SIAM J. Discrete Math.
Vol. 15, No. 4, pp. 435–449

APPROXIMATING BANDWIDTH BY MIXING LAYOUTS OF INTERVAL GRAPHS*

D. KRATSCH† AND L. STEWART‡

Abstract. We examine the bandwidth problem in circular-arc graphs, chordal graphs with a bounded number of leaves in the clique tree, and k-polygon graphs (fixed k). We show that all of these graph classes admit efficient approximation algorithms which are based on exact or approximate bandwidth layouts of related interval graphs. Specifically, we obtain a bandwidth approximation algorithm for circular-arc graphs that executes in $O(n \log^2 n)$ time and has performance ratio 2, which is the best possible performance ratio of any polynomial time bandwidth approximation algorithm for circular-arc graphs. For chordal graphs with at most k leaves in the clique tree, we obtain a performance ratio of $2k$ in $O(k(n+m))$ time, and our algorithm for k-polygon graphs has performance ratio $2k^2$ and runs in time $O(n^3)$.

Key words. interval graphs, circular-arc graphs, chordal graphs, polygon graphs, bandwidth problem, approximation algorithm

AMS subject classifications. 05C78, 05C85, 68R10, 68W25

PII. S0895480199359624

1. Introduction.

A *layout* of a graph $G = (V, E)$ is an assignment of distinct integers from $\{1, \ldots, n\}$ to the elements of V. Equivalently, a layout L may be thought of as an ordering $L(1), L(2), \ldots, L(n)$ of V, where $|V| = n$. We shall use $<_L$ to denote the ordering of the elements in a layout L. The *width* of a layout L, $b(G, L)$, is the maximum over all edges $\{u, v\}$ of G of $|L(u) - L(v)|$. That is, it is the length of the longest edge in the layout. The *bandwidth* of G, $bw(G)$, is the minimum width over all layouts. A *bandwidth layout* for graph G is a layout satisfying $b(G, L) = bw(G)$.

The problem of finding the bandwidth of a graph has applications in sparse matrix computations. An overview of the bandwidth problem is given in [5]. The minimum bandwidth decision problem (Given a graph $G = (V, E)$ and integer k, is $bw(G) \leq k$?) is known to be NP-complete [27], even for trees having maximum degree 3 [15], caterpillars with hairs of length at most 3 [26], and cobipartite graphs [22]. The problem is polynomially solvable for caterpillars with hairs of length 1 and 2 [2], cographs [18], graphs with few P_4's [24], and interval graphs [19, 25, 29].

To date there was not much known about the approximation hardness of the bandwidth minimization problem for graphs in general. Recently, Feige presented an approximation algorithm with performance ratio $O(\log^{9/2} n)$ [12]. Very recently, Unger has shown in [30] that, assuming P≠NP, there is no polynomial time approximation algorithm with constant performance ratio for the bandwidth minimization problem for graphs, even when the inputs are restricted to a special class of trees known as caterpillars of hairlength 3.

*Received by the editors July 30, 1999; accepted for publication (in revised form) March 22, 2002; published electronically July 16, 2002. This work appeared in a preliminary form in *Proceedings of the 16th International Symposium on Theoretical Aspects in Computer Science,* Trier, Germany, 1999, and was partially supported by the NSERC.
http://www.siam.org/journals/sidma/15-4/35962.html

†Fakultät für Mathematik und Informatik, Friedrich-Schiller-Universität, 07740 Jena, Germany. Current address: Université de Metz, Laboratoire d'Informatique Théorique et Appliquée, 57045 Metz Cedex 01, France (kratsch@lita.univ-metz.fr).

‡Department of Computing Science, University of Alberta, Edmonton, AB, Canada, T6G 2E8 (stewart@cs.ualberta.ca).

Since the bandwidth minimization problem remains NP-complete for such simple classes of graphs, and since no polynomial time algorithm for approximating the bandwidth of general graphs, or even trees, to within a constant factor exists unless P=NP, it is worthwhile to investigate approximation algorithms for this problem on restricted classes of graphs. Some results in this direction have been presented in [22].

In this paper, we examine the bandwidth problem in circular-arc graphs, chordal graphs with a bounded number of leaves in the clique tree, and k-polygon graphs (fixed k). All of these graph classes admit efficient approximation algorithms which are based on exact or approximate bandwidth layouts of related interval graphs.

Specifically, we obtain a bandwidth approximation algorithm for circular-arc graphs that has performance ratio 2 and executes in $O(n \log^2 n)$ time or performance ratio 4 while taking $O(n)$ time. For chordal graphs with at most k leaves in the clique tree, we obtain a performance ratio of $2k$ in $O(k(n+m))$ time, and our algorithm for k-polygon graphs has performance ratio $2k^2$ and runs in time $O(n^3)$.

Finally, it is worth mentioning that our approximation algorithm with performance ratio 2 for circular-arc graphs has *optimal* performance ratio, since there is no polynomial time bandwidth approximation algorithm for circular-arc graphs with performance ratio $2 - \epsilon$ for any $\epsilon > 0$ unless P=NP [30].

2. Preliminaries. For $G = (V, E)$, we will denote $|V|$ as n and $|E|$ as m. We sometimes refer to the vertex set of G as $V(G)$ and the edge set as $E(G)$. We let $N(v)$ denote the set of vertices adjacent to v. The *degree* of a vertex v, $degree(v)$, is the number of vertices adjacent to v. $\Delta(G)$ denotes the maximum degree of a vertex in graph G. The subgraph of $G = (V, E)$ induced by $V' \subseteq V$ will be referred to as $G[V']$.

The following well-known lower bound on the bandwidth of a graph is in [6].

LEMMA 1 (the degree bound [7]). *For any graph G, $bw(G) \geq \Delta(G)/2$.*

The distance in graph $G = (V, E)$ between two vertices $u, v \in V$, $d_G(u, v)$, is the length of a shortest path between u and v in G. For any graph $G = (V, E)$, the dth power of G, G^d, is the graph with vertex set V and edge set $\{\{u, v\} | d_G(u, v) \leq d\}$.

LEMMA 2 (the distance bound [22], also attributed in part to [7] in [5]). *Let G and H be graphs with the same vertex set V, such that $E(G) \subseteq E(H) \subseteq E(G^d)$ or $E(H) \subseteq E(G) \subseteq E(H^d)$ for an integer $d \geq 1$, and let L be an optimal layout for H, i.e., $b(H, L) = bw(H)$. Then L approximates the bandwidth of G by a factor of d, i.e., $b(G, L) \leq d \cdot bw(G)$.*

Many references, including [17], contain comprehensive overviews of the many known structural and algorithmic properties of interval graphs.

DEFINITION. *A graph $G = (V, E)$ is an interval graph if there is a one-to-one correspondence between V and a set of intervals of the real line such that, for all $u, v \in V$, $\{u, v\} \in E$ if and only if the intervals corresponding to u and v have a nonempty intersection.*

A set of intervals whose intersection graph is G is termed an *interval model* for G. Many algorithms exist which, given a graph $G = (V, E)$, determine whether or not G is an interval graph and, if so, construct an interval model for it in $O(n+m)$ time (see, for example, [4, 8]). We assume that an interval model is given by a left endpoint and a right endpoint for each interval, namely, $left(v)$ and $right(v)$ for all $v \in V$. Furthermore, we assume that we are also given a sorted list of the endpoints and that the endpoints are distinct. We will sometimes blur the distinction between an interval and its corresponding vertex, when no confusion can arise.

Polynomial time algorithms for computing the exact bandwidth of an interval graph have been given in [19, 25, 29]. For an interval graph with n vertices, Kleitman and Vohra's algorithm solves the decision problem ($bw(G) \leq k$?) in $O(nk)$ time and can be used to produce a bandwidth layout in $O(n^2 \log n)$ time, and Sprague has shown how to implement Kleitman and Vohra's algorithm to answer the decision problem in $O(n \log n)$ time and thus produce a bandwidth layout in $O(n \log^2 n)$ time.

The following two lemmas demonstrate that, for interval graph G, a layout L with $b(G, L) \leq 2 \cdot bw(G)$ can be obtained in time $O(n)$, assuming the sorted interval endpoints are given. The second proof is similar to the first and is therefore omitted.

LEMMA 3. *Given an interval graph G, the layout L consisting of vertices ordered by right endpoints of corresponding intervals has $b(G, L) \leq 2 \cdot bw(G)$.*

Proof. Let L be the layout of vertices ordered by right interval endpoints. We first observe that, for all $u, v \in V$ such that $\{u, v\} \in E$ and $u <_L v$, all vertices between u and v in L are adjacent to v. Now consider a longest edge in L, i.e., an edge $\{u, v\}$ such that $|L(u) - L(v)| = b(G, L)$. Assume, without loss of generality, that $u <_L v$. From the previous observation, it must be that $degree(v) \geq L(v) - L(u) = b(G, L)$. Now the degree bound (Lemma 1) implies $bw(G) \geq b(G, L)/2$. □

LEMMA 4. *Given an interval graph G, the layout L consisting of vertices ordered by left endpoints of corresponding intervals has $b(G, L) \leq 2 \cdot bw(G)$.*

We will use the following lemma in subsequent sections of the paper.

LEMMA 5. *Let I be a set of intervals on the real line corresponding to interval graph $G = (V, E)$. Let p_1 be a point on the line such that at least one interval endpoint is to the left of p_1 and only left endpoints are to the left of p_1. Let p_2 be a point on the line such that at least one interval endpoint is to the right of p_2 and only right endpoints are to the right of p_2. Let C_1 be the set of all intervals that contain p_1, and let C_2 be the set of all intervals that contain p_2. If L is a layout for G in which vertices are ordered by increasing left endpoints of corresponding intervals or by increasing right endpoints, or if L is a layout produced by Kleitman and Vohra's bandwidth algorithm* [19], *then*

(i) *for all $v \in C_1$: $\{v, L(1)\} \in E$, and*

(ii) *for all $v \in C_2$: $\{v, L(n)\} \in E$.*

Proof. Part (i) for the left endpoint ordering follows from the fact that $L(1) \in C_1$ and C_1 is a clique. In the other two layouts, $L(1)$ is the interval with the smallest right endpoint. This interval is either in C_1 or is contained in all intervals of C_1. Thus, (i) holds for the three layouts.

Part (ii) follows immediately for the right endpoint layout, since $L(n) \in C_2$. In the left endpoint order, $L(n)$ is either in C_2 or contained in all intervals of C_2, implying (ii).

Finally, we prove (ii) for Kleitman–Vohra layouts. Please refer to the algorithm of [19]. Consider the moment when the vertex of C_2 with largest left endpoint, c, is labelled. If only vertices of C_2 remain to be labelled, then the last vertex will be an element of C_2, and we are done. Otherwise, there is an interval i with a smaller right endpoint that remains to be labelled. This implies that $c \in S^q_{j_0}$ was chosen in Step 8, and $i \notin S^q_{j_0}$. Since $i \notin S^q_{j_0}$, we have $q + j_0 < n$. Thus, $M(c) < n$, and there is some vertex already labelled that is adjacent to c but not to i; otherwise, we contradict the current choice of c. Thus, the interval i is properly contained in c and, therefore, i is properly contained in all intervals corresponding to vertices of C_2. This completes the proof. □

3. Circular-arc graphs.

3. Circular-arc graphs. Circular-arc graphs are the intersection graphs of arcs on a circle. Thus, a graph $G = (V, E)$ is a circular-arc graph if and only if it has a (not necessarily unique) circular-arc model or representation, consisting of a set of arcs on a circle, such that, for all $u, v \in V$, $\{u, v\} \in E$ if and only if the arcs corresponding to u and v have a nonempty intersection. In such a model, we assume, without loss of generality, that the arc endpoints are distinct, and we label the endpoints from 1 to $2n$ in clockwise order around the circle, starting at an arbitrary endpoint. Thus, each vertex $v \in V$ corresponds to an arc given by its counterclockwise endpoint, $ccw(v)$, and its clockwise endpoint, $cw(v)$. We refer to any segment of the circle by its two endpoints and the direction of traversal; i.e., $[p_1, p_2]_{cw}$ refers to the closed arc covered by a clockwise traversal beginning at p_1 and ending at p_2. The arc $[p_1, p_2]_{ccw}$ is the set of all points in a counterclockwise traversal from p_1 to p_2, and parentheses will indicate that the arc is open at one or both ends. Note that, for any two points on the circle, p_1 and p_2, the arcs $[p_1, p_2]_{cw}$ and $[p_1, p_2]_{ccw}$ cover the entire circle, and their intersection is $\{p_1, p_2\}$.

Eschen and Spinrad [11] have given an $O(n^2)$ algorithm which determines whether or not an n-vertex graph is a circular-arc graph. If so, the algorithm produces a circular-arc model for the graph. Our algorithms assume that the input circular-arc graph is given as a set of arcs on a circle. We are not aware of any previous results on the bandwidth of circular-arc graphs.

Henceforth, we will refer to a set of $2n$ *scanpoints* on the circle, none of which is an arc endpoint, such that exactly one of these points is between each consecutive pair of arc endpoints. We shall label these points from 1 to $2n$ in clockwise order, beginning at any one.

Our bandwidth approximation algorithm works as follows for a circular-arc graph G. Roughly speaking, we cut the circular-arc representation in half to form two equal-sized interval graphs, compute exact or approximate bandwidth layouts for the two interval graphs, and then mix the two layouts to form an approximate bandwidth layout for G.

Let $G = (V, E)$ be a circular-arc graph with corresponding circular-arc representation. The first step is to find a scanpoint p on the circle such that $|C_1 \cup C_2 \cup A| = |C_1 \cup C_2 \cup B|$, where C_1 is the set of arcs that contain scanpoint 1, C_2 is the set of arcs that contain scanpoint p, A is the set of arcs entirely contained in $(1, p)_{cw}$, and B is the set of arcs entirely contained in $(1, p)_{ccw}$. Note that $C_1 \cup C_2 \cup A \cup B = V$. We will use scanpoints 1 and p to cut the circle and create two equal-sized interval graphs.

```
ALGORITHM 1. Procedure FINDp.
Let $C_1 \leftarrow C_2 \leftarrow$ all arcs that contain scanpoint 1;   $A \leftarrow \emptyset$;   $B \leftarrow V \setminus C_1$
$a \leftarrow |C_1|$;   $b \leftarrow n$   { $a = |C_1 \cup C_2 \cup A|$;   $b = |C_1 \cup C_2 \cup B|$ }
$p \leftarrow 1$
repeat until $a = b$ or $p = 2n$
        { Invariant: $a \leq b$}
        { Variant: $2n - p$}
    $p \leftarrow p + 1$
    if the endpoint between $p - 1$ and $p$ is a ccw endpoint (say of arc $i$) then
        $C_2 \leftarrow C_2 \cup \{i\}$
        if $i \notin C_1$ then
            $B \leftarrow B \setminus \{i\}$;   $a \leftarrow a + 1$
    if between $p - 1$ and $p$ is a cw endpoint (of arc $i$) then
        $C_2 \leftarrow C_2 \setminus \{i\}$
        if $i \notin C_1$ then
```

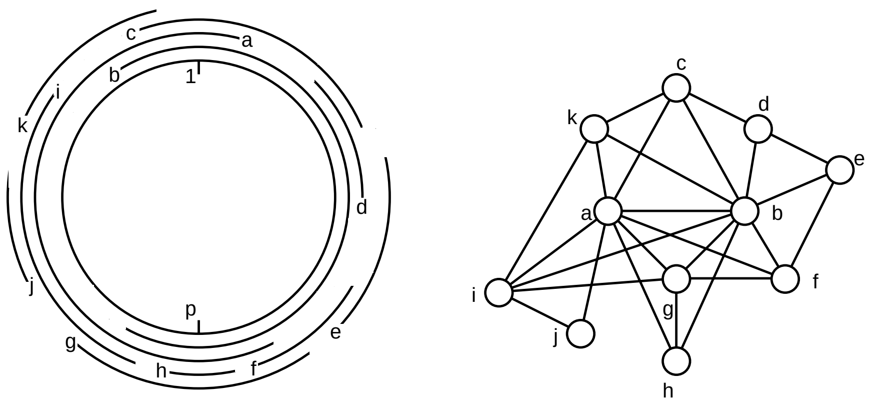

FIG. 1. *A set of arcs on a circle and the corresponding circular-arc graph.*

```
        $A \leftarrow A \cup \{i\}$;   $b \leftarrow b - 1$
    { *Now $C_2$ is the set of arcs that contain point $p$*}
    $\{|C_1 \cup C_2 \cup A| = |C_1 \cup C_2 \cup B|\}$.
```

Claim 1. **Procedure** *FINDp* will terminate with $a = b$.

Proof. We leave it to the reader to verify the stated invariant and variant. If the loop terminates with $p = 2n$, then all arc endpoints will have been examined. For all arcs, except those of C_1, a will have been incremented by 1 and b will have been decremented by 1. Let a_i and a_f be the initial and final values, respectively, of variable a and b_i and b_f the initial and final values, respectively, of variable b. Upon termination of the loop with $p = 2n$, $a_f = a_i + n - |C_1| = |C_1| + n - |C_1| = n$ and $b_f = b_i - (n - |C_1|) = n - n + |C_1| = |C_1|$. However, then $b_f < a_f$ (assuming $C_1 \neq V$), contradicting our invariant. □

We may assume that A and B will be nonempty; otherwise, G can be partitioned into two cliques, one of which must have size at least $n/2$, implying (by Lemma 1) $bw(G) \geq n/2 - 1$. Thus, any layout in which the first and last vertices are not adjacent is a 2-approximation.

A set of arcs on a circle and the corresponding graph are shown in Figure 1, along with possible choices of scanpoints 1 and p. In this example, $C_1 = \{a, b, c\}$, $C_2 = \{a, b, g, h\}$, $A = \{d, e, f\}$, and $B = \{i, j, k\}$.

We now describe how to construct two interval subgraphs of G by cutting the circle at scanpoints 1 and p. We wish to cut the circle and the arcs of C_1 and C_2 at scanpoints 1 and p, producing two line segments, each with a set of intervals that correspond to an interval graph. However, if any arc, say v, contains both scanpoints 1 and p, then it covers one entire part of the circle (i.e., $[1, p]_{cw}$ or $[1, p]_{ccw}$) and appears as two disconnected pieces in the other part. Thus, this second part of the circle may not correspond to an interval subgraph, as vertex v is represented by two disconnected intervals. We eliminate this problem by shrinking v's arc on the circle so that it no longer contains p and thus v is removed from C_2. The altered set of arcs might not represent all of the edges of G; specifically, some edges between v and elements of A (or B) may be missing. Let E' denote edges of G that are not represented by the changed arcs. Note that the sets $C_1 \cup C_2 \cup A$ and $C_1 \cup C_2 \cup B$ remain unchanged. These alterations, applied to the circular-arc model of Figure 1, yield the set of arcs shown in Figure 2. After the alterations, C_2 is changed to $\{g, h\}$,

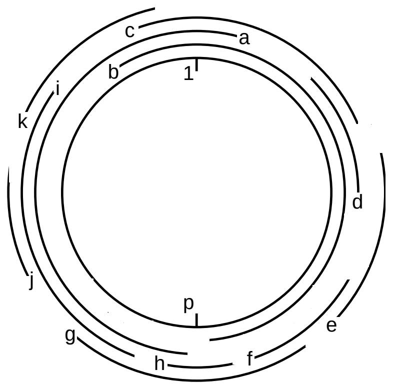

FIG. 2. *Altering the circular-arc model.*

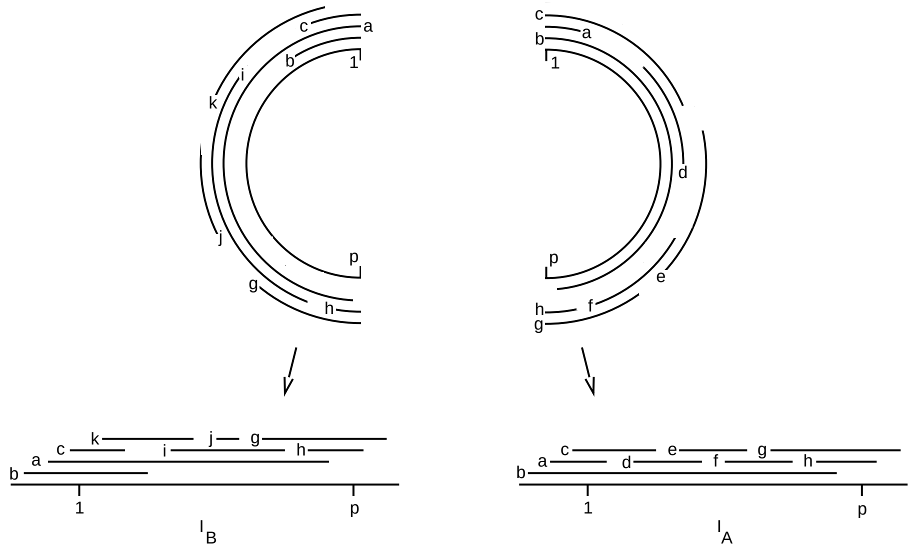

FIG. 3. *Cutting the circular-arc model to form two interval graphs.*

C_1, A, and B remain unchanged, and $E' = \{\{a, f\}, \{b, i\}\}$.

Now we can cut the circle and the arcs of C_1 and C_2 at scanpoints 1 and p, producing two line segments, $[1, p]_{cw}$ and $[1, p]_{ccw}$. The arcs of the circular-arc model become intervals on the two lines. Let I_A (respectively, I_B) be the resulting set of intervals on the line segment $[1, p]_{cw}$ (respectively, $[1, p]_{ccw}$). We may assume that the intervals of $C_1 \cup C_2$ are altered slightly in I_A and in I_B without changing intersections, so that interval endpoints are distinct.

Let $G_A = (V_A, E_A)$ and $G_B = (V_B, E_B)$ be the intersection graphs of I_A and I_B, respectively. Now G_A and G_B are both interval graphs and (not necessarily induced) subgraphs of G. Furthermore, $|V_A| = |V_B|$, and $E_A \cup E_B \cup E' = E$. Figure 3 illustrates this process for the example of Figures 1 and 2.

Our method for obtaining an approximate bandwidth layout for a circular-arc graph is to first compute exact or approximate bandwidth layouts, L_A and L_B, for G_A and G_B, respectively, and then mix the two layouts.

Different methods of computing L_A and L_B yield different approximation bounds and time complexities for our algorithm.

Regardless of how we obtain L_A and L_B, the mixing is done as follows.

Let $k = |C_1 \cup C_2 \cup A| = |C_1 \cup C_2 \cup B|$. Given

$$L_A = L_A(1), L_A(2), \ldots, L_A(k)$$

and

$$L_B = L_B(1), L_B(2), \ldots, L_B(k)$$

we begin by producing

$$L_M = L_A(1), L_B(1), L_A(2), L_B(2), \ldots, L_A(k), L_B(k).$$

For convenience, we will refer to elements of L_A as having the color *red* and elements of L_B as having the color *blue*. Notice that L_M will contain two copies of each vertex of $C_1 \cup C_2$—one red and one blue. For each $v \in C_1 \cup C_2$, we shall distinguish between the two copies of v in L_M as follows: the red copy will be referred to as v_{red} and the blue copy as v_{blue}. Each vertex of $A \cup B$ occurs only once in L_M.

From L_M, we produce L by deleting the leftmost copy of each vertex of C_1 and the rightmost copy of each vertex of C_2. Recall that we constructed C_1 and C_2 so that no vertex appears in both. Thus, L is a layout for G. We now prove a bound on the width of L in terms of the widths of L_A and L_B.

LEMMA 6. *Let $G = (V, E)$ be a circular-arc graph, and let I_A, I_B, G_A, and G_B be constructed, as previously described, from a circular-arc model for G. Let L_A and L_B be layouts for G_A and G_B, respectively, satisfying*

- *for all $v \in C_1$: $\{v, L_A(1)\}, \{v, L_B(1)\} \in E$, and*
- *for all $v \in C_2$: $\{v, L_A(k)\}, \{v, L_B(k)\} \in E$.*

Let L_M and L be obtained from L_A and L_B as previously described. Then

$$b(G, L) \leq 2 \cdot \max[b(G_A, L_A), b(G_B, L_B)].$$

Proof. We will consider an arbitrary edge of G, $\{u, v\} \in E$. We first observe that if u and v have the same color, say red, then $|L(u) - L(v)| \leq |L_M(u) - L_M(v)| = 2 \cdot |L_A(u) - L_A(v)|$. Such edges, therefore, cannot contradict the claim. We shall refer to such edges as red edges or blue edges, depending upon the color of the endpoints. Similarly, any edge for which we can find a longer red or blue edge in L_M cannot contradict the claim.

Consider the edge $\{u, v\} \in E$, where $u <_L v$. We must show that $|L(u) - L(v)| \leq 2 \cdot \max[b(G_A, L_A), b(G_B, L_B)]$.

Case 1. The intervals corresponding to u and v intersect in I_A or I_B or both.

Hence $\{u, v\} \in E \setminus E'$. Suppose, without loss of generality, that the intervals intersect in I_A. If u and v are both red, then our earlier observation applies, and we are done.

Next, suppose that u is red and v is blue. When L was formed from L_M, v_{red} must have been deleted. If v_{red} is to the right of v_{blue} in L_M, then there is a longer red edge $\{u, v\}$ in L_M, and this completes the proof. Suppose v_{red} is to the left of v_{blue} in L_M. This implies that $v \in C_1$, since the leftmost copy was deleted from L_M to form L. However, then v_{blue} is adjacent to the first blue vertex in L_M, implying that $|L(v) - L(u)| \leq |L_M(v_{blue}) - L_M(2)| \leq 2 \cdot |L_B(v) - L_B(1)|$.

Now consider the case where u is blue and v is red. The red copy of u has been deleted. If u_{red} is to the left of u_{blue} in L_M, then there is a longer red edge in L_M. Otherwise, we have $u \in C_2$. However, then u_{blue} is adjacent to the last vertex of L_M, giving a longer blue edge.

Finally, we consider the case where u and v are both blue. If the corresponding intervals intersect in I_B, then we are done by the previous argument. Otherwise, one of u and v is in C_1 and the other is in C_2. If $u \in C_1$ and $v \in C_2$, then the red edge $\{u_{red}, v_{red}\}$ is longer in L_M than $\{u, v\}$ in L. If $u \in C_2$ and $v \in C_1$, then u_{blue} is adjacent to the last vertex of L_M, giving a blue edge in L_M longer than $\{u, v\}$ in L.

Case 2. The intervals corresponding to u and v intersect neither in I_A nor in I_B.

Hence $\{u, v\} \in E'$. Then it must be that exactly one of the vertices corresponds to an arc which, in the original circular-arc representation, covers all of one side of the circle and extends into the other side covering both scanpoints 1 and p. Assume, without loss of generality, that the arc covers $[1, p]_{cw}$ and appears as two disconnected arcs in $[1, p]_{ccw}$. In constructing I_B, the part of the arc that covered p and extended into $[1, p]_{ccw}$ was removed. This must be the area where the arcs corresponding to u and v intersected in the original circular-arc representation. This implies that the other arc is in B, and it therefore occurs as a blue vertex only in L_M and in L.

Suppose that u is the arc that was altered. Then $u \in C_1$ and $v \in B$. Thus, it is the rightmost copy of u that remains in L. The red copy of u in L_M is adjacent to all other red vertices, including $L_M(2k-1)$. Thus, if u in L is red, then there is a red edge in L_M that is longer than the $\{u, v\}$ edge in L. If u in L is blue, then u_{red} has a longer edge in L_M to $L_M(2k-1)$.

Now consider the case where v was altered. Then $v \in C_1$ and $u \in B$. The rightmost copy of v from L_M remains in L, and the red copy of v is adjacent to all other red vertices in L_M, including $L_M(1)$. If v in L is red, then the red edge $\{v, L_M(1)\}$ is longer than the edge $\{u, v\}$ in L. If v is blue in L, then v is adjacent to $L_M(2)$ by Lemma 5 and the construction of L_M; thus, there is a longer blue edge. $\square$

THEOREM 7. *The bandwidth of a circular-arc graph can be approximated to within a factor of four in $O(n)$ time and to within a factor of two in $O(n \log^2 n)$ time.*

Proof. We have three approximation algorithms for approximating the bandwidth of a circular-arc graph, namely, the algorithm previously described in which

(i) L_A and L_B are layouts of vertices ordered by left endpoints of intervals,

(ii) L_A and L_B are layouts of vertices ordered by right endpoints of intervals, or

(iii) L_A and L_B are layouts computed by Kleitman and Vohra's algorithm.

Algorithms (i) and (ii) have time complexity $O(n)$, provided the sorted arc endpoints are given, and they output a layout L that satisfies $b(G, L) \leq 2 \cdot \max[b(G_A, L_A), b(G_B, L_B)] \leq 4 \cdot bw(G)$.

Algorithm (iii) requires $O(n \log^2 n)$ time but produces a layout L satisfying $b(G, L) \leq 2 \cdot \max[b(G_A, L_A), b(G_B, L_B)] \leq 2 \cdot bw(G)$.

These performance ratios follow from Lemmas 5 and 6 and the fact that any subgraph of graph G has bandwidth not larger than $bw(G)$. $\square$

4. Chordal graphs with clique trees having a bounded number of leaves.

A graph G is a *chordal* graph if every cycle of length greater than three has a chord. Chordal graphs are exactly the intersection graphs of subtrees in a tree [16]. More precisely, for each chordal graph $G = (V, E)$, there exists a tree T such that

- the vertices of T correspond to the maximal cliques of G, and

• the vertices of T corresponding to cliques of G containing any fixed vertex $v \in V$ induce a subtree T_v of T.

Note the consequence that two vertices of G are adjacent if and only if their corresponding subtrees have nonempty intersection. For a given chordal graph $G = (V, E)$, such a tree, called a *clique tree* for G, will have at most n nodes and can be constructed in $O(n + m)$ time [3].

We use the idea of mixing layouts of interval graphs, as in the previous section. While a circular-arc graph roughly consists of two interval graphs arranged in a circle, a chordal graph may be thought of as several interval graphs arranged in a tree-like structure. We restrict our attention to chordal graphs having a bounded number of leaves in their clique trees. A chordal graph with k leaves in its clique tree may be viewed as a collection of k interval graphs. For a chordal graph $G = (V, E)$ with at most k leaves in the corresponding clique tree, we compute a layout L such that $b(G, L) \leq 2k \cdot bw(G)$.

The method is as follows, assuming a clique tree T has been computed for a given chordal graph $G = (V, E)$.

```
1. Root $T$ at an arbitrary vertex, $r$.
2. Let $k$ be the number of leaves of $T$ (excluding $r$). For each root-to-leaf path
   $P_i$ in $T$, the collection of subtrees $T_v$ (for $v \in V$), restricted to $P_i$, form a
   set of intervals. Let $I_i$ be this set of intervals in which the left endpoint of
   each interval is taken to be the one closer to $r$. Let $G_i = (V_i, E_i)$ be the
   corresponding interval graph.
3. for $i \leftarrow 1$ to $k$ do
       $L_i \leftarrow$ layout for $G_i$ consisting of $V_i$ ordered by increasing left endpoints
              of intervals (with ties broken arbitrarily but the same way in all
              the $L_i$'s)
4. Mix the $L_i$'s to form $L_M$, as follows:
       $L_M \leftarrow L_1(1)L_2(1)L_3(1)\dots L_k(1)L_1(2)L_2(2)\dots L_k(2)\dots$.
5. For each vertex $v \in V$ that appears in more than one of the $G_i$'s, delete all
   but the rightmost copy of $v$ from $L_M$. The result is a layout $L$ for $G$.
```

The following lemmas apply in the context of the previously described method.

LEMMA 8. *Each G_i is an interval graph.*

Proof. This follows from the construction of the G_i's and properties of the clique tree. ∎

LEMMA 9. $E_1 \cup E_2 \cup \cdots \cup E_k = E$.

Proof. If $\{u, v\} \in E$, then u and v occur together in some clique corresponding to a vertex of T. Thus the edge $\{u, v\}$ will occur in every G_i whose corresponding path P_i contains that vertex of T. ∎

LEMMA 10. *For all $\{u, v\} \in E$, either*

• *for all $1 \leq i \leq k : u \in V_i$ implies $(v \in V_i$ and $\{u, v\} \in E_i)$, or*

• *for all $1 \leq i \leq k : v \in V_i$ implies $(u \in V_i$ and $\{u, v\} \in E_i)$.*

Proof. Let $\{u, v\} \in E$. Then T_u and T_v intersect. Let c_{uv} be the vertex of T, closest to r, at which T_u and T_v intersect. c_{uv} is the closest to r vertex for at least one of T_u and T_v; otherwise, we contradict our choice of c_{uv}, since the path from c_{uv} to r in T is unique and since T_u and T_v are both connected.

Suppose c_{uv} is the vertex of T_u closest to r in T. Then, for any V_i that contains u, the corresponding path P_i must contain c_{uv}, and the conclusion follows.

Similarly, if c_{uv} is the vertex of T_v closest to r, then, for every V_i containing v, the corresponding path P_i contains c_{uv}. ∎

LEMMA 11. *Each G_i is an induced subgraph of G.*

Proof. This follows by an argument similar to the previous proof. □

LEMMA 12. *$b(G, L) \leq 2k \cdot bw(G)$.*

Proof. Let $\{u, v\} \in E$ and consider the length of $\{u, v\}$ in L, i.e., $|L(u) - L(v)|$. Assume, without loss of generality, that $u <_L v$. If the copies of u and v remaining in L are from the same interval subgraph G_i, then

$$\begin{aligned} |L(u) - L(v)| &\leq |L_M(u) - L_M(v)| \\ &\leq k \cdot |L_i(u) - L_i(v)| \\ &\leq 2k \cdot bw(G_i) \\ &\leq 2k \cdot bw(G). \end{aligned}$$

Suppose the occurrences of u and v in L are from different interval subgraphs, G_u and G_v, respectively. Since $\{u, v\} \in E$, we know by Lemma 10 that

- $v \in G_u$ and $\{u, v\} \in G_u$, or
- $u \in G_v$ and $\{u, v\} \in G_v$.

If $u \in G_v$ then the occurrence of u in G_v is to the left (in L_M) of the occurrence of u in G_u. Thus

$$\begin{aligned} |L(u) - L(v)| &\leq |L_M(u \text{ of } G_v) - L_M(v \text{ of } G_v)| \\ &\leq k \cdot |L_v(u) - L_v(v)| \\ &\leq 2k \cdot bw(G_v) \\ &\leq 2k \cdot bw(G). \end{aligned}$$

Otherwise, $u \notin G_v$ and $v \in G_u$, implying that the vertex of T_v closest to r is closer to r than the vertex of T_u closest to r. That is, in I_u, $left(v) < left(u)$, and hence $v <_{L_u} u$.

Let f_{uv} be the last vertex of T (i.e., farthest from the root) that is in both G_u and G_v. The set of left endpoints from r to f_{uv} are identical in both I_u and I_v. Suppose there are q of them. Then, in L_M, both occurrences of v appear in the first $k \cdot q$ positions, and the occurrence of u from G_u is to the right. This contradicts that the occurrences of u and v under consideration satisfy $u <_L v$. □

THEOREM 13. *Let $G = (V, E)$ be a chordal graph having a clique tree with at most k leaves. Then a layout L for G satisfying $b(G, L) \leq 2k \cdot bw(G)$ can be computed in $O(k(n + m))$ time.*

Proof. The proof follows from the previous discussion. □

Recently, Fomin [13] showed how to improve the previous bound by choosing the root r of the clique tree T such that at most $\frac{2}{3}k$ of the resulting interval graphs can be laid out on either side of the vertices of r. The layout L obtained after mixing the L_i's and removing duplicate vertices has $b(G, L) \leq \frac{4}{3}k \cdot bw(G)$.

Since the bandwidth problem remains NP-complete for trees, a subclass of chordal graphs, it is worth mentioning that our algorithm outputs a layout L satisfying $b(G, L) \leq k$ if G is a tree with at most k leaves. Notice that Ando, Kaneko, and Gervacio showed in [1] that every tree with k leaves has bandwidth at most $\lceil k/2 \rceil$. Furthermore, their construction can easily be transformed into an efficient algorithm to compute a layout of width at most $\lceil k/2 \rceil$.

5. *k*-polygon graphs for fixed *k*.

A *triangulation* of a graph G is a chordal graph H with the same vertex set as G such that G is a subgraph of H. A triangulation

H of a graph G is called a *minimal triangulation* of G if no proper subgraph of H is a triangulation of G.

In this section we combine results of the previous section with results on minimal triangulations of k-polygon graphs as follows. First, we generalize a result on minimal triangulations of AT-free graphs in [22] to show that every minimal triangulation H of a k-polygon graph G is a spanning subgraph of G^k (Theorem 19). Second, we use a representation theorem for minimal triangulations of a circle graph provided in [23] to obtain a representation theorem for minimal triangulations of k-polygon graphs (Theorem 22). Then we show how to transform any k-polygon graph G into a minimal triangulation H of G (and thus H is a chordal graph) such that H has a clique tree with at most k leaves. Combining all this with Lemma 2, and the approximation algorithm of the previous section, we obtain an $O(n^3)$ approximation algorithm for the bandwidth of k-polygon graphs which has performance ratio $2k^2$ (or $\frac{4}{3}k^2$, in light of Fomin's improvement).

A graph $G = (V, E)$ is a k-polygon graph if it is the intersection graph of chords inside a convex k-polygon, where each chord has its endpoints on two different sides of the polygon. A k-polygon representation, or diagram, for $G = (V, E)$ is a k-sided convex polygon together with a set of chords such that, for all $u, v \in V$, $\{u, v\} \in E$ if and only if the chords corresponding to u and v cross.

Circle graphs are the intersection graphs of chords inside a circle. A circle model, or diagram, for circle graph $G = (V, E)$ is a set of chords in a circle such that two vertices are adjacent in G if and only if their corresponding chords cross. Clearly, every polygon representation of a graph G can also be seen as a circle model of G. Thus, for each $k \geq 2$, every k-polygon graph is a circle graph. (Permutation graphs are to be considered as 2-polygon graphs.)

There is an $O(n^2)$ algorithm [28] which determines whether or not a given graph is a circle graph and, if so, produces a circle model for it. Given a graph $G = (V, E)$, it can be determined in $O(|V|^k)$ time whether or not G is a k-polygon graph and, if so, a polygon representation can be constructed [10]. However, given a circle graph G, the problem of determining the minimum k such that G is a k-polygon graph remains NP-complete [10].

Our algorithm assumes that a k-polygon representation for the input graph is provided.

All of the notation in this section is either identical to that of [22] and [23] or inspired by those two papers. (See also [20].)

Let $G = (V, E)$ be a graph and a, b two nonadjacent vertices of G. The set $S \subseteq V$ is an *a, b-separator* if the removal of S separates a and b in distinct connected components. If no proper subset of S is an a, b-separator, then S is a *minimal a, b-separator*. A *minimal separator* is a set of vertices S that is a minimal a, b-separator.

Lemma 14 (see [9]). *Let S be a minimal a, b-separator of the graph $G = (V, E)$, and let C_a and C_b be the connected components of $G[V \setminus S]$ containing a and b, respectively. Then every vertex of S has at least one neighbor in C_a and at least one neighbor in C_b.*

We denote by $\mathfrak{Sep}(H)$ the set of all minimal separators of a graph H. We shall need the following properties of minimal triangulations of a graph.

Theorem 15 (see [22]). *A triangulation H of a graph G is a minimal triangulation of G if and only if the following three conditions are satisfied:*

1. *If a and b are nonadjacent vertices of H, then every minimal a, b-separator of H is also a minimal a, b-separator of G.*

2. *If S is a minimal separator of H and C a connected component of $H[V \setminus S]$, then the vertex set of C induces a connected component in $G[V \setminus S]$.*
3. *$H = G_{\mathfrak{Sep}(H)}$, where $G_{\mathfrak{Sep}(H)}$ is the graph obtained from G by adding edges between every pair of vertices contained in the same set S for any $S \in \mathfrak{Sep}(H)$.*

To obtain an algorithm to approximate the bandwidth of k-polygon graphs, in a first step we generalize definitions and results of [22] to show that every minimal triangulation H of a k-polygon graph G is a subgraph of G^k.

DEFINITION. *A minimal separator S is* d-good *if, for every nonadjacent pair x and y in S, $d_G(x, y) \leq d$. A triangulation H of G is* d-good *if, for every edge $\{a, b\}$ in H, $d_G(a, b) \leq d$; i.e., H is a subgraph of G^d.*

The following theorem is a consequence of the characterization of minimal triangulations given in Theorem 15.

THEOREM 16. *If every minimal separator of a graph G is d-good, then every minimal triangulation H of G is d-good.*

Proof. Let $\{a, b\}$ be an edge of H but not an edge of G. By Theorem 15, $H = G_{\mathfrak{Sep}(H)}$. Hence there is a minimal separator S of H such that $\{a, b\} \subseteq S$. By Theorem 15, S is also a minimal separator of G. Therefore S is d-good and $d_G(a, b) \leq d$.

Consequently, H is d-good. □

LEMMA 17. *Let G be a graph without a chordless cycle of length greater than $2k + 1$. Then every minimal separator of G is k-good.*

Proof. Assume there is some minimal separator S containing nonadjacent vertices x and y such that $d_G(x, y) > k$. Now, by Lemma 14, we can find an x, y-path in C_x and one in C_y. If we choose shortest such paths, then their union is a chordless cycle of length at least $2(k + 1)$, a contradiction. □

LEMMA 18. *Let G be a k-polygon graph. Then G has no chordless cycle of length greater than $2k$.*

Proof. It is proved in [14] that chordless cycles have unique representations as chords in a circle. Suppose G has a chordless cycle of length at least $2k+1$ and consider the unique representation as chords in a circle. The number of chord endpoints must be at least $2(2k + 1) = 4k + 2$. Each side of the k-polygon can contain at most four chord endpoints; otherwise, the two endpoints of a chord would have to be on the same side. Thus there must be at least $\lceil \frac{4k+2}{4} \rceil = k + 1$ sides. □

THEOREM 19. *Every minimal triangulation H of a k-polygon graph G is k-good, and thus H is a subgraph of G^k.*

Proof. By Lemma 17 and 18, every minimal separator of a k-polygon graph G is k-good. Thus, by Theorem 16, every minimal triangulation of G is k-good. □

In a second step we use a representation theorem for the minimal triangulations of a circle graph given in [23] to obtain a similar theorem for k-polygon graphs. We shall need some preparations.

Assume that an n-vertex circle graph is given as a set of chords in a circle. Between each two consecutive endpoints of chords, add a point called a *scanpoint.* Let Z be the set of $2n$ scanpoints. A *scanline* is a chord of the circle connecting two scanpoints. Let c_1 and c_2 be two chords of the circle model. A scanline s is *between* c_1 and c_2 if every path from an endpoint of c_1 to an endpoint of c_2 along the circle passes through a scanpoint of s. For any scanline s, we denote by $S(s)$ the set of all vertices v of G for which the corresponding chord intersects s.

THEOREM 20 (see [21]). *Let a and b be nonadjacent vertices of the circle graph $G = (V, E)$. For every minimal a, b-separator S of G, there exists a scanline s between* *the chords of a and b such that $S = S(s)$.*

Note that this implies that, for every minimal a, b-separator S of a k-polygon graph G, there is a scanline s with $S = S(s)$ such that the endpoints of s are on two different sides of the polygon.

In [23] Kloks, Kratsch, and Wong give the following representation theorem for all minimal triangulations of a circle graph in terms of planar triangulations of the polygon $\mathcal{P}(Z)$, which is the convex polygon with vertex set Z.

THEOREM 21 (see [23]). *Let $G = (V, E)$ be a circle graph given as a set of chords in a circle, and let Z be the corresponding set of scanpoints. Then for every minimal triangulation H of G there is a planar triangulation T of the polygon $\mathcal{P}(Z)$ such that $H = H(T)$, where $H(T)$ is the graph with vertex set V, and vertices u and v are adjacent in $H(T)$ if there exists a triangle in T that is intersected by the chords corresponding to u and v.*

Let $G = (V, E)$ be a k-polygon graph and thus a circle graph. Consider a k-polygon representation of G consisting of a set of chords C inside a k-sided polygon $\mathcal{P}_G$. Let Z be the set of scanpoints on $\mathcal{P}_G$, and let $\mathcal{P}(Z)$ be the convex polygon with vertex set Z.

THEOREM 22. *Let $G = (V, E)$ be a k-polygon graph given as a set of chords in a k-polygon, and let Z be the corresponding set of scanpoints. Then for every minimal triangulation H of G there is a planar triangulation T of the polygon $\mathcal{P}(Z)$ such that*

- *every diagonal in T has endpoints on two different sides of the k-polygon, and*
- *$H = H(T)$, where $H(T)$ is the graph with vertex set V, and vertices u and v are adjacent in $H(T)$ if there exists a triangle Q in T that is intersected by the chords corresponding to u and v.*

Proof. Theorem 22 is an immediate consequence of Theorem 21, except for the property that no diagonal of the planar triangulation T of $\mathcal{P}(Z)$ has both its endpoints on one side of $\mathcal{P}_G$. We sketch only how to construct such a planar triangulation T following the lines of the proof of Theorem 21.

First, for each minimal separator S of H we choose a scanline s such that $S = S(s)$, and this can be done such that no two scanlines cross each other. As mentioned below Theorem 20, none of these scanlines has both endpoints on one side of $\mathcal{P}_G$. Now we choose all these scanlines as diagonals of a triangulation of $\mathcal{P}(Z)$. If this is not yet a triangulation T of $\mathcal{P}(Z)$ we add more diagonals to obtain a planar triangulation such that we never add a diagonal with both endpoints on one side of $\mathcal{P}_G$. □

Consequently, a minimum triangulation H of a k-polygon graph G can be computed by finding a minimum weight triangulation of $\mathcal{P}(Z)$, in which we consider only chords with endpoints on different sides of the polygon. The $O(n^3)$ dynamic programming algorithm for this computation for circle graphs [23] can be adapted to the domain of k-polygon graphs; the adapted algorithm retains its $O(n^3)$ complexity.

It remains to show how to construct a clique tree with at most k leaves for H. This is done by using the planar triangulation T of $\mathcal{P}(Z)$ with $H = H(T)$ for the minimum triangulation H, which is also provided by the algorithm computing H. Now a clique tree of H is constructed as follows. Take the dual graph of the planar triangulation T (without taking a vertex for the exterior face); i.e., each vertex of the dual graph corresponds to a triangle of T. It is well known that this dual graph of a planar triangulation is a tree. Two vertices of the tree are adjacent if and only if the corresponding triangles of T share a diagonal, and we assign to each vertex of the tree the set of all chords intersecting the corresponding triangle of T. This tree has at most k leaves, since any leaf corresponds to a triangle containing a corner of $\mathcal{P}_G$.

Finally, we remove all nonmaximal cliques by contracting suitable edges of the tree and obtain a clique tree of H with at most k leaves.

THEOREM 23. *There is an $O(n^3)$ algorithm to compute for a given k-polygon graph G a clique tree of a minimum triangulation such that this clique tree has at most k leaves.*

Thus, our approximation algorithm from the previous section applies to this triangulation H of a k-polygon graph G.

Remark. One can show analogously that there is an $O(n^3)$ algorithm that computes for a given minimal triangulation H of a k-polygon graph a clique tree with at most k leaves.

Finally, we combine the main results of this section to obtain an algorithm to approximate the bandwidth of k-polygon graphs.

THEOREM 24. *There is an $O(n^3)$ algorithm to compute for a k-polygon graph G given with a k-polygon representation a layout L satisfying $b(G, L) \leq 2k^2 \cdot bw(G)$.*

Proof. By Theorem 23, there is an $O(n^3)$ algorithm to compute for a k-polygon graph G given with a k-polygon representation, a minimum triangulation H of G, and a clique tree of H such that this clique tree has at most k leaves. By Theorem 13, a layout L satisfying $b(H, L) \leq 2k \cdot bw(H)$ can be computed by a $O(k(n+m))$ algorithm. By Lemma 2, any layout L of a d-good triangulation H of G with $b(H, L) \leq c \cdot bw(H)$ fulfills $b(G, L) \leq d \cdot c \cdot bw(G)$, where $c, d \geq 1$ are constants. Consequently, $b(G, L) \leq 2k^2 \cdot bw(G)$. $\square$

Remark. By [13], the performance ratio in the previous theorem can be improved to $b(G, L) \leq \frac{4}{3}k^2 \cdot bw(G)$.

REFERENCES

[1] K. Ando, A. Kaneko, and S. Gervacio, *The bandwidth of a tree with k leaves is at most $\lceil \frac{k}{2} \rceil$*, Discrete Math., 150 (1996), pp. 403–406.

[2] S. F. Assmann, G. W. Peck, M. M. Sysło, and J. Zak, *The bandwidth of caterpillars with hairs of length* 1 *and* 2, SIAM J. Alg. Disc. Meth., 2 (1981), pp. 387–393.

[3] J. R. S. Blair and B. Peyton, *An introduction to chordal graphs and clique trees*, in Graph Theory and Sparse Matrix Computation, IMA Vol. Math. Appl. 56, A. George, J. R. Gilbert, and J. W. H. Liu, eds., Springer, New York, 1993, pp. 1–29.

[4] K. S. Booth and G. S. Lueker, *Testing for the consecutive ones property, interval graphs, and graph planarity using PQ-tree algorithms*, J. Comput. System Sci., 13 (1976), pp. 335–379.

[5] P. Z. Chinn, J. Chvátalová, A. K. Dewdney, and N. E. Gibbs, *The bandwidth problem for graphs and matrices—a survey*, J. Graph Theory, 6 (1982), pp. 223–254.

[6] V. Chvátal, *A remark on a problem of Harary*, Czech Math. J., 20 (1970), pp. 109–111.

[7] J. Chvátalová, A. K. Dewdney, N. E. Gibbs, and R. R. Korfhage, *The Bandwidth Problem for Graphs: A Collection of Recent Results*, Research report #24, Department of Computer Science, University of Western Ontario, London, ON, Canada, 1975.

[8] D. G. Corneil, S. Olariu, and L. Stewart, *The ultimate interval graph algorithm?*, in Proceedings of the Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, San Francisco, CA, 1998, ACM, New York, 1998, pp. 175–180.

[9] G. A. Dirac, *On rigid circuit graphs*, Abh. Math. Sem. Univ. Hamburg, 25 (1961), pp. 71–76.

[10] E. S. Elmallah and L. K. Stewart, *Polygon graph recognition*, J. Algorithms, 26 (1998), pp. 101–140.

[11] E. M. Eschen and J. P. Spinrad, *An $O(n^2)$ algorithm for circular-arc graph recognition*, in Proceedings of the Fourth Annual ACM-SIAM Symposium on Discrete Algorithms, Austin, TX, 1993, ACM, New York, 1993, pp. 128–137.

[12] U. Feige, *Approximating the bandwidth via volume respecting embeddings*, J. Comput. System Sci., 60 (2000), pp. 510–539.

[13] F. Fomin, *private communication*, St. Petersburg, Russia, 1999.

[14] C. P. Gabor, K. J. Supowit, and W.-L. Hsu, *Recognizing circle graphs in polynomial time*, J. Assoc. Comput. Mach., 36 (1989), pp. 435–473.

[15] M. R. Garey, R. L. Graham, D. S. Johnson, and D. E. Knuth, *Complexity results for bandwidth minimization*, SIAM J. Appl. Math., 34 (1978), pp. 477–495.

[16] F. Gavril, *The intersection graphs of subtrees in a tree are exactly the chordal graphs*, J. Combinatorial Theory Ser. B, 16 (1974), pp. 47–56.

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

[18] S. Jiang, *The Bandwidth Problem and Bandwidth of Cographs*, manuscript, 1992.

[19] D. J. Kleitman and R. V. Vohra, *Computing the bandwidth of interval graphs*, SIAM J. Discrete Math., 3 (1990), pp. 373–375.

[20] T. Kloks, *Treewidth—Computations and Approximations*, Lecture Notes in Comput. Sci. 842, Springer, Berlin, 1994.

[21] T. Kloks, *Treewidth of circle graphs*, Internat. J. Found. Comput. Sci., 7 (1996), pp. 111–120.

[22] T. Kloks, D. Kratsch, and H. Müller, *Approximating the bandwidth for asteroidal triple-free graphs*, J. Algorithms, 32 (1999), pp. 41–57.

[23] T. Kloks, D. Kratsch, and C. K. Wong, *Minimum fill-in on circle and circular-arc graphs*, J. Algorithms, 28 (1998), pp. 272–289.

[24] T. Kloks and R.B. Tan, *Bandwidth and topological bandwidth of graphs with few P_4's*, Discrete Appl. Math., 115 (2001), pp. 117–133.

[25] R. Mahesh, C. P. Rangan, and A. Srinivasan, *On finding the minimum bandwidth of interval graphs*, Inform. and Comput., 95 (1991), pp. 218–224.

[26] B. Monien, *The bandwidth minimization problem for caterpillars with hair length* 3 *is NP-complete*, SIAM J. Alg. Disc. Meth., 7 (1986), pp. 505–512.

[27] C. H. Papadimitriou, *The NP-completeness of the bandwidth minimization problem*, Computing, 16 (1976), pp. 263–270.

[28] J. Spinrad, *Recognition of circle graphs*, J. Algorithms, 16 (1994), pp. 264–282.

[29] A. P. Sprague, *An $O(n \log n)$ algorithm for bandwidth of interval graphs*, SIAM J. Discrete Math., 7 (1994), pp. 213–220.

[30] W. Unger, *The complexity of the approximation of the bandwidth problem*, in Proceedings of the Thirty-Ninth Annual IEEE Symposium on Foundations of Computer Science, Palo Alto, CA, 1998, pp. 82–91.