corresponds to a prime graph. Moreover, each gadget (with the indicated corners now taken into account) forms a 1-optimal diagram (cf. Section 4) with respect to the unique marker (ℓ_{rz} or ℓ_{ru}) in the gadget. Hence, at least as many corners will be introduced by each such gadget in a (complete) polygon diagram of the circle graph G. In describing the structure of the center node r below, we find it convenient to draw a marker ℓ_{ru} representing a literal gadget with an arrow at one endpoint to signal the existence of at least one corner in its corresponding 1-optimal diagram. Likewise, it is convenient to draw the marker ℓ_{rz} in D_r with an arrow at each endpoint to signal the existence of at least one corner in each of the ℓ_{rz}-semicircles of D_z.

We now consider the center node r. Roughly speaking, the corresponding circle diagram D_r can be viewed with the "bidirected" marker ℓ_{rz} placed horizontally (see, Figure 7.3(a), for an example), and two special sets of chords: a set of clause gadgets, placed in the upper ℓ_{rz}-semicircle, and a set of variable gadgets, placed in the lower semicircle. In D_r, each "directed" marker that represents a literal gadget intersects a clause gadget (at one end), the marker ℓ_{rz} (at the middle), and a variable gadget (at the other end). Figure 7.1(c) illustrates a typical clause gadget. The gadget has exactly one chord; the three other directed chords correspond to markers of three literals that occur in that particular clause.

Variable gadgets are slightly more complex than the previous ones. The exact structure of the gadget associated with variable x_i that appears p times in a positive form and q times in a negative form ($p + q \leq 5$), depends on p and q. It suffices, however, to outline the structures for $p \leq q$. Figure 7.2 (a) illustrates a typical variable gadget for $p = 2$ and $q = 3$. In the diagram, chords drawn as undirected solid lines are called clusters. Here, clusters R_{-1} (1 chord), R_{-2} (2 chords), R_c (2 chords) and R_+ (2 chords) are incident with literal gadgets (directed solid lines) and link structures (directed dashed lines - link structures are mentioned above as having circle diagrams identical to that of literal gadgets).

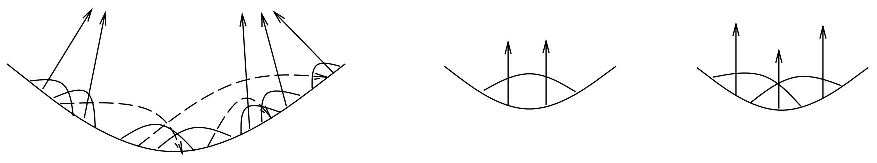


Figure 7.2 Possible structures of variable gadgets

A configuration of a variable gadget is an orientation of the directed solid chords (the literals), and the directed dashed chords (the link structures).

Thus, in a typical configuration some of the directed solid chords are oriented towards the diagram while the remaining chords are oriented away from the diagram towards the clause gadgets. Some of the configurations are well-covered in that each chord in $R_+ \bigcup R_c \bigcup R_{-1} \bigcup R_{-2}$ "straddles" at least one arrow. The NP-completeness reduction assigns a 0/1 value to each variable x_i in F from a well-covered configuration of the corresponding variable gadget as follows. First, suppose that the arrow associated with the middle dashed line in the <u>control</u> cluster R_c lies inside that cluster. To cover R_+, the positive literal arrows must then lie inside that cluster. Consequently, one or more negative literal arrows may then be used to reduce the deficiency of some clauses, and hence we set $x_i = 0$. Conversely, if that particular control arrow is placed inside R_+ then all negative arrows must lie inside $R_{-1} \bigcup R_{-2}$ and the positive arrows can be directed towards the clause gadgets, and hence we set $x_i = 1$.

Variable gadgets for other distributions of p and q, $p \leq q$, can be constructed in a similar way. For $p = 1$ (and $q \geq 3$) the cluster R_+ is replaced with one similar to R_{-1}. For $q = 4$, the cluster R_{-1} is replaced with one similar to R_{-2}. For $[p, q] = [2, 2]$ the control cluster R_c has only one solid chord with two incident link structures. In simpler situations, where $[p, q]$ =$[1, 1]$ or $[1, 2]$ we use the gadgets shown in Figures 7.2(b) and (c), respectively. Clearly, gadgets used to represent cases where $p > q$ are similar to the above. Finally, $\Delta(F)$ is defined to be the number of link structures in the overall diagram of D_r. That is, if we let N_{pq} be the number of variables that appear p times in a positive form and q times in a negative form, and let m_{pq} be the number of link chords in a typical gadget associated with any such variable then $\Delta(F) = \sum_{p+q \leq 5} m_{pq} N_{pq}$. By inspection, $m_{1,1} = m_{1,2} = m_{2,1} = 0$, $m_{2,2} = 2$ and $m_{p,q} = 3$ for all other possible values.

$F = C_1 \wedge C_2 \wedge C_3$

$C_1 = (x_1 \vee x_2 \vee \overline{x}_3)$, $C_2 = (\overline{x}_1 \vee x_2 \vee x_4)$, $C_3 = (\overline{x}_2 \vee x_3 \vee \overline{x}_4)$

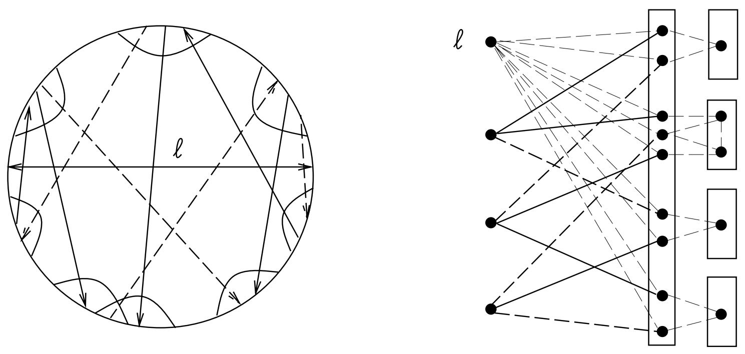


Figure 7.3 An example of a formula F and the associated diagram D_r

Finally, the set $V(G_r)$ of vertices is the union of four disjoint sets: $V_z = \{\ell_{rz}\}$, the set of clause markers $V_C = \{C_1, \cdots, C_m\}$, the set of literal gadgets $V_U = \{u_1, \cdots, u_{3m}\}$, and a family of n sets of vertices $V_X = \{X_1, \cdots, X_n\}$, where X_i corresponds to all chords in the variable gadget associated with x_i. For two sets of vertices, say V_A and V_B, let E_{AB} denote the set of edges with one end in V_A and the other in V_B. Using the above notation, one can write $E(G_r) = E_{zU} \bigcup E_{CU} \bigcup E_{UU} \bigcup E_{UX} \bigcup E_{XX}$.

Figure 7.3(a) gives an example of D_r, where solid chords with single arrows represent positive literals and dashed chords represent negative literals. Here, $\Delta(F) = 0$ and the diagram leads to an embedding of the entire graph G in an 11-gon; the corresponding 0/1 assignments are: $x_1 = x_4 = 1$, $x_2 = 0$ and $x_3 =$ *don't care*. Figure 7.3(b) illustrates the graph G_r with all edges in E_{UU} omitted. To prove that T is a standard $*$-decomposition tree, it then remains to show that:

Lemma 7.1: G_r is prime.
Proof:

To derive a contradiction, assume that $V(G_r)$ has a split $\{V_1, V_2\}$, for which $W_1 \subseteq V_1$ and $W_2 \subseteq V_2$ are the interface vertices, as defined in Section 1. First, observe that the indecomposable formula F yields a 2-connected graph G_r. Consequently, $|W_1|, |W_2| \geq 2$. We next assert the following claims in order. To simplify the presentation, however, we sometimes use the same symbol (or simply, the word "literal") to refer to any particular binary literal in F and the corresponding vertex in G_r. A similar remark applies for any clause in F.

Claim 7.1A: No vertex of V_C appears as an interface vertex.
Proof: Assume to the contrary that some vertex $C \in V_C$ appears as an interface vertex, say $C \in W_1$. Since $|W_1| \geq 2$, there exists another interface vertex $w \in W_1 \backslash \{C\}$. We **remark** that $C \in W_1$ implies that W_2 consists of two or all of the three literals that occur in the binary clause C. The following observations then hold true for w:

1. $w \notin V_C$, since no two clause vertices of V_C share a common literal of V_U.

2. $w \notin V_X$, since C shares at most one literal with any vertex in any variable gadget.

3. If $w \in V_U$ then ℓ_{rz} is an interface vertex, since ℓ_{rz} is adjacent to $w \in W_1$ and all vertices of W_2. In this case, ℓ_{rz} must lie in W_1, and not in W_2 (since $(\ell_{rz}, C) \notin E(G_r)$). We may then choose $w = \ell_{rz}$ (instead of $w \in V_U$), and deal only with the next case.

4. $w = \ell_{rz}$: then all vertices in $(V_U \backslash W_2)$ must lie in V_1 (since $\ell_{rz} \in W_1$). Choose any arbitrary literal $u'' \in W_2$ (recall from the above remark that u'' occurs in the binary clause C). By assumption, each variable occurs at least twice in F. Hence, F contains another literal u' that is associated with the same binary variable as u''; this literal corresponds to a vertex that lies in V_1 (since $(V_U \backslash W_2) \subseteq V_1$, as mentioned above). Now consider the variable gadget X corresponding to the binary variable of which u' and u'' are two literals. By the construction of X, there exists a path from u' to u'' passing through the vertices of X only. This implies that at least one vertex $x \in X$

appears as an interface vertex. Nevertheless, it is impossible to have $x \in W_1$ since W_2 has at least two literals corresponding to two different binary variables. On the other hand, $x \notin W_2$ since $(\ell_{rz}, x) \notin E(G_r)$. Thus, $w \neq \ell_{rz}$.

The above argument exhausts all possible choices of w, and completes the proof. $\square$

Claim 7.1B: No vertex of V_X appears as an interface vertex.
Proof: Similar to the above proof, assume to the contrary that some vertex, say x, of a variable gadget X appears in W_1. The strategy is to exhaust all possibilities for a second interface vertex $w \in W_1 \backslash \{x\}$, using the following observations:

1. $w \notin V_C$ by the previous claim.
2. $w \notin V_X$, since no two vertices of V_X share two or more neighbours (note: this remark applies for any possible structure of the variable gadget X). We may then conclude that no two vertices of V_X appear in W_1.

For the remaining cases, we **remark** that (2) above implies (by symmetry) that no two vertices of V_X appear in W_2. Hence, at least one literal, say u'', appears in W_2.

3. If $w \in V_U$ then each of W_1 and W_2 contains a literal. Consequently, ℓ_{rz} must be one of the interface vertices. Hence it must be the case that $\ell_{rz} \in W_1$, and not in W_2 (since $(\ell_{rz}, x) \notin E(G_r)$). Again, we may choose $w = \ell_{rz}$, and deal with the next case.
4. $w = \ell_{rz}$: here we have $\{\ell_{rz}, x\} \subseteq W_1$; this implies that all vertices of W_2 are literals of the same binary variable that X represents. The remaining literals $(V_U \backslash W_2)$ must then appear in V_1. Recall from the above remark that $u'' \in W_2$. Denote by C the binary clause in which u'' occurs, and note that the other two literals of C must lie in V_1. This implies that C is an interface vertex; a contradiction by Claim 7.1A.

This completes the proof of the claim. $\square$

Now suppose (without loss of generality) that V_1 contains a clause vertex C. Then $C \in V_1 \backslash W_1$ (by Claim 7.1A), and hence all literals in C lie in V_1. Claim 7.1B then implies that all variable gadgets corresponding to these literals appear in $V_1 \backslash W_1$. A similar note holds if $V_1 \backslash W_1$ contains any vertex of a variable gadget X. Here, Claim 7.1B implies that all of the remaining vertices of the gadget X must lie in $V_1 \backslash W_1$, since the subgraph of G_r induced on X is connected. Consequently, all literals of the binary variable represented by X appear in V_1, and hence all clauses in which these literals occur appear in $V_1 \backslash W_1$ (by Claim 7.1A). The above observations, combined with the assumption that F is indecomposable, imply that if a clause vertex C lies in V_1 then all vertices in $V_C \bigcup V_U \bigcup V_X$ lie in V_1. That is, $|V_2| \leq 2$; a contradiction. Hence, no such split $\{V_1, V_2\}$ exists and Lemma 7.1 follows. $\blacksquare$

We are now ready to prove Theorem 7.1:

It is routine to check that F can be validated and that G can be constructed in polynomial time. Lemma 7.1 ensures that T is a valid $*$-decomposition tree of G and hence any optimal diagram of G can be obtained by assigning orientations to the circle diagrams associated with nodes in T and evaluating the tree. A lower bound on the cardinality of an optimal kernel of G is $3m + \Delta(F) + 2$. In fact, it is easy to verify that F is satisfiable if and only if G is a $(3m + \Delta(F) + 2)$-polygon graph. Both directions are straightforward using the above rules of mapping configurations of variable gadgets to 0/1 values of the variables in F. ■

8. Disconnected Polygon Graphs

Up to this point, our main emphasis has been on connected circle graphs. A need may exist, however, for computing optimal polygon representations of arbitrary disconnected circle graphs, and hence, a word should be said in this respect. To this end, we recall the following notation: for any circle graph G, $k(G)$ denotes the size of the smallest polygon required to represent G. We also allow $k(G) = 2$ if G is a permutation graph. In addition, if P is a polygon then let $k(P)$ denote P's size. Now, let $G = \bigcup_{i=1}^{r} G_i$ be a disjoint union of r connected circle graphs $\{G_i | \ 1 \leq i \leq r\}$; the main result of this section is:

Lemma 8.1: $k(G) = (\sum_{i=1}^{r} k(G_i)) \ - \ 2(r-1)$.

Proof:

We first describe a simple iterative construction to show that $k(G) \leq (\sum_{i=1}^{r} k(G_i)) \ - \ 2(r-1)$. For $1 \leq i \leq r$, let P_i be a $k(G_i)$-polygon representation of G_i. The construction performs $r-1$ iterations. At the ith iteration, we construct a polygon representation of $\bigcup_{j=1}^{i+1} G_j$ by $\underline{\text{merging}}$ a polygon diagram P' representing $\bigcup_{j=1}^{i} G_j$ with $P'' = P_{i+1}$. The new polygon representation has $k(P') + k(P'') - 2$ sides, and hence the final polygon diagram for G has size equal to the right-hand side of the inequality. To merge P' with P'', we first identify a corner $c' \in P'$ with a corner $c'' \in P''$, as sketched in Figure 8.1(a), and call the new double-corner point c. In P', let a' and b' be the two sides incident with c', so that the sequence (a', c', b') appears in a clockwise traversal of P'. Similarly, let (b'', c'', a'') be the corresponding sequence in P''. Next, split the double-corner point c into two new points c_u and c_d, where c_u is incident with the two sides a' and a'', and c_d is incident with b' and b''. Finally, remove c_u and c_d, and unify a' with a'' (also, b' with b'') into a single side. This completes the construction.

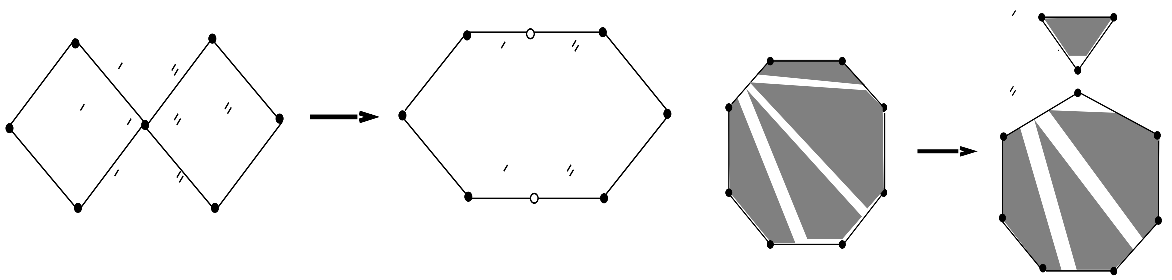


Figure 8.1 Merging and splitting of polygon diagrams

We next induct on r to show that $k(G) \geq (\sum_{i=1}^{r} k(G_i)) - 2(r-1)$. The inequality holds trivially for $r = 1$. So, let $r \geq 2$ and assume that it holds for all circle graphs with at most $r-1$ components, and let $G = \bigcup_{i=1}^{r} G_i$ be an arbitrary circle graph. In addition, let P be a $k(G)$-polygon representation of G. Call a side of P shared if it contains the endpoints of chords associated with two or more components of G. Note that chords of each component of G can be identified with a closed region inside P, as sketched in Figure 8.1(b). Moreover, no two regions intersect each other. It then follows that at least two regions in P are peripheral in that each one has at most two shared sides. Without loss of generality, we may assume that the region corresponding to G_1 is peripheral. Denote by a and c the first and last sides of P that are incident with endpoints of chords in G_1. It is then possible to split P into two polygons:

1. P' (representing G_1) obtained by deleting all chords not in G_1, and all sides not used by G_1, and subsequently joining the two special sides a and c at a new corner point, and
2. P'' (representing $G \setminus G_1$) obtained in a similar way by deleting all chords of G_1.

We then have $k(G) = k(P') + k(P'') - 2$. The inequality then follows since, by definition, $k(P') \geq k(G_1)$, and $k(P'') \geq k(G \setminus G_1) \geq (\sum_{i=2}^{r} k(G_i)) - 2(r-2)$, by the induction hypothesis. ■

Acknowledgment

The authors gratefully acknowledge financial support from the Natural Sciences and Engineering Research Council of Canada.

References

[1] A. Bouchet. Reducing prime graphs and recognizing circle graphs. *Combinatorica*, 7:243–254, 1987.

[2] W.H. Cunningham. Decomposition of directed graphs. *SIAM J. Alg. Disc. Meth.*, 3(2):214–228, 1982.

[3] W.H. Cunningham and J. Edmonds. A combinatorial decomposition theory. *Canadian Journal of Mathematics*, 22:734–765, 1980.

[4] E.S. Elmallah. Algorithms for k-terminal reliability problems with node failures. *Networks*, 22:369–384, 1992.

[5] E.S. Elmallah and L.K. Stewart. Independence and domination in polygon graphs. *Discrete Applied Mathematics*, 40:65–77, 1993.

[6] S. Even, A. Pnueli, and A. Lempel. Permutation graphs and transitive graphs. *J. ACM*, 19:400–419, 1972.

[7] C. P. Gabor, K. J. Supowit, and W.-L. Hsu. Recognizing circle graphs in polynomial time. *J. ACM*, 36:435–473, 1989.

[8] M.R. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-completeness.* W.H. Freeman, San Francisco, 1979.

[9] M. C. Golumbic. *Algorithmic Graph Theory and Perfect Graphs.* Academic Press, 1980.

[10] T.-H. Ma and J. Spinrad. An $O(n^2)$ algorithm for the undirected split decomposition. *J. of Algorithms*, 16:145–160, 1994.

[11] W. Naji. Reconnaissance des graphes de cordes. *Discrete Math.*, 54:329–337, 1985.

[12] J. Spinrad. On comparability and permutation graphs. *SIAM J. Comput.*, 14(3):658–670, 1985.

[13] J. Spinrad. Recognition of circle graphs. *J. of Algorithms*, 16:264–282, 1994.