Approximating the minimum clique cover and other hard problems in subtree filament graphs

J. Mark Keil[*] Lorna Stewart[†]

March 20, 2006

Abstract

Subtree filament graphs are the intersection graphs of subtree filaments in a tree. This class of graphs contains subtree overlap graphs, interval filament graphs, chordal graphs, circle graphs, circular-arc graphs, cocomparability graphs, and polygon-circle graphs. In this paper we show that, for circle graphs, the clique cover problem is NP-complete and the h-clique cover problem for fixed h is solvable in polynomial time. We then present a general scheme for developing approximation algorithms for subtree filament graphs, and give approximation algorithms developed from the scheme for the following problems which are NP-complete on circle graphs and therefore on subtree filament graphs: clique cover, vertex colouring, maximum k-colourable subgraph, and maximum h-coverable subgraph.

Key Words: subtree filament graph, circle graph, clique cover, NP-complete, approximation algorithm.

1 Introduction

Subtree filament graphs were defined in [10] to be the intersection graphs of subtree filaments in a tree, as follows. Consider a tree $T = (V_T, E_T)$ and a multiset of $n \geq 1$ subtrees $\{T_i \mid 1 \leq i \leq n\}$ of T. Let T be embedded in a plane P, and consider the surface S that is perpendicular to P, such that the intersection of S with P is T. A subtree filament corresponding to subtree T_i of T is a connected curve in S, above P, connecting all the leaves of T_i, such that, for all $1 \leq i \leq n$:

- if $T_i \cap T_j = \emptyset$ then the filaments corresponding to T_i and T_j do not intersect, and

[*]Department of Computer Science, University of Saskatchewan, Saskatoon, Saskatchewan, Canada, S7N 5C9, keil@cs.usask.ca, 306-966-4894.

[†]Department of Computing Science, University of Alberta, Edmonton, Alberta, Canada, T6G 2E8, stewart@cs.ualberta.ca, 780-492-2982.

- if $T_i \cap T_j \neq \emptyset$, $T_i \not\subseteq T_j$, and $T_j \not\subseteq T_i$ then the filaments corresponding to T_i and T_j do intersect, and
- if $T_i \subseteq T_j$ or $T_j \subseteq T_i$ then the filaments corresponding to T_i and T_j may or may not intersect.

Note that, if all filaments corresponding to containment do intersect, then the subtree filament graph is the intersection graph of $\{T_i \mid 1 \leq i \leq n\}$; if none of the filaments corresponding to containment intersect, then the subtree filament graph is the overlap graph of $\{T_i \mid 1 \leq i \leq n\}$.

For graph $G = (V, E)$, we shall use n to denote $|V|$ and m to denote $|E|$.

A graph $G = (V, E)$, is called a circle graph if there is a one-to-one correspondence between V and a set C of chords of a circle such that two vertices are adjacent if and only if the corresponding chords intersect. C is called the chord intersection model for G. Circle graphs are equivalent to the overlap graphs of intervals [8]. An interval model of a circle graph is a set of n closed intervals in the real line such that interval i overlaps interval j (interval i intersects interval j but neither contains the other) if and only if vertex i is adjacent to vertex j. Without loss of generality, we may assume that the endpoints of the intervals are distinct. Spinrad [21] has an $O(n^2)$ time algorithm which given a graph will determine whether or not it is a circle graph and if it is will produce a chord intersection model for it. Given a chord intersection model, an interval model can be obtained in $O(n)$ time [8]. In the following we will make use of the interval model and interchangeably refer to the ith vertex or the ith interval as being adjacent to or overlapping with vertex j or interval j.

A graph $G = (V, E)$ is a comparability graph is there exists an orientation of E that is transitive. Cocomparability graphs are the complements of comparability graphs. Circle graphs and cocomparability graphs are subclasses of subtree filament graphs [10].

The problem of finding a maximum clique in a circle graph can be solved in polynomial time [8], as can the maximum independent set problem [8]. In fact, these algorithms can be extended to solve the weighted versions of the maximum clique and maximum independent set problems for subtree filament graphs [10]. However, the colouring problem [7], the problem of finding a maximum k-colourable subgraph [4] and the dominating set problem [18] remain NP-complete when restricted to circle graphs.

The clique cover problem for a graph is the problem of partitioning the vertex set into the minimum number of subsets such that the subgraph induced by each subset is a clique. The clique cover problem was one of the first problems shown to be NP-complete by Karp [17]. However for the large class of perfect graphs the ellipsoid algorithm based method of Grötschel, Lovász and Schrijver [13] provides a polynomial solution for the clique cover problem. For circular-arc graphs, Hsu and Tsai [15] give a linear time algorithm for the clique cover problem.

The computational complexity of the clique cover problem on circle graphs was mentioned as open in 1980 by Golumbic [12], and again in 1985 by Johnson in his NP-completeness column [16]. More recently Spinrad [22] again raises it as an open problem in his book. The next section contains a proof that the clique cover problem on circle graphs is NP-complete. We also show that the fixed parameter version, where the number of cliques in a cover is bounded by a constant, is solvable in polynomial time.

In Section 3, we outline a general divide-and-conquer scheme for developing approximation algorithms for subtree filament graphs, and provide some example algorithms developed using the scheme. Included among these is an approximation algorithm for the clique cover problem.

2 NP-completeness of clique cover on circle graphs

The decision problem formulation of the clique cover problem on circle graphs is as follows: given a circle graph $G = (V, E)$ and a positive integer k, is there a clique cover of size k or less for G, that is, a partition of V into k disjoint subsets $V_1, V_2, \ldots V_k$ such that, for $1 \leq i \leq k$, the subgraph induced by V_i is a complete graph?

Theorem 2.1 *The clique cover problem on circle graphs is NP-complete.*

Proof. It is not hard to see that the problem is in NP. To show that the problem is NP-hard we construct a polynomially computable reduction from a restriction of the SATISFIABILITY problem where there are 3 literals per clause and each variable occurs in at most 4 clauses (3-4 SAT). 3-4 SAT was shown to be NP-complete by Tovey [23].

Let $\mathcal{F}$ be an arbitrary instance of the 3-4 SAT problem. Let $X = \{x_1, x_2, \ldots, x_n\}$ be the set of boolean variables and $C = \{c_1, c_2, \ldots, c_m\}$ be the set of clauses in $\mathcal{F}$. We shall construct, in polynomial time, a set I of intervals in the real line (with integer endpoints) and an integer k such that the overlap graph of the intervals in I has a clique cover of cardinality k if and only if $\mathcal{F}$ is satisfiable.

When we create I, we will create 12 intervals associated with each variable, x_i, these are the intervals $B_i^l, l = 1, 2, \ldots 12$. For $l = 1 \ldots 6, B_i^l = [z_i + 10l, z_i + 10l + 5]$ where $z_i = 100(i - 1)$. The purpose of the offset z_i is to ensure the intervals associated with different variables are disjoint. The six remaining intervals associated with variable x_i are $B_i^7 = [z_i{+}13, z_i{+}22], B_i^8 = [z_i{+}23, z_i{+}42], B_i^9 = [z_i{+}43, z_i{+}64], B_i^{10} = [z_i{+}11, z_i{+}32], B_i^{11} = [z_i + 33, z_i + 52]$, and $B_i^{12} = [z_i + 53, z_i + 62]$.

Since the first six intervals associated with a variable are disjoint, any clique cover of the overlap graph of the 12 intervals associated with a variable must consist of at least six cliques. In fact there are exactly two ways to cover this graph with six cliques. We will call one of the two clique covers T and the other F and then associate the two clique covers with the truth value of the associated variable. Each of the cliques in either covering is of size two.

In the overlap representation of the circle graph a clique can be represented as the portion of the real line common to all the elements in the clique. Using this idea we can represent a clique cover as a set of segments on the real line. We do this in figure 1 to indicate the two clique covers of size six. For example in clique cover T, the first clique consists of B_i^1 and B_i^{10}.

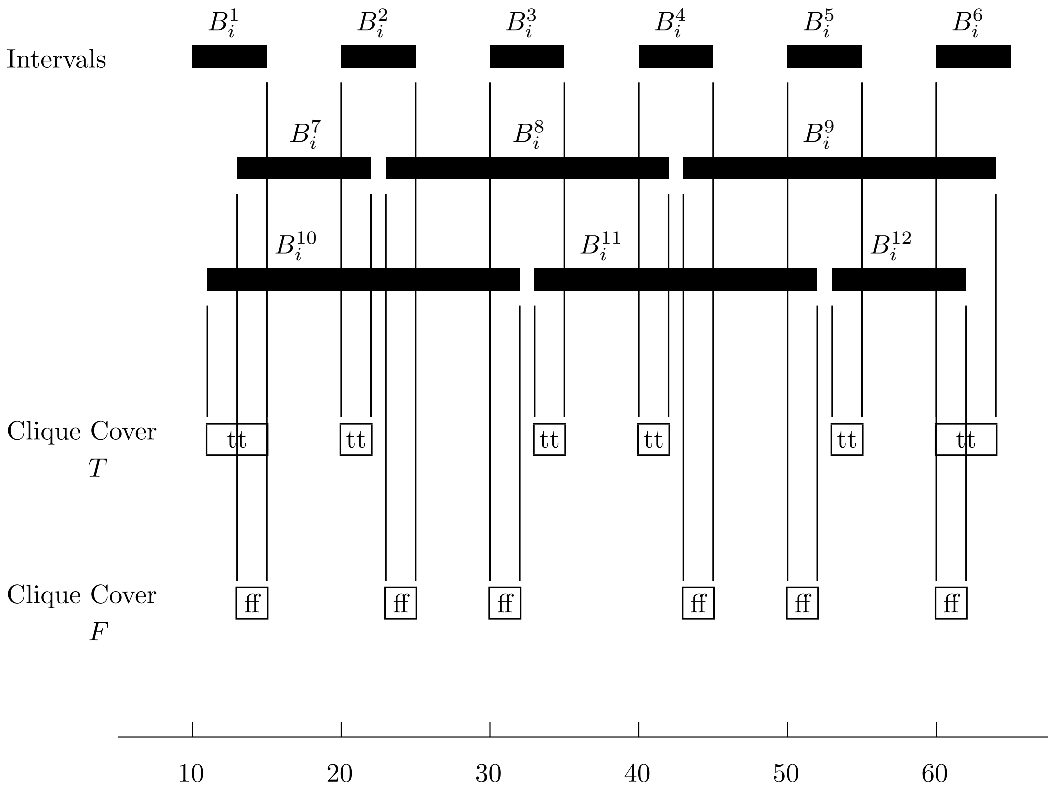

Figure 1: The twelve intervals associated with variable x_i and the two clique covers of size six for these intervals.

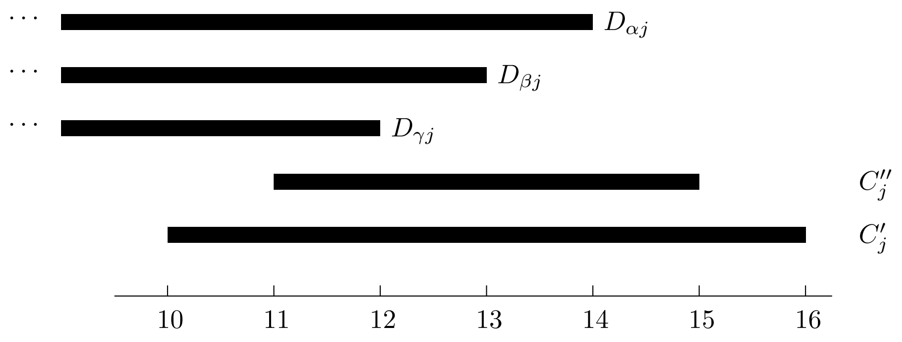

Figure 2: C'_j and C''_j are associated with clause c_j, and $D_{\alpha j}$, $D_{\beta j}$ and $D_{\gamma j}$ connect variables x_α, x_β and x_γ, respectively, with clause c_j.

We also create two intervals C'_j and C''_j for each clause $c_j \in C$, $j = 1, \dots m$ with coordinates as follows: $C'_j = [y_j, y_j + 6]$ and $C''_j = [y_j + 1, y_j + 5]$ where $y_j = 100n + 10j$. See figure 2.

Since C'_j and C''_j do not overlap, two cliques are needed in any clique cover to include both.

In the overall layout of the intervals, the intervals associated with the variables occur left of the intervals associated with the clauses. The z_i and y_j offsets ensure that the intervals associated with each variable and clause are disjoint.

We now need a way to associate literals with the clauses containing them. Of the six cliques in the T and F clique covers of the subgraph associated with a variable x_i, we will ignore the leftmost and rightmost and allow extension of the middle four cliques to provide a way to communicate information to the clauses. Notice that the intervals representing the middle four cliques in the T clique cover are disjoint from the intervals representing the middle four cliques in the F clique cover.

If variable x_i is contained as an unnegated literal in clause c_j, we will introduce an interval D_{ij}, that overlaps every interval in one of the interior cliques in the T clique cover of the overlap graph of the intervals associated with the variable x_i, and also overlaps with the two intervals associated with clause c_j. Similarly, if c_j contains $\overline{x}_i$, then D_{ij} will overlap with every interval in one of the interior cliques of the F clique cover of the overlap graph of the intervals associated with x_i and also D_{ij} will overlap with the two intervals associated with clause c_j.

We also require that if x_i is involved in a literal in four different clauses, c_{j_1}, c_{j_2}, c_{j_3} and c_{j_4} with $j_1 < j_2 < j_3 < j_4$, that interval D_{ij_4} contains D_{ij_3} which contains D_{ij_2} which contains D_{ij_1}. Figure 3 illustrates an example in which literal $\overline{x}_i \in c_2, x_i \in c_3, \overline{x}_i \in c_5$ and $x_i \in c_7$.

If clause c_j contains literals involving variables α, β, and γ, with $\alpha < \beta < \gamma$ then we ensure that $D_{\alpha j}$ contains $D_{\beta j}$ which contains $D_{\gamma j}$ by setting the right endpoint of $D_{\alpha j}$ to $y_j + 4$, the right endpoint of $D_{\beta j}$ to $y_j + 3$, and the right endpoint of $D_{\gamma j}$ to $y_j + 2$. See figure

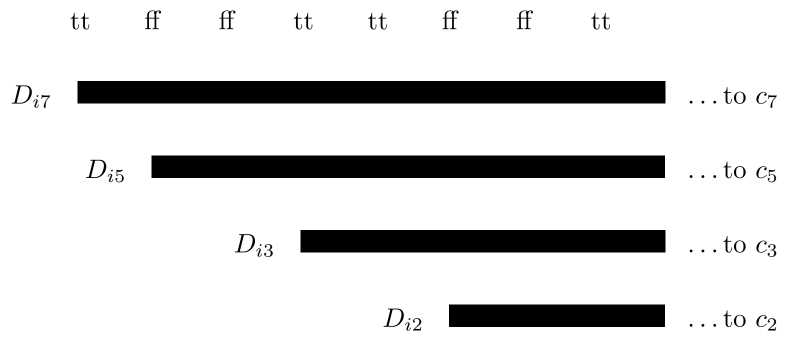

Figure 3: The interior cliques of the T and F clique covers for the overlap graph of the intervals associated with x_i. Also shown are the intervals D_{ij}.

2.

This completes the construction of the $12n + 2m + 3m$ intervals. Let G be the overlap graph of the constructed intervals. Let $k = 6n + 2m$.

Claim 2.2 *G has a clique cover of size k if and only if formula $\mathcal{F}$ is satisfiable.*

Proof of Claim. Given a satisfying truth assignment A for $\mathcal{F}$ we show how to construct a clique cover of size k for G.

If x_i is true in A, we include the T clique cover for intervals associated with x_i. Note that an interval D_{ij} can also be included in these six cliques if x_i appears unnegated in clause c_j. Similarly, if x_i is false in A, we include the F clique cover for the intervals associated with x_i. Note that an interval D_{ij} can also be included in these six cliques if x_i appears negated in clause c_j. Including either the T or F clique cover for the intervals associated with each variable will contribute a total of $6n$ cliques to the partial clique cover of G. It remains to cover the two intervals associated with each clause and any intervals D_{ij} which are not covered on the variable side.

For each clause c_j we add two more cliques to the clique cover. The first includes interval C'_j plus one of the D_{ij} not previously covered. The second includes C''_j plus another of the D_{ij} not previously covered (if necessary). Note that, since A is a satisfying truth assignment, at least one of the intervals D_{ij} for clause c_j is covered on the variable side. The total size of the clique cover is $6n + 2m = k$ as required.

To complete the proof of the claim we show how to produce a satisfying truth assignment A for $\mathcal{F}$ from a clique cover for G of size k. For each clause c_j, since interval C''_j contains interval C'_j, the clique cover for G must have one clique K'_j which contains C'_j and another clique K''_j which contains C''_j. The only other intervals that K'_j and K''_j can contain are the three intervals $D_{\alpha j}$, $D_{\beta j}$, and $D_{\gamma j}$. But no two of these three intervals overlap, thus K'_j and K''_j are of size two at most and do not contain any of the intervals associated with the variables. The clique cover thus contains $k - 2m = 6n$ cliques to contain the intervals associated with the variables. Since the intervals associated with a single variable require six cliques to cover them, and since this can be done in only two ways, the clique cover for G must include either the T clique cover or the F clique cover for the intervals associated with each variable x_i. If the T clique cover is used to cover the intervals associated with x_i, we set x_i to true and if the F clique cover is used to cover the intervals associated with x_i, we set x_i to false in A.

If the T (F) clique cover is used for the overlap graph of the intervals associated with variable x_i, we can include in the six cliques also any intervals D_{ij} for which x_i appears unnegated (negated) in c_j. Thus the use of the T (F) clique cover for the overlap graph of the intervals associated with x_i ensures the satisfiability of any clause c_j containing x_i unnegated (negated). Since the clique cover for G covers all vertices in G and since for each clause c_j at least one of the intervals $D_{\alpha j}$, $D_{\beta j}$, and $D_{\gamma j}$ must be included in a clique on the variable side, the derived truth assignment A must satisfy every clause as required.

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

In light of the NP-completeness result, we consider two fixed parameter versions of the problem. First, we observe that the proof of the preceding theorem implies that the clique cover problem for circle graphs, where each clique in the cover is restricted to be of size at most three, is NP-complete.

Second, the following algorithm demonstrates that the clique cover problem, where we restrict the number of cliques in the cover to be at most h, can be solved in polynomial time for any fixed constant h. This is in contrast to the complementary k-vertex colouring problem for circle graphs, the decision version of which was shown to be NP-complete for $k \geq 4$ [24] and in P for $k = 3$ [25].

The algorithm makes use of an interval overlap model for n-vertex circle graph $G = (V, E)$. Interval endpoints are assumed to be distinct and intervals are indexed in left-to-right order of right endpoints. For $1 \leq i \leq n$, interval I_i has left endpoint L_i and right endpoint R_i, and corresponds to vertex $v_i \in V$.

For $1 \leq i \leq n$, let G_i be the subgraph of G induced by $\{v_1, v_2, \ldots v_i\}$ and let $\mathcal{C}_i$ be the set of clique covers of size at most h for G_i.

Intervals corresponding to vertices of a clique share the portion of the real line from the largest left endpoint of an interval in the clique to the smallest right endpoint of an interval in the clique. Therefore, because of the order in which vertices are visited, vertex v_i can be added to a clique C in some clique cover of $\mathcal{C}_{i-1}$ to obtain a clique cover of $\mathcal{C}_i$ if and only if L_i is in the portion of the real line shared by all of the intervals of C. Thus, at the beginning of stage i of the algorithm, it suffices to know $\mathcal{I}_{i-1}$, the set of sets of shared portions of the real line corresponding to $\mathcal{C}_{i-1}$.

- Initially, $\mathcal{I}_1$ contains one interval (L_1, R_1) corresponding to the single clique $\{v_1\}$ and, for all $2 \leq i \leq n$, $\mathcal{I}_i$ is empty.

- For $i = 2$ to n:

  - For each set of intervals X in $\mathcal{I}_{i-1}$:

  - $*$ For each interval (L, R) in X:
    - $\cdot$ If $L < L_i < R$ then replace (L, R) in X with (L_i, R) and add the resulting set of intervals to $\mathcal{I}_i$.
  - $*$ If $|X| < h$ then add the new interval (L_i, R_i) to X and add the resulting set of intervals to $\mathcal{I}_i$.

- G is h-coverable if and only if $\mathcal{I}_n \neq \emptyset$.

By an inductive argument, it can be seen that $\mathcal{I}_i$ contains exactly the sets of intervals corresponding to $\mathcal{C}_i$. Clearly, $\mathcal{I}_1$ correctly represents $\mathcal{C}_1$; suppose $\mathcal{I}_{i-1}$ contains exactly the sets of intervals corresponding to $\mathcal{C}_{i-1}$. Then each set in $\mathcal{I}_i$ corresponds to a clique cover of $\mathcal{C}_i$, since all vertices are covered by each set, and since v_i is implicitly added only to cliques with which it forms a clique. In addition, since any clique cover Y for G_i, with vertex v_i removed, is a clique cover for G_{i-1} and thus is represented by a set of intervals in $\mathcal{I}_{i-1}$, it must be that intervals representing Y appear in $\mathcal{I}_i$.

There are fewer than hn^{2h} elements of the $\mathcal{I}_i$ sets. Each element is considered at most once and the algorithm performs $O(h)$ steps for each element considered. Therefore, the algorithm executes in $O(h^2n^{2h}) = O(n^{2h})$ time.

3 Approximation algorithms for subtree filament graphs

Since there is no known polynomial time recognition algorithm for subtree filament graphs, we assume that, as input to our approximation algorithms, we are given the adjacency list representation of subtree filament graph $G = (V_G, E_G)$, along with a tree $T = (V_T, E_T)$ and subtrees $\{T_i \mid 1 \leq i \leq |V_G|\}$ of T such that there exist subtree filaments corresponding to $\{T_i \mid 1 \leq i \leq |V_G|\}$ of which the intersection graph is G. Like the algorithms of [11], our algorithms do not require explicit representation of the filaments. We assume that every vertex of T appears in some T_i, $1 \leq i \leq |V_G|$, and that $\sum_{i=1}^{|V_G|} |T_i|$ is bounded above by a polynomial in $|V_G|$, $P(|V_G|)$.

Alternatively, the input to our algorithms could be a subtree filament graph G and a chordal graph GI with the property that, for any clique tree of GI, the corresponding subtree intersection representation for GI consisting of $T = (V_T, E_T)$ and set of subtrees $\{T_i \mid 1 \leq i \leq |V_G|\}$ of T, there exist subtree filaments corresponding to $\{T_i \mid 1 \leq i \leq |V_G|\}$ of which the intersection graph is G. The existence of such a chordal graph for any subtree filament graph follows from Theorem 4 of [10]. In this case, T and $\{T_i \mid 1 \leq i \leq |V_G|\}$ can be constructed from GI in $O(|V_{GI}| + |E_{GI}|)$ time and $P(|V_G|) \in O(|V_G|^2)$ (see [12]).

As observed by Gavril [10], for all $x \in V_T$, the subgraph of G induced by $\{v_i \in V_G \mid x \in T_i\}$ is a cocomparability graph. This follows from the observation that each nonedge of such a subgraph corresponds to two subtrees, one of which is contained in the other, and this relation is transitive.

The general divide-and-conquer scheme for an approximation algorithm on subtree filament graph G proceeds as follows. If G is small the problem is solved exactly using a brute force algorithm. Otherwise, we partition G into a cocomparability graph and a number of induced subgraphs, each of which is a subtree filament graph having at most $|V_G|/2$ vertices. The algorithm combines an exact solution for the problem on the cocomparability graph with recursively computed approximate solutions for the smaller induced subtree filament subgraphs.

We now describe how to accomplish, in polynomial time, a suitable partition for G. We seek a point $x \in V_T$ such that, for all subtrees H of $T \backslash \{x\}$:

$$|\{v_i \in V_G \mid T_i \text{ is entirely contained in } H\}| \ \leq \ |V_G|/2.$$

We visit the vertices of T, beginning at the leaves, visiting a vertex only when all, or all but one, of its neighbours have been visited. For each vertex v of T, we compute $S(v)$, the number of subtrees of $\{T_i \mid 1 \leq i \leq |V_G|\}$ that are entirely contained in the subtree of T consisting of v and all subtrees of $T \backslash \{v\}$ whose vertices have been previously visited, as follows. If v is a leaf in T then $S(v)$ is the number of elements of $\{T_i \mid 1 \leq i \leq |V_G|\}$ that consist of the single vertex v; if v is the last vertex of T to be visited then $S(v) = |V_G|$. In all other cases, $S(v)$ is the sum of the S values of its previously visited neighbours plus $|\{T_i \mid v \in T_i \text{ and all other vertices of } T_i \text{ have been visited}\}|$.

Let x be the first vertex encountered with $S(x) > |V_G|/2$. Each subtree of $T \backslash \{x\}$ entirely contains at most $|V_G|/2$ elements of $\{T_i \mid 1 \leq i \leq |V_G|\}$ since $S(x) > |V_G|/2$ and since x is the first such vertex to be visited.

Since all S values can be computed in time linear in $|V_T| + \sum_{i=1}^{|V_G|} |T_i|$, this process finds vertex $x \in V_T$ with the desired property in $O(|V_T| + \sum_{i=1}^{|V_G|} |T_i|) \ = \ O(P(|V_G|)$ time, which is linear in the size of the input, and polynomial in the size of the subtree filament graph.

Now, the approximation algorithm computes the exact solution for the cocomparability graph, GC, the subgraph of G induced by $\{v_i \in V_G \mid x \in T_i\}$, and combines it with the approximate solutions for each of the subgraphs G_j of G induced by $\{v_i \in V_G \mid T_i \text{ is entirely contained in } H_j\}$ for each subtree H_j of $T \backslash \{x\}$, for all $1 \leq j \leq l$, where $T \backslash \{x\}$ is a forest consisting of l trees. The fact that each G_j has size at most $|V_G|/2$ will be important in the analysis of the approximation algorithms.

Let Opt (respectively $OptC$, Opt_j) be the size of an optimal solution to the problem in G (respectively, GC, G_j).

This scheme will produce polynomial time approximation algorithms for problems with certain properties. First we require that there exists a polynomial time exact algorithm for the problem on cocomparability graphs. It is also necessary that there is a polynomial time computable method for combining the subsolutions into a good overall solution. As part of this, for minimization problems we require that $Opt \geq OptC$ and $Opt \geq Opt_j$ for all j, $1 \leq j \leq l$. For maximization problems we require that $Opt \leq OptC + \sum_{j=1}^{l} Opt_j$.

In the special case that the input to our algorithms is restricted to circle graphs, we make use of an interval overlap representation in which the interval endpoints are located at the integers from 1 to $2n$. If we consider the set, S, of intervals containing a particular point p, then the induced subgraph GS of G formed with the vertices corresponding to the intervals in the set S is a permutation graph. To see this note that the only pairs of vertices in GS that are not adjacent correspond to a pair of intervals where one contains the other. Thus $\overline{GS}$, the complement of GS, is an interval containment graph. It has long been known that a graph is an interval containment graph if and only if it is a permutation graph [5], thus $\overline{GS}$ is a permutation graph. That GS is a permutation graph then follows from the fact that the complement of a permutation graph is also a permutation graph [12].

The general divide-and-conquer scheme for the approximation algorithm on a circle graph $G = (V, E)$ proceeds as above, with the difference that we partition V into only three parts: all vertices which correspond to intervals containing the point $q = n + 0.5$ are placed in a set C, all vertices corresponding to intervals completely to the left of q are placed in a set L, and all vertices corresponding to intervals completely to the right of q are placed in a set R. We have $|L| \leq |V_G|/2$ and $|R| \leq |V_G|/2$, and we know that the subgraph induced by the vertices in C, GC, is a permutation graph. The algorithm combines an exact solution for the problem on GC with recursively computed approximate solutions to the problem in the subgraphs induced by L and R, GL and GR, respectively. In this case, our algorithms may be considered to be robust by virtue of the polynomial time recognition algorithm of [21]. In addition, we may obtain faster approximation algorithms in this case because the partitioning can be done in linear time in the size of the input graph, and since the exact algorithms on permutation graphs may be faster than those available for the larger class of cocomparability graphs.

We note that the general divide-and-conquer approximation approach may be applied to other graph classes. Indeed, in the case of boxicity two graphs, the approach has been used to achieve $O(\log n)$ approximation algorithms for the maximum independent set problem [1] and the clique cover problem [19].

The decision versions of all of the problems that we consider are known to be NP-complete on circle graphs and therefore on subtree filament graphs, and polynomially solvable on cocomparability graphs.

3.1 The clique cover problem

As our first example we show how the divide-and-conquer scheme can produce a $\log n$-approximation algorithm for the clique cover problem on subtree filament graphs. The NP-completeness of the decision version of this problem on circle graphs was shown in Section 2. For general graphs the problem is not approximable within $|V|^{\frac{1}{7}-\epsilon}$ for any $\epsilon > 0$ [2].

Since GC is a cocomparability graph, we have $OptC$ equal to the size of a maximum independent set in GC. Thus, since any independent set in GC is also an independent set in G, $Opt \geq OptC$. Also since no clique can contain vertices of G_i and G_j where $i \neq j$, we have that $Opt \geq \sum_{j=1}^{l} Opt_j$.

If G is not small the clique cover for G is formed as the union of the exact clique cover for GC and the $\log n$-approximate clique covers of G_j for all j, $1 \leq j \leq l$.

Theorem 3.1 *The algorithm produces a $\log n$-approximate clique cover for a subtree filament graph G in $O(nP(n) + n^3)$ time and for a circle graph in $O(n \log n \log \log n)$ time.*

Proof. We show by induction on n that the algorithm produces a $\log n$-approximate clique cover. If n is small then the algorithm produces an optimal clique cover. Otherwise consider an n vertex subtree filament graph and assume that for smaller subtree filament graphs the algorithm does produce a solution of size at most $\log n$ times the size of the optimal solution. Since the algorithm combines $\log n$-approximate solutions for the G_js for all j, $1 \leq j \leq l$, with an optimal solution for GC, the size of the clique cover produced for G is at most

$$OptC \ + \ \sum_{j=1}^{l} \log |V_j| Opt_j$$

Since each G_j, $1 \leq j \leq l$, has at most $n/2$ vertices, the size of the approximate clique cover for G is at most

$$OptC \ + \ \log(n/2) \sum_{j=1}^{l} Opt_j$$

We also have $Opt \geq OptC$ and $Opt \geq \sum_{j=1}^{l} Opt_j$. Therefore, the algorithm returns a solution of size at most

$$Opt + \log(n/2) Opt \ = \ \log n \cdot Opt$$

We now consider the running time of the algorithm. If n is less than some constant then the running time, $f(n)$, is bounded above by a constant. Otherwise $f(n) \leq P(n) + cn^2 + \sum_{j=1}^{l} f(|V_j|)$, where c is a constant, as the clique cover problem on cocomparability graphs can be solved in linear, that is, $O(n^2)$ time [22]. Solving the recurrence yields $f(n) \in O(nP(n) + n^3)$.

When the input is restricted to circle graphs, the algorithm has the same approximation ratio but runs in $O(n \log n \log \log n)$ time, since the partitioning can be done in linear time and the clique cover problem on permutation graphs can be solved in $O(n \log \log n)$ time [22]. □

3.2 The vertex colouring problem

A (vertex) colouring of a graph is an assignment of colours to vertices such that no pair of adjacent vertices is assigned the same colour. The circle graph colouring problem was shown to be NP-complete in 1980 [7]. Using the general divide-and-conquer scheme we now present a $\log n$-approximation algorithm for the problem of colouring a subtree filament graph with the minimum number of colours. For general graphs the problem is not approximable within $|V|^{\frac{1}{7}-\epsilon}$ for any $\epsilon > 0$ [2]. The colours assigned by an optimal colouring for G will be a feasible colouring for GC and for each G_j, $1 \leq j \leq l$. We therefore have $Opt \geq OptC$ and $Opt \geq Opt_j$ for all j, $1 \leq j \leq l$.

If G is not small the approximate colouring for G contains the recursively computed approximate colourings for G_j, $1 \leq j \leq l$, each independently using colours beginning from 1. Altogether this partial colouring of G requires at most $\max_{j=1}^{l} \log |V_j| Opt_j$ colours. An optimal colouring for GC is computed, but the colours for vertices in GC are renamed to begin at $\max_{j=1}^{l} \log |V_j| Opt_j + 1$.

Theorem 3.2 *The algorithm produces a $\log n$-approximate colouring for a subtree filament graph G in $O(nP(n) + n^{\frac{7}{2}})$ time and for a circle graph in $O(n \log n \log\log n)$ time.*

Proof. We show by induction on n that the algorithm produces a colouring for G using at most $\log n \cdot Opt$ colours. If n is small the algorithm produces a colouring with the minimum number of colours. Otherwise consider an n vertex subtree filament graph and assume that for smaller subtree filament graphs the algorithm will produce a colouring using at most $\log n$ times the minimum number of colours.

The algorithm produces a legal colouring with no more than $OptC + \max_{j=1}^{l} \log |V_j| Opt_j$ colours. Since each $|V_j|$ is less than or equal to $n/2$, this is no more than $OptC + \log(n/2) \max_{j=1}^{l} Opt_j$. We also have $Opt \geq OptC$ and $Opt \geq \max_{j=1}^{l} Opt_j$. Therefore, the algorithm returns a solution of size less than or equal to $Opt + \log(n/2) Opt = \log n \cdot Opt$.

The colouring problem can be solved optimally for cocomparability graphs in the time required to solve the bipartite matching problem, that is, in $O(\sqrt{n}m)$, or $O(n^{\frac{5}{2}})$, time [14]. To consider the running time, $f(n)$, of the approximation algorithm, we have $f(n) \leq c_2$ if $n \leq c_1$, and $f(n) \leq P(n) + c_3 n^{\frac{5}{2}} + \sum_{j=1}^{l} f(|V_j|)$ otherwise, where c_1, c_2, and c_3 are constants. Resolving the recurrence implies that $f(n) \in O(nP(n) + n^{\frac{7}{2}})$. For circle graphs, the algorithm runs in $O(n \log n \log\log n)$ time. $\square$

3.3 The maximum k-colourable subgraph problem

Given a subtree filament graph G and integer k, the problem is to find a vertex-induced subgraph H of G which is k-colourable and contains the maximum number of vertices. This problem is NP-complete for circle graphs since it contains the vertex colouring problem as a special case. In addition, Cong and Liu [4] have shown that this problem, on circle graphs, is NP-complete for any fixed $k \geq 2$. They also have shown that this problem is equivalent to the k-layer topological via minimization problem of circuit design [4] when the routing region is a switchbox and each net is a two-terminal net. We now use the general divide-and-conquer scheme to produce an approximation algorithm for the maximum k-colourable subgraph problem in subtree filament graphs. For general graphs the problem is as hard to approximate as maximum independent set [20]; it is not approximable to within $|V|^{1-\epsilon}$ for any $\epsilon > 0$.

The optimal solution for G cannot include more vertices from GC, respectively G_j, than the optimal solution for GC, respectively G_j. Thus we have that $Opt \leq OptC + \sum_{j=1}^{l} Opt_j$ or equivalently $Opt - OptC \leq \sum_{j=1}^{l} Opt_j$.

If G is not small the approximate k-colourable subgraph for G will consist of the larger of either (1) the exact optimal solution for GC or (2) the union of the approximate solutions for G_j, for all j, $1 \leq j \leq l$.

Theorem 3.3 *The algorithm produces a $\frac{1}{\log n}$-approximate solution to the problem of producing the maximum k-colourable subgraph of a subtree filament graph G in $O(nP(n) + kn^3)$ time. For circle graphs, if k is small the algorithm runs in $O(kn \log^2 n)$ time and, for large k, $O(n^{\frac{3}{2}} \log^2 n)$ time is sufficient.*

Proof. We show by induction on n that the algorithm produces a k-colourable subgraph of G containing at least $\frac{1}{\log n}$ times the optimal number of vertices. If n is small the algorithm produces a k-colourable subgraph with the maximum number of vertices. Otherwise consider an n vertex subtree filament graph and assume that for smaller subtree filament graphs the algorithm produces a k-colourable subgraph of size at least $\frac{1}{\log n}$ times the size of the maximum k-colourable subgraph.

The algorithm produces a solution containing at least $\max[OptC, \ \sum_{j=1}^{l}(Opt_j \cdot \frac{1}{\log |V_j|})]$ vertices. Since $|V_j| \leq n/2$ for all j, $1 \leq j \leq l$, the size of the solution is at least $\max[OptC, \ \frac{1}{\log(n/2)} \cdot \sum_{j=1}^{l} Opt_j]$. We also have $Opt - OptC \leq \sum_{j=1}^{l} Opt_j$. Therefore, the algorithm returns a solution of size at least $\max[OptC, \ \frac{Opt-OptC}{\log(n/2)}]$.

If $OptC \geq \frac{(Opt-OptC)}{\log(n/2)}$ then $OptC \cdot \log(n/2) + OptC \geq Opt$; therefore $OptC \cdot \log n \geq Opt$ and $OptC \geq \frac{Opt}{\log n}$. Thus in this case the solution used is greater than or equal to $\frac{Opt}{\log n}$.

The remaining case has $OptC < \frac{(Opt-OptC)}{\log(n/2)}$. This is equivalent to $OptC < \frac{Opt}{\log n}$. The algorithm uses the union of the approximate solutions for G_j, for all $1 \leq j \leq l$, as the overall solution. This solution will be of size at least $\frac{(Opt-OptC)}{\log(n/2)} \geq \frac{(Opt-\frac{Opt}{\log n})}{\log(n/2)} = \frac{Opt(1-\frac{1}{\log n})}{\log(n/2)}$ $= \frac{Opt[\frac{(\log n-1)}{\log n}]}{\log(n/2)} = \frac{Opt}{\log n}$.

An $O(kn^2)$ algorithm to solve this problem optimally on cocomparability graphs appears in [9]. Thus, the approximation algorithm runs in time $f(n) \leq P(n) + ckn^2 + \sum_{j=1}^{l} f(|V_j|)$, where c is a constant. Thus $f(n) \in O(nP(n) + kn^3)$.

A maximum k-colourable subgraph of a permutation graph can be found in $O(kn \log n)$ or, for large k, in $O(n^{\frac{3}{2}} \log n)$ time [6]. For small k this gives a recurrence $f(n) \leq 2f(\frac{n}{2}) + cknlogn$ with solution $f(n) \in O(kn \log^2 n)$ and for large k the recurrence is $f(n) \leq 2f(\frac{n}{2}) + cn^{\frac{3}{2}} \log n$ with solution $O(n^{\frac{3}{2}} \log^2 n)$. $\Box$

3.4 The maximum h-coverable subgraph problem

An induced subgraph of graph G is h-coverable if its vertices can be partitioned into at most h cliques. In the maximum h-coverable subgraph problem, the goal is to find an h-coverable subgraph containing the maximum number of vertices. The NP-completeness of this problem on subtree filament graphs follows from the NP-completeness of clique cover on circle graphs. For general graphs the problem is as hard to approximate as maximum independent set [20]; it is not approximable within $|V|^{1-\epsilon}$ for any $\epsilon > 0$.

An $O(|V|^3 \log |V|)$ algorithm to solve this problem exactly on cocomparability graphs appears in [9].

As in Gavril [9], for graph $G = (V, E)$ and for $0 \leq h \leq |V|$, we let $d_h(G)$, be the size of a maximum h-coverable subgraph of G. We assume that an h-coverable subgraph of size at least $\frac{1}{\log n}$ times optimal can be computed for graphs having $n < |V|$ vertices, for any h, $0 \leq h \leq n$, and let $d_h^{apx}(G)$ be the size of an approximate maximum h-coverable subgraph of graph G that is returned by the algorithm. In addition, we define two sequences of integers, $\{h_i | 1 \leq i \leq l\}$ and $\{h'_i | 1 \leq i \leq l\}$. In both cases, the l elements of the sequence sum to h.

Let h_i, $1 \leq i \leq l$, be such that:

- $0 \leq h_i \leq h$, for all $1 \leq i \leq l$,
- $$\sum_{i=1}^{l} h_i = h, \text{ and}$$
- over all $\{h_i | 1 \leq i \leq l\}$ satisfying the two properties above, the following is maximized:

$$\sum_{i=1}^{l} d_{h_i}^{apx}(G_i)$$

For the second sequence, let h'_i, $1 \leq i \leq l$, be such that:

- $0 \leq h'_i \leq h$, for all $1 \leq i \leq l$,
- $$\sum_{i=1}^{l} h'_i = h, \text{ and}$$
- over all $\{h'_i | 1 \leq i \leq l\}$ satisfying the two properties above, the following is maximized:

$$\sum_{i=1}^{l} d_{h'_i}(G_i)$$

The approximation algorithm for G takes the larger of the exact value for $OptC$ and the largest sum of approximate values of the G_i's, specifically:

$$\max \; [\; OptC, \; \sum_{i=1}^{l} d_{h_i}^{apx}(G_i) \;]$$

The quantity

$$\sum_{i=1}^{l} d_{h_i}^{apx}(G_i)$$

can be computed as follows:

We compute $apx_k(i,j)$, the size of an approximate maximum k-coverable subgraph of $G_i \cup G_{i+1} \cup \ldots \cup G_j$, for all $1 \leq i \leq j \leq l$, $0 \leq k \leq h$, by dynamic programming based on the following:

1. $apx_0(i,j) = 0$ for all $1 \leq i \leq j \leq l$
2. $apx_k(i,i)(\ = d_k^{apx}(G_i))$ can be computed, by assumption, for all $1 \leq i \leq l$ and all $1 \leq k \leq h$
3. For all $1 \leq i < j \leq l$, $k > 0$: $apx_k(i,j) = \max_{0 \leq k' \leq k}[apx_{k'}(i, j-1) \ + \ apx_{k-k'}(j,j) \ ]$
4. Finally, the value $apx_h(1,l)$ is the desired quantity

Theorem 3.4 *The algorithm produces a $\frac{1}{\log n}$-approximate solution to the maximum h-coverable subgraph problem of subtree filament graph G in $O(hnP(n) + hn^4 \log n)$ time. For circle graphs, the algorithm runs in $O(h^2 n \log^2 n)$ time for small h and in $O(hn^{\frac{3}{2}} \log^2 n)$ time for large h.*

Proof. Let $G = (V,E)$ be a subtree filament graph having n vertices and assume that for smaller subtree filament graphs the algorithm produces an h-coverable subgraph of size at least $\frac{1}{\log n}$ times the size of the maximum h-coverable subgraph.

The algorithm returns a value which is

$$= \max[OptC, \ \sum_{i=1}^{l} d_{h_i}^{apx}(G_i) \ ]$$

$$\geq \max[OptC, \ \sum_{i=1}^{l} d_{h'_i}^{apx}(G_i) \ ]$$

$$\geq \max[OptC, \ \sum_{i=1}^{l} d_{h'_i}(G_i) \cdot \frac{1}{\log |V_i|} \ ]$$

$$\geq \max[OptC, \ \frac{1}{\log(n/2)} \cdot \sum_{i=1}^{l} d_{h'_i}(G_i) \ ]$$

by the choice of the h_i and h'_i sequences, and since $|V_i| \leq n/2$, for all $1 \leq i \leq l$. By considering an optimal solution restricted to GC and to $\cup_{i=1}^{l} G_i$, we see that we also have

$$Opt \leq OptC + \sum_{i=1}^{l} d_{h'_i}(G_i)$$

Therefore, the algorithm returns a solution of size

$$\geq \max[OptC, \ \frac{Opt - OptC}{\log(n/2)} \ ]$$

which is greater than or equal to $\frac{Opt}{\log n}$ by the same reasoning as that used in the proof of Theorem 3.3.

The running time of the algorithm can be expressed as $f(n) \leq P(n) + n^3 \log n + \sum_{i=1}^{l}(f(|V_i|) \cdot h) + l^2 h$, that is, $f(n) \in O(hnP(n) + hn^4 \log n)$. Since a maximum h-coverable subgraph in G is a maximum k-colourable subgraph in $\overline{G}$, these two problems have the same time complexity for permutation graphs. Thus, for circle graphs, the approximation algorithm runs in time $f(n) \leq n + chn \log n + 2hf(n/2)$, that is, $f(n) \in O(h^2 n \log^2 n)$, for small h, and $f(n) \leq n + cn^{\frac{3}{2}} \log n + 2hf(n/2)$, that is, $f(n) \in O(hn^{\frac{3}{2}} \log^2 n)$, for large h. $\Box$

3.5 Improved approximation algorithm for clique cover in circle graphs

In the case of circle graphs for the clique cover problem we are able to improve the approximation factor from $\log n$ to $\log Opt$ by dividing the intervals in the middle of the optimal solution rather than at the fixed position $n + 0.5$. Since we do not know the optimal solution in advance we need to try all possible dividing positions. To do this we use dynamic programming.

Let $G_{ij}, 1 \leq i < j \leq 2n$ be the induced subgraph of G containing the vertices of G whose corresponding intervals lie completely within the closed interval $[i, j]$. Let O_{ij} be the size of the optimal clique cover of G_{ij}. Let $G_{ij}^k, i \leq k \leq j$, be the permutation subgraph of G_{ij} consisting of the induced subgraph of G_{ij} containing the vertices corresponding to intervals that contain the integer k. Let O_{ij}^k be the size of the optimal clique cover of G_{ij}^k. Since G_{ij}^k is an induced subgraph of G_{ij}, we have $O_{ij}^k \leq O_{ij}$ for all $i \leq k \leq j$.

The dynamic programming algorithm computes CC_{ij}, the size of an approximate clique cover for G_{ij}, for all $1 \leq i < j \leq n$. If $j - i \leq 2$ then G_{ij} consists of at most one vertex and accordingly zero or one can be placed in CC_{ij} as the size of the optimal clique cover of G_{ij}. Otherwise, the algorithm computes

$$CC_{ij} = \min_{i<k<j} [CC_{i,(k-1)} + CC_{(k+1),j} + O_{ij}^k]$$

in increasing order of $j - i$.

Theorem 3.5 *The dynamic programming algorithm computes the size of a* $\log Opt$*-approximation to the clique cover problem where Opt is the size of an optimal clique cover. This is done in* $O(n^3 \log \log n)$ *time.*

Proof. The proof is by induction on $j-i$ that CC_{ij} contains the size of a $\log O_{ij}$-approximate clique cover of G_{ij}. If $j - i \leq 2$ then CC_{ij} contains the size of an optimal clique cover of G_{ij}. Otherwise consider the computation of CC_{ij} where $j - i = q > 2$ and assume that all CC_{ij} for $j - i < q$ have been correctly computed.

One way to describe the location of a clique in G_{ij} is by the coordinate of the leftmost right endpoint of an interval corresponding to a vertex in the clique. All of the intervals corresponding to the clique will contain this point. See figure 4. In the computation of CC_{ij}

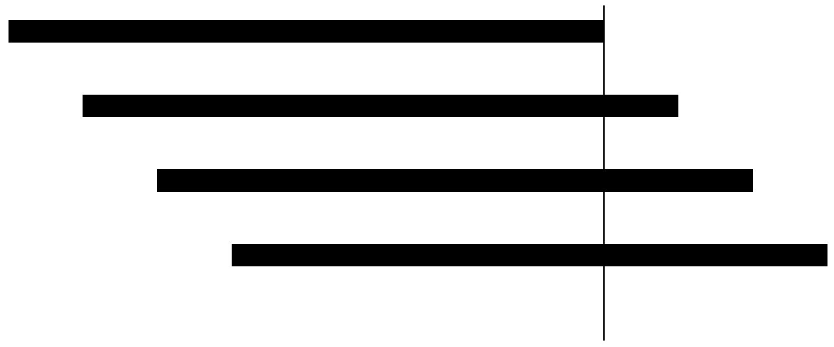

Figure 4: All of the intervals corresponding to the vertices of a clique contain the leftmost right endpoint of such an interval

all possible k are considered between i and j. In particular the value of k corresponding to the median value, k^*, of the leftmost right endpoint of the cliques in an optimal solution for G_{ij} is examined. This choice of k^* implies that $O_{i,(k^*-1)} \leq \frac{O_{ij}}{2}$ and $O_{(k^*+1),j} \leq \frac{O_{ij}}{2}$. The properties of k^* and the fact that $O_{ij}^k \leq O_{ij}$, for $i < k < j$ ensure that

$$\begin{aligned} CC_{ij} &\leq O_{i,k^*-1} \log(O_{i,(k^*-1)}) + O_{k^*+1,j} \log(O_{k^*+1,j}) + O_{ij}^{k^*} \\ &\leq \frac{O_{ij}}{2} \log(\frac{O_{ij}}{2}) + \frac{O_{ij}}{2} \log(\frac{O_{ij}}{2}) + O_{ij} \\ &= O_{ij} \log(\frac{O_{ij}}{2}) + O_{ij} = O_{ij}[\log(\frac{O_{ij}}{2}) + 1] \\ &= O_{ij}[\log(O_{ij}) - 1 + 1] = O_{ij} \log O_{ij} \end{aligned}$$

as required.

The algorithm computes n^2 of the C_{ij}, and each of these requires $O(n \log\log n)$ time to compute. Overall the algorithm runs in $O(n^3 \log\log n)$ time. □

Acknowledgment

This work was supported financially by the Natural Sciences and Engineering Research Council of Canada. The authors thank two anonymous referees for suggestions that improved the paper.

References

[1] P.K. Agarwal, M. van Kreveld and S. Suri, Label placement by maximum independent set in rectangles. *Computational Geometry: Theory and Applications*, 11(1998), 209-218.

[2] M. Bellare, O. Goldreich and M. Sudan, Free bits, PCPs and non-approximability - towards tight results, *SIAM Journal on Computing*, 27(1998), 804-915.

[3] E. Čenek and L. Stewart, Maximum independent set and maximum clique algorithms for overlap graphs, *Discrete Applied Mathematics*, 131 (2003), 77-91.

[4] J. Cong and C.L. Liu, On the k-layer Planar Subset and Topological Via Minimization Problems, *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 10 (1991), 972-981.

[5] B. Dushnik and E.W. Miller, Partially Ordered Sets, *American Journal of Mathematics*, 63(1941), 600-610.

[6] S. Felsner and L. Wernisch, Maximum k-chains in planar point sets: Combinatorial structure and Algorithms, *SIAM Journal on Computing*, 28(1998), 192-209.

[7] M.R. Garey, D.S. Johnson, G. L. Miller, and C.H. Papadimitriou, The complexity of coloring circular arcs and chords, *SIAM Journal on Algebraic and Discrete Methods*, 1(1980), 216-227.

[8] F. Gavril, Algorithms for a maximum clique and a maximum independent set in a circle graph, *Networks*, 3(1973), 261-273.

[9] F. Gavril, Algorithms for maximum k-colorings and k-coverings of transitive graphs, *Networks*, 17 (1987), 465-470.

[10] F. Gavril, Maximum weight independent sets and cliques in intersection graphs of filaments, *Information Processing Letters*, 73 (2000), 181-188.

[11] F. Gavril, Perfect interval filament graphs, DIMACS technical report 2003-37, 2003.

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

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

[14] J.E. Hopcroft and R.M. Karp, An $n^{\frac{5}{2}}$ algorithm for maximum matchings in bipartite graphs, *SIAM Journal on Computing*, 2(1973), 225-231.

[15] W.L. Hsu and K.H. Tsai, Linear time algorithms on circular-arc graphs, *Information Processing Letters*, 40(1991), 123-129.

[16] D. Johnson, NP-completeness Column: An Ongoing Guide, *Journal of Algorithms*, 6(1985), 434-451.

[17] R.M. Karp, Reducibility among combinatorial problems, in R.E. Miller and J.W. Thatcher (eds.) *Complexity of Computer Computations*, Plenum Press, 1972, New York, 85-103.

[18] J.M. Keil, The complexity of the domination problems in circle graphs, *Discrete Applied Mathematics*, 42(1993), 51-63.

[19] F. Nielsen, Fast stabbing of boxes in high dimensions, *Theoretical Computer Science*, 246(2000), 53-72.

[20] A. Panconesi and D. Ranjan, Quantifiers and approximation, *Theoretical Computer Science*, 107(1993), 104-163.

[21] J. Spinrad, Recognition of circle graphs, *Journal of Algorithms*, 16(1994), 264-282.

[22] J. Spinrad, *Efficient Graph Representations*, Fields Institute Monographs 19, American Mathematical Society, 2003.

[23] C. A. Tovey, A simplified satisfiability problem, *Discrete Applied Mathematics*, 8(1984), 85-89.

[24] W. Unger, On the k-colouring of circle-graphs, STACS 88, 61-72, Lecture Notes in Computer Science, 294, Springer, Berlin, 1988.

[25] W. Unger, The complexity of colouring circle graphs, STACS 92, 389-400, Lecture Notes in Computer Science, 577, Springer, Berlin, 1992.