October, 1983

CLUSTERING AND DOMINATION IN PERFECT GRAPHS

D.G. Corneil[1] and Y. Perl[1,2]

DCS-TR-135

[1]Department of Computer Science
University of Toronto

[2]Computer Science Department
Rutgers University
and Ban-Ilan University

Department of Computer Science
Rutgers University
New Brunswick, NJ

October 1983

CLUSTERING AND DOMINATION IN PERFECT GRAPHS

D.G. Corneil[1] and Y. Perl[2]

DCS-TR-135

[1]Department of Computer Science
University of Toronto

[2]Computer Science Department
Rutgers University
and Ban-Ilan University

Department of Computer Science
Rutgers University
New Brunswick, New Jersey 08903

CLUSTERING AND DOMINATION IN PERFECT GRAPHS

D. G. Cornell
Department of Computer Science
University of Toronto

Y. Perl
Computer Science Department
Rutgers University
and Ban-Ilan University

Abstract

A K-cluster in a graph is an induced subgraph on **k-vertices which maximizes** the number of edges. Both the K-cluster problem and the K-dominating set problem are NP-complete for graphs in general. In this paper we investigate the complexity status of these problems on various subclasses of perfect graphs. In particular, we examine comparability graphs, chordal graphs, bipartite graphs, split graphs, cographs and K-trees. For example, it is shown that the K-cluster problem is NP-complete for both bipartite and chordal graphs and the independent K-dominating set problem is NP-complete for bipartite graphs. Furthermore, where the K-cluster problem is polynomial we study the weighted and connected versions as well. Similarly we also look at the minimum K-dominating set problem on families which have polynomial K-dominating set algorithms.

Introduction

In this paper we examine the complexity of two important graph theory concepts namely, clustering and domination. A k-cluster in a graph is a subset of k vertices which maximizes the number of edges in the subset (i.e. a k-cluster is a subset of k vertices with maximum density of edges). This notion encompasses that of *k-clique*, namely, a subset of k vertices with all $\binom{k}{2}$ edges being present. A *k-dominating set* in a graph is a subset D of k vertices such that all vertices in the graph are either in D or adjacent to a vertex in D. Both the k-clique and the k-dominating set problems are well known members of the NP-complete class of problems. One immediately sees that the NP-completeness of the k-clique problem establishes the NP-completeness of the decision problem formulation of the k-cluster problem: namely, given a graph $G(V,E)$ and positive integers k and m, does there exist an induced subgraph on k vertices such that this subgraph has at least m edges.

For many restricted families of graphs, the k-clique problem is known to have a polynomial time algorithm [9]. These families are cases where the complexity of the k-cluster problem is open to investigation. Similarly, the k-dominating set problem has been shown to have polynomial time solutions for various restricted families of graphs [9], [13]. From an application point of view, perfect graphs play an important role and many problems have been examined for various families of perfect graphs [11].

We shall see that both the k-cluster problem and the k-dominating set problem are NP-complete for perfect graphs and demonstrate very similar behaviour for various families of perfect graphs. In particular, the families of comparability graphs and chordal graphs will be examined in some detail. Furthermore, when one of these problems has a polynomial time algorithm, we will examine the complexity of the weighted version of the problem on that

family. The weighted version of the k-clustering problem involves searching a graph with weighted edges for the k-subset with maximum total weight on its edges. The weighted version of the dominating set problem is the standard minimum weight dominating set where the vertices of the graph are assigned weights. Also we are often interested in finding a connected cluster and an independent dominating set.

Before discussing these results we present the relevant definitions and notation. The k-cluster problem is examined in section 3; the k-dominating set problem in section 4. The final section provides some concluding remarks and open problems.

2. Notation and Definitions

A graph is *perfect* if for all induced subgraphs, the chromatic number equals the clique number. A *comparability graph* is one which is transitively orientable; i.e. there exists an orientation of G such that if $a \rightarrow b$ and $b \rightarrow c$ then $a \rightarrow c$. A *chordal graph* has the property that every cycle of length greater than or equal to four has a chord. Both comparability graphs and chordal graphs are known to be perfect. See [11] for properties and algorithms for these and other perfect graphs.

Bipartite graphs are one example of comparability graphs. Another example is that of *cographs* (or complement reducible graphs). These graphs have many equivalent definitions; two of which are (i) they are the graphs which have no induced P_4 and (ii) the complement of any connected induced subgraph is disconnected. Furthermore, cographs have the important property of having a unique tree representation. In this representation (called a *cotree*) the vertices of the graph are the leaves of the tree. The internal nodes of the tree are labelled (0) or (1). The root r of the tree is labelled (1) and on each

path from the root the labels of the internal nodes alternate. Two vertices x and y of G are adjacent iff the unique path from x to r first meets the unique path from y to r at a (1) node. This unique tree representation leads to many fast algorithms for cographs. The reader is referred to [4] for further properties and algorithms on cographs. An example of a cograph and its cotree is presented in Fig.1.

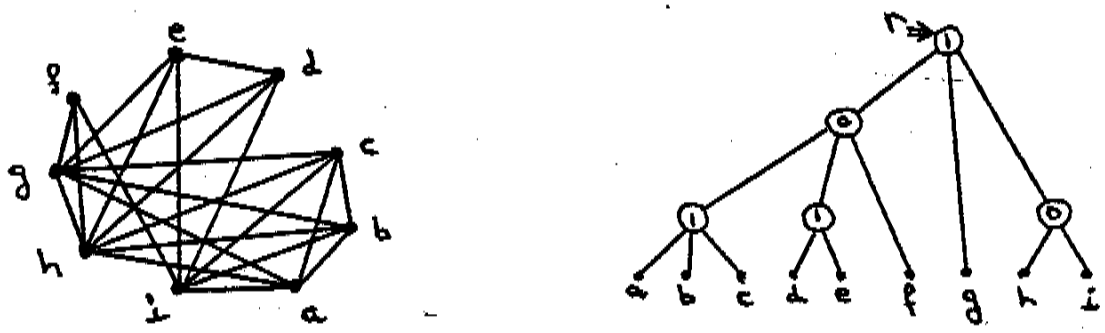

Figure 1

A *simplicial* vertex is one for which the set of neighbours forms a clique. A graph has a *perfect elimination scheme* if there exists an order of eliminating the vertices such that each vertex is simplicial at the time it is eliminated. A graph is chordal iff it has a perfect elimination scheme [11]. This perfect elimination scheme indicates that the maximum cliques of a chordal graph interlock in a very tree-like way. This tree is called the clique tree (see [10]).

A graph is a *split graph* if there exists a partitioning of the vertex set V into sets V_1 and V_2 where the induced subgraph on V_1 is complete and the induced subgraph of V_2 is void. Clearly split graphs are chordal. Another class of chordal graphs is k-trees. A *k-tree* is defined recursively as follows. K_k is a k-tree; if G is a k-tree then so is G' the graph formed by adding a new vertex to G and making it adjacent to all vertices of a K_k in G. An example of a 2-tree and its clique tree is presented in Fig.2. Note that k-trees are chordal and that a 1-tree is a tree in the normal sense. Furthermore, 2-trees contain the class of

maximal outerplanar graphs (MOPs).

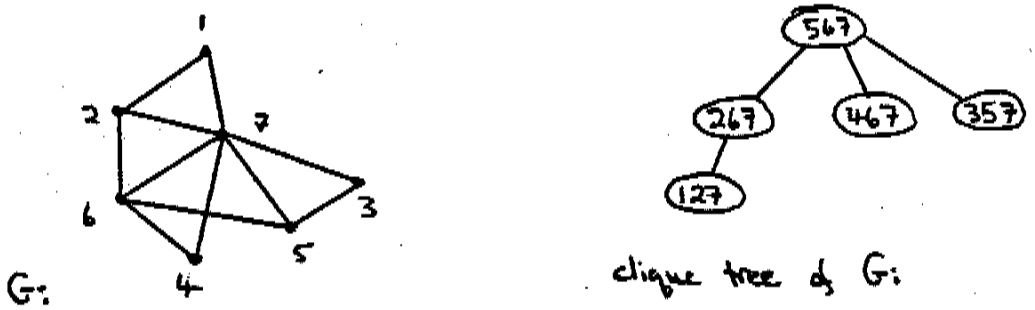

Figure 2. A 2-tree G where the vertex numbering indicates the order of the perfect elimination scheme.

3. The k-cluster Problem

Although the k-clique problem is polynomial for perfect graphs [12], we will see that the k-cluster problem is NP-complete. In fact, we will see that it is NP-complete for families where the k-clique problem is trivial, for example, bipartite graphs. We first explore clustering on comparability graphs and then examine chordal graphs.

3.1 Clustering on Comparability Graphs

As mentioned above the NP-completeness of the k-clustering problem on perfect graphs and comparability graphs follows from the result on bipartite graphs which we now present.

Theorem 3.1 The k-cluster problem on bipartite graphs is NP-complete.

Proof The reduction is from the k-clique problem on general graphs. Given a graph $G(V,E)$ we construct its bipartite incidence graph $H(W,F)$ where $W = V \cup E$. Let $k' = k + \binom{k}{2} = \binom{k+1}{2}$. We claim that H has a k'-cluster with at least $2\binom{k}{2}$ edges iff G has a k-clique.

Clearly, if G has a k-clique C then the $\binom{k}{2}$ edges of C in G are represented by $\binom{k}{2}$ vertices of W in H and each of them is adjacent to two vertices in $C \subseteq W$.

Thus there is a k'-cluster in H with $2\binom{k}{2}$ edges.

Now suppose that H contains a k'-cluster L with at least $2\binom{k}{2}$ edges. Let A denote $L \cap V$ and let B denote $L \cap E$ in the vertex set of H. If $|A| = a$ and $|B| = k' - a$ then we note that at most $\binom{a}{2}$ nodes in B may have degree 2 and the rest have degree at most 1.

If $a < k$, then the number of edges in the cluster L is at most $2\binom{a}{2} + \binom{k}{2} + k - a - \binom{a}{2} = \binom{k}{2} + \binom{a}{2} + k - a < 2\binom{k}{2}$ which contradicts the fact that L has $2\binom{k}{2}$ edges. If $a > k$, then $|B| < \binom{k}{2}$ and thus since all vertices in F have degree = 2 the number of edges in the cluster is again $< 2\binom{k}{2}$.

Hence $|A| = k$ and the only way to obtain a k'-cluster with $2\binom{k}{2}$ edges is for each of the $\binom{k}{2}$ vertices in B to contribute two edges to the k'-cluster; this in turn implies that L represents a k-clique in G. □

As a consequence of the above theorem we have the following:

Corollary 3.2 The following problems are NP-complete:

(i) The connected k-cluster problem on bipartite graphs.

(ii) The (connected) k-cluster problem on comparability graphs

(iii) The (connected) k-cluster problem for triangle-free graphs.

Remark: The proof of .Theorem 3.1 constructs a bipartite graph where the degree of all vertices of one part is two. An interesting open problem involves the complexity of k-clustering on bipartite graphs where all vertices have a fixed degree bound.

Some families of comparability graphs do have polynomial k-clustering algorithms. For example, in trees any subtree on k vertices is clearly a (connected) maximum k-cluster. An $O(k^2n)$ dynamic programming algorithm for

the weighted connected k-cluster in a tree appears in [15]. A similar algorithm solves the weighted k-cluster problem for a tree.

We now turn our attention to cographs. As mentioned in section 2, a cograph may be uniquely represented by a cotree. Using this cotree representation linear algorithms are known for problems such as maximum clique and minimum colouring [4]. We now present a polynomial time algorithm for the k-cluster problem on cographs. We assume that the cograph G is presented as a cotree T. Given a graph G one can determine if it is a cograph and if so construct its cotree in linear time [5].

In this algorithm we let the density of a k-cluster indicate the number of edges in that cluster. Thus we are searching for a k-cluster with maximum density. The algorithm which we present calculates the maximum density of any i-cluster for $i = 1,2,...,k$. The i-cluster itself can easily be reconstructed by storing appropriate pointers while performing the algorithm. These details are left to the reader.

For each vertex v of T we compute a vector D of k weights such that $D(v,i)$ $1 \le i \le k$ is the maximum density of any i-cluster in the subgraph of G represented by the subtree of T rooted at v. If there is no such i-cluster, then $D(v,i) = \Lambda$. Clearly $D(r,i)$ where r is the root of T, $1 \le i \le k$ is the maximum density of any i-cluster in G. The algorithm operates using a dynamic programming approach and processes the vertices of T in a bottom-up ordering computing the D vector for each vertex of T.

For every leaf of T, set $D(v,0) = D(v,1) = 0$ and $D(v,i) = \Lambda$ for $1 < i \le k$. In subsequent steps we assume that a sum containing a Λ equals Λ, and that in comparisons $\Lambda < 0$.

Assume an internal vertex v of T has d sons $x_1, x_2, ..., x_d$ for which the density vectors were already computed. Then taking into account all partitions of i

into d we can find D(v,i). However, the number of such partitions is $O(i^{d-1})$ and thus the computation is not guaranteed to be completed in polynomial time. In order to obtain a polynomial algorithm we perform the computation of D(v,i) by a step-by-step process (see e.g. [15]). Let D(v,i,j), $1 \le j \le d$, be the maximum density of any i-cluster in the subtree rooted at v containing its sons $x_1, x_2, \ldots, x_j$, and all their descendants. Clearly D(v,i) = D(v,i,d).

For i = 1,2,...,k set

$$D(v,i,1) = D(x_1,i), \quad D(v,0) = 0 \text{ and } D(v,0,m) = 0, 1 \le m \le d,$$

for j = 2,...,d

if label (v) = 0 then

$$D(v,i,j) = \max_{0 \le m \le i} [D(v,m,j-1) + D(x_j,i-m)]$$

and if label (v) = 1 then

$$D(v,i,j) = \max_{0 \le m \le i} [D(v,m,j-1) + D(x_j,i-m) + m \cdot (i-m)].$$

The validity of this computation follows from the validity of the principle of optimality in this case since all relevant partitions of i into d are considered by this step-by-step process. A 6-cluster for the example in Fig.1 {a,b,c,g,h,i} contains 14 edges. The complexity of computing the densities of v is $O(k^2 d)$. The sum of the degrees of all vertices of T is bounded by twice the number of leaves in the tree which is $|V| = n$. Hence the algorithm computing the densities of all vertices of T (including the root r) requires complexity $O(k^2 n)$. Note that the algorithm requires only O(nk) space since only one vector is required for computing the densities D(v,i,j) since the computation is in increasing order of i.

The following theorem shows that the above algorithm yields a solution for the connected k-cluster problem as well.

Theorem 3.3 If k<n then any k-cluster in a connected cograph G(V,E) on

n vertices is connected.

Proof For any subset S of V the *junction* of S is the unique lowest internal vertex v of the cotree T for which the subtree rooted at v contains all vertices of S. Clearly a subset S is connected if and only if its junction is labelled (1).

Assume now that a k-cluster C in G is disconnected. Hence the junction v of C is labelled (0). Since G itself is connected there exists an ancestor w of v labelled (1) whose degree is at least 2. Thus there must exist a vertex x in $V \backslash C$ which is adjacent to all vertices in C. Replacing any vertex in C by x yields a k-cluster with more edges than C, a contradiction. Hence C is connected. □

Consider now the weighted k-cluster problem for cographs. We have the following theorem for the binary weighted case, namely, where the edges are assigned weights either 0 or 1.

Theorem 3.4 The binary weighted k-cluster problem on cographs is NP-complete.

Proof The reduction is from the k-cluster problem for arbitrary graphs. Given a graph $G(V,E)$ where $|V| = n$, we form the cograph K_n. Edges in this graph which correspond to edges in G are given weight 1; all other edges are given weight 0. Clearly the weighted cograph has a k-cluster of weight m iff G has a k cluster with m edges. □

3.2 Clustering on Chordal Graphs

As pointed out in section 2, chordal graphs have a very explicit clique structure. It seems possible that this clique structure might lead to a polynomial algorithm for the k-cluster problem on chordal graphs; however, as the following theorem shows this is not likely.

Theorem 3.5 The k-cluster problem on chordal graphs is NP-complete.

Proof The reduction is from the k-clique problem on arbitrary graphs. Given a graph G(V,E) where $|V| = n$ we construct a chordal graph H(W,F) as follows. The vertices in G form a complete graph in H. For every edge $e = (v_i, v_j)$ in G construct a complete graph of n vertices and connect all of these new vertices to both v_i and v_j. Thus each edge in G is represented by a K_{n+2} in H. Clearly H has a perfect elimination scheme and thus is chordal.

Let $k' = k + \binom{k}{2}n$. We claim that H has a k'-cluster with at least $\binom{k}{2}\ \binom{n+2}{2}$ edges iff G has a k-clique. (In fact H will have a k'-cluster with exactly this number of edges). Clearly if G has a k-clique then H has such a k'-cluster.

The argument for the converse is very similar to that employed in the proof of Theorem 3.1. Suppose that H contains a k'-cluster L with at least $\binom{k}{2}\ \binom{n+2}{2}$ edges where A of cardinality a denotes $L \cap V$ and B denotes $L \cap E$ in the vertex set of H. At most $\binom{a}{2}n$ nodes in B may have "cross-degree" 2 and the rest have "cross-degree" at most 1.

If $a<k$ then the number of edges in L is at most

$$\binom{a}{2} + (k'-a)/n\ \binom{n}{2} + \binom{a}{2}2n + k' - a - \binom{a}{2}n =$$
$$= \binom{a}{2} + (k+\binom{k}{2}n-a)/n\ \binom{n}{2} + \binom{a}{2}2n + k + \binom{k}{2}n-a-\binom{a}{2}n =$$
$$= \binom{a}{2} + (k+\binom{k}{2}\ n-a)\ (n-1)/2 + \binom{a}{2}n+k+\binom{k}{2}n-a =$$
$$= \binom{k}{2}\binom{n}{2} + \binom{k}{2}\ n + k\ (n+1)/2 + \binom{a}{2}(n+1) - a(n+1)/2 =$$
$$= \binom{k}{2}\binom{n+1}{2} + k\ (n+1)/2 + a(a-2)(n+1)/2 =$$
$$< \binom{k}{2}\binom{n+1}{2} + (k+k(k-2))(n+1)/2 = \binom{k}{2}\ \binom{n+1}{2} + \binom{k}{2}(n+1) = \binom{k}{2}\binom{n+2}{2}.$$

which contradicts the fact that L has at least $\binom{k}{2}\ \binom{n+2}{2}$ edges.

If $a>k$ let $d=a-k$. Then $|B| = \binom{k}{2}n-d$ and even if all vertices in B have "cross degree" = 2 then the number of edges in L is at most

$$
\begin{aligned}
&(\tbinom{k}{2}-1)\tbinom{n}{2} + \tbinom{n-d}{2}+\tbinom{a}{2}+2(\tbinom{k}{2}n-d) = \\
&= \tbinom{k}{2}\tbinom{n}{2}-\tbinom{n}{2}+\tbinom{n}{2}-nd+\tbinom{d+1}{2}+\tbinom{a}{2}+ 2\tbinom{k}{2}n -2d = \\
&=\tbinom{k}{2}(\tbinom{n}{2} +2n) - (n+2)(a-k) + \tbinom{a-k+1}{2} + \tbinom{a}{2} = \\
&=\tbinom{k}{2}(\tbinom{n}{2} + 2n) - (n+2)(a-k) + \tbinom{a+1}{2} - ak+\tbinom{k}{2} + \tbinom{a}{2} = \\
&=\tbinom{k}{2}(\tbinom{n}{2} +2n+1) - (n+2)(a-k) + a(a-k) = \\
&=\tbinom{k}{2}\tbinom{n+2}{2} - (n+2-a)(a-k) < \tbinom{k}{2}\tbinom{n+2}{2}
\end{aligned}
$$

again contradicting the fact the L has at least $\binom{k}{2}\binom{n+2}{2}$ edges.

Thus a=k and the only way to obtain a k'-cluster with $\binom{k}{2}\binom{n+2}{2}$ edges is for each of the vertices in B to contribute two "cross-edges" to the k'-cluster and for B to consist of the union of $\binom{k}{2}$ K_n 's. This in turn implies that L represents a k-clique in G. □

Corollary 3.6 The connected k-cluster problem in chordal graphs is NP-complete.

We now examine various subfamilies of chordal graphs and demonstrate polynomial time algorithms for clustering on these graphs. The first class is that of split graphs where V can be partitioned into a complete subgraph C and an independent set I. If $k \leq c = |C|$, then a k-cluster of G is any k-subgraph of C. If k>c then the k-cluster contains C and the k-c vertices of I with the highest degrees.

The weighted k-cluster problem on split graphs is NP-complete as we now demonstrate where the edges are assigned weights 0 or 1.

Theorem 3.7 The binary weighted k-cluster problem on split graphs is NP-complete.

Proof The reduction is from the k-cluster problem for bipartite graphs. Given a bipartite graph $G(V,E)$ where $V = S \cup T$ we form a split graph H by making all vertices in S adjacent to each other. These edges are given weight 0; all others are given the weight 1. The proof is immediate. □

We now show that h-trees also have a polynomial time k-clustering algorithm. This algorithm is a generalization of the clustering algorithm for trees. First we form a clique tree representation of the h-tree. If $k \leq h+1$ then we immediately choose a K_k and are finished. Otherwise we choose a subtree in the clique tree on $k-h$ vertices. Regardless of the subtree chosen it is clear that the corresponding subgraph of the h-tree has k vertices and exactly $h(k - \frac{h+1}{2})$ edges. This is the maximum possible.

We now look at the domination problems.

4. The k-dominating Set Problems

As with the k-cluster problem we first look at comparability graphs.

4.1 Domination in Comparability Graphs

In section 3.1 we saw that the NP-completeness of clustering on comparability graphs follows from the NP-completeness of the problem on bipartite graphs. We now show that a similar result holds for the k-dominating set problem. A number of proofs of the NP-completeness of the k-dominating set problem on bipartite graphs have been independently obtained. The first such proof seems to be by Dewdney [6]. The proof which we present is for the independent domination problem and was developed by Farber [7].

Theorem 4.1 The independent k-dominating set problem is NP- complete for bipartite graphs.

Proof The reduction is from the h-dominating set problem for general graphs. Given a graph G, form the bipartite graph G' by replacing each edge in G by a P_5. We claim that G' has an independent dominating set of size m+h iff G has a dominating set of size h where $m = |E_G|$.

Given D' an independent dominating set of G' where $|D'| = h+m$, we construct D a dominating set of G such that $|D| = h$ as follows: First construct a dominating set D" of G' where $|D"| = |D'|$. Initially set D" = D'. Let $P_e = (u,a_e,b_e,c_e,v)$ correspond to the edge $e=(u,v) \in E_G$.

Let $Q_e = P_e \cap D'$. If $|Q_e| = 3$ then by the independence of D', $Q_e = \{u,v,h_e\}$. If $|Q_e| = 1$ then by the domination $Q_e = \{b_e\}$. If $|Q_e| = 2$ then unless $Q_e = \{a_e,c_e\}$, Q_e contains exactly one of the vertices a_e,b_e,c_e. In the case $Q_e = \{a_e,c_e\}$ we replace c_e by v in D".

It is clear that D" is a dominating set of G' and that $|D"| = |D'|$. Furthermore, since D' is independent and minimal it is straightforward to show that for each path P_e exactly one internal vertex is in D". Now set $D=\{v|v \in V_G \cap D"\}$ and note that D is a dominating set of G and that $|D| = h$.

Now assume that G has a dominating set D where $|D| = h$. We augment D to D' which is an independent dominating set of G' of cardinality h+m by considering each edge of G as follows:

(i) if $u,v \in D$ or $u,v \notin D$, add the middle vertex of P_e to D'.

(ii) if $u \in D$, $v \notin D$, add c_e, the vertex in P_e adjacent to v to D'.

It is straightforward to show that D' is an independent dominating set of G' and that $|D'| = m+h$. □

As a consequence of the above theorem we have the following:

Corollary 4.2 The following problems are NP-complete:

(i) The independent k-dominating set problem for comparability graphs.

(ii) The independent k-dominating set problem for triangle-free graphs.

(iii) The independent dominating set problem for bipartite graphs where all vertices in one part have degree ≤ 3 and all vertices in the other part have degree $=2$.

Point (iii) follows from the fact that the dominating set problem on cubic graphs is NP-complete (see [9]). Note that the problem is polynomial if all vertices in one cell have degree ≤ 2 and all vertices in the other cell have degree $=2$.

For some families of comparability graphs we do have polynomial time domination algorithms. The first result of this type is the linear algorithm for the domination of trees [2]. Cographs also have a straightforward linear time algorithm. In fact, it is easy to see that any connected cograph either has a dominating set of size one (i.e. a universal vertex) or a dominating set of size two. In the following we illustrate how the cotree representation of a cograph can be used to yield a linear time algorithm for the minimum dominating set problem on cographs. In this description of the algorithm only the weight of such a set is calculated. Minor modifications are required to determine the elements in the set. As with the cograph clustering algorithm, the domination algorithm uses a dynamic programming approach and processes the vertices of the cotree T in a bottom-up ordering.

Let the weight of a vertex v in G (and thus a leaf in T) be denoted by $w(v)$. For every vertex x in T we calculate two quantities for $G(x)$ the subgraph of G induced by the vertices in the subtree of T rooted at x. The first, $a(x)$ denotes the weight of a minimum dominating set of $G(x)$: the second, $b(x)$ records the smallest weight of any vertex in $G(x)$. Clearly $a(r)$ is the weight of a minimum dominating set of G. For x an internal node of T, let $x_1,\ldots,x_k$ denote the sons of

x. The algorithm proceeds as follows:

1. If x is a leaf, set a(x) = b (x) = w(x).

2. If x is a (0)-node, then no vertex in $G(X_i)$ dominates any vertex in $G(X_j)$ $j \neq i$. Thus set

$$a(x) = \sum_{1 \le i \le k} a(x_i)$$

$$b(x) = \underset{1 \le i \le k}{MIN}\{b(x_i)\}.$$

3. If x is a (1)-node, then any dominating set of $G(X_i)$ dominates G(x). An alternative dominating set contains a vertex of $G(X_i)$ to dominate $G(x)-G(x_i)$ and a vertex of $G(x_j)$ $j \neq i$ to dominate $G(x_i)$. Thus set

$$a(x) = \underset{1 \le i \le k}{MIN}\{\underset{1 \le i \le k}{MIN}(a(x_i)), \underset{1 \le i \le k}{MIN}(b(x_i)) + \underset{i \neq j}{MIN}\{b(x_j)\}\}$$

$$b(x) = \underset{1 \le i \le k}{MIN}\{b(x_i)\}$$

For example consider the cograph of Fig.1 where the cotree and the a and b values are given in Fig.3. The numbers under the leaves indicate the weight of the vertex (and the a and b values of the vertex). For internal nodes the a value is printed above the b value. Since the a value of the root is 3, there exists a dominating set of weight 3, namely {a,h}.

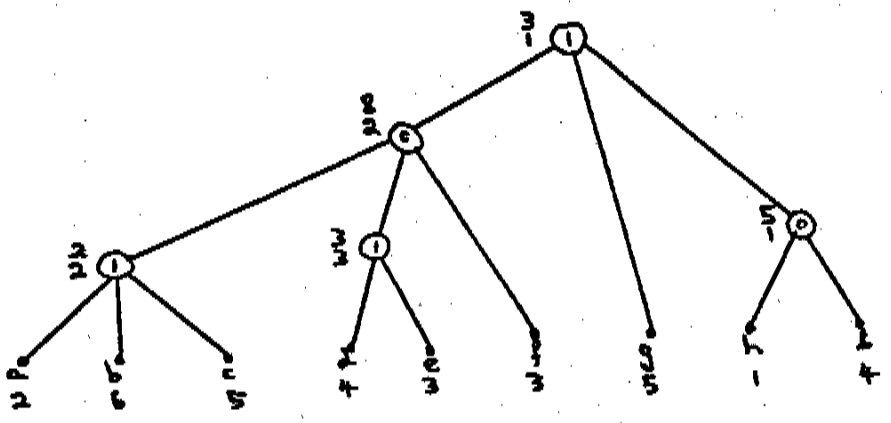

Figure 3

4.2 Domination in Chordal Graphs

In a recent paper Booth and Johnson [1] present many results on the domination problem in chordal graphs. They show that the k-domination problem is NP- complete for undirected path graphs, a restriction of chordal graphs and present linear algorithms for interval graphs and directed path graphs. We

continue their investigation by first studying split graphs.

Theorem 4.3 The k-dominating set problem is NP-complete for split graphs.

Proof The reduction is from the h-vertex cover problem on general graphs. Given a graph $G(V,E)$ construct first the incidence graph $I = (W,F)$, $W = V \cup E$. Change I into a split graph H by transforming the subgraph induced by V into a complete subgraph. We shall show that G has a vertex cover of h vertices if and only if H has a dominating set of h vertices. Clearly, if A is a vertex cover of G then A is a dominating set of H. On the other hand, let B be a minimum dominating set in H. Any vertex $e \in E$ in B can be replaced by a neighbour of e in V. Thus we obtain a dominating set $B' \subset V$, $|B'| = |B|$, such that B' is a vertex cover in G. □

Another chordal graph dominating set problem which has attracted considerable attention is the domination problem on k-trees for $k>1$. In [14] there is a polynomial time algorithm for domination of series-parallel graphs. Since a 2-tree is series parallel this settles the question for $k = 2$ (and thus also for MOPs, maximally outerplanar graphs). For k fixed it has recently been shown that the problem is polynomial, however if k is arbitrary then the problem is NP-complete [3]. Furthermore, the domination problem on permutation graphs is now known to have a polynomial time algorithm [8].

5. Concluding Remarks

It is interesting to note that the k-dominating set problem and the k-cluster problem seem to have the same complexity status for many families of perfect graphs. (See table 1.) One is tempted to conjecture that for perfect graphs the k-dominating set problem is more difficult than the k-cluster

Family	*k-cluster*	*k-dominating set*
trees	P	P
bipartite graphs	NPc	NPc
cographs	P	P
comparability graphs	NPc	NPc
k-trees, k fixed	P	P
k-trees, k arbitrary	P	NPc
split graphs	P	NPc
chordal graphs	NPc	NPc
permutation graphs	?	P

Table 1: Summary of results

problem in the sense that if the k-cluster problem is NP-complete then so is the k-dominating set problem. This however is false as can be seen by considering the following family of chordal graphs. G belongs to this family if it has a universal vertex x and $G\backslash\{x\}$ is a chordal graph. Obviously, the dominating set problem is polynomial whereas the k-cluster problem remains NP-complete.

We conclude this paper with a list of families of graphs where the complexity status of the k-cluster problem remains open. See Table 2.

interval graphs
planar graphs
fixed degree graphs
permutation graphs
bipartite graphs with fixed degree

Table 2: Open k-clustering problems

Acknowledgements

The authors wish to thank the Natural Sciences and Engineering Research Council of Canada for financial assistance. We also wish to thank Martin Farber for the use of his proof of Theorem 4.1 and both Martin Farber and Mark Keil for their comments on the domination problem.

References

[1] K.S. Booth and J.H. Johnson, Dominating sets in chordal graphs, *SIAM J. Comput.*, 11, 1 (1982), 191-199.

[2] E. Cockayne, S. Goodman and S. Hedetniemi, A linear algorithm for the domination number of a tree *Information Processing Letters* 4, 2 (1975), 41-44.

[3] D.G. Corneil and J.M. Keil, Domination in k-trees, in preparation.

[4] D.G. Corneil, H. Lerchs and L. Stewart Burlingham, Complement reducible graphs, *Discrete Applied Mathematics* 3 (1981), 163-174.

[5] D.G. Corneil, Y. Perl and L. Stewart Burlingham, Cographs: recognition, applications and algorithms, submitted for publication.

[6] A.K. Dewdney, Fast Turing reductions between problems in NP; Chapter 4: Reductions between NP-complete problems, Report #71, Department of Computer Science, University of Western Ontario (1981).

[7] M. Farber, private communications.

[8] M. Farber and M. Keil, Domination in permutation graphs, in preparation.

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

[10] F. Gavril, Algorithms for minimum coloring, maximum clique, minimum covering by cliques, and maximum independent set of a chordal graph, *SIAM J. Comput.* 1 (1972), 180-187.

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

[12] M. Grotschel, L. Lovasz and A. Schrijver, Polynomial algorithms for perfect graphs. Report No.81-176-OR, University of Bonn, 1981.

[13] S. Hedetniemi, Bibliography on the theory of domination in graphs, manuscript, 1981.

[14] T. Kikuno, N. Yoshida and Y. Kakuda, Dominating set in planar graphs, Tech. Rep. AL79-9, Institute of Electronics and Communication Engineers of Japan, Tokyo, Japan, 1979, 21-30.

[15] Y. Perl and Y. Shiloach, Efficient optimization of monotonic functions on trees, *Proceedings of the 6th CAAP Colloquium on Trees in Algebra and Programming*, Genoa 1981 (Springer Verlag Lecture Notes in Computer Science 112), 332-339. Also to appear *SIAM J. Alg. and Disc. Meth.*