UNIVERSITY OF IOANNINA

SCHOOL OF SCIENCES

S. D. Nikolopoulos
DEPARTMENT OF MATHEMATICS
Division of Applied Mathematics and Mechanics
Section: Computer Science
GR - 453 32, Ioannina
GREECE

Ioannina 12-12-85

Dear Dr. K. S. Booth,

I would greatly appreciate a reprint of your article :

Dominating Sets in chordal Graphs,
Report CS-80-34 , University of waterloo,
1980.

Thank you for your courtesy.

Sincerely yours,

Stavros Nikolopoulos

Mailed Report Jan 2/86

Dominating Sets in Chordal Graphs

Kellogg S. Booth

CS-80-34

July, 1980

Dominating Sets in Chordal Graphs

Kellogg S. Booth

Department of Computer Science
University of Waterloo
Waterloo, Ontario, Canada N2L 3G1

ABSTRACT

A set of vertices D is a dominating set for a graph if every vertex is either in D or adjacent to a vertex which is in D. We show that the problem of finding a minimum dominating set in a chordal graph is NP-complete and exhibit a linear time greedy algorithm for the problem restricted to the subclass of directed path graphs. Streamlined to handle only trees, our algorithm becomes the algorithm of Cockayne, Goodman and Hedetniemi. An interesting parallel with graph isomorphism is pointed out.

Keywords: chordal graph, directed path graph, dominating set, graph isomorphism, interval graph, minimum dominating set, undirected path graph

CR Categories: 5.25, 5.32

This material has been submitted for journal publication. The work reported here was supported in part by the Natural Sciences and Engineering Research Council of Canada under Grant No. A4307.

Dominating Sets in Chordal Graphs

Kellogg S. Booth

Department of Computer Science
University of Waterloo
Waterloo, Ontario, Canada N2L 3G1

1. Introduction

A set of vertices D is a *dominating set* for a graph $G = (V,E)$ if every vertex is either in D or adjacent to a vertex which is in D. A smallest such set is a *minimum dominating set.* For arbitrary graphs the problem of finding a minimum dominating set is NP-complete [6]. For the special case of trees there are algorithms which run in linear time [3,14]. We improve upon both of these results by showing that the problem remains NP-complete when restricted to chordal graphs but that a further restriction to directed path graphs admits a linear time solution.

Our method is a simple greedy algorithm which walks a tree representation of a directed path graph. It utilizes a linear time recognition algorithm for directed path graphs devised by Dietz, Furst and Hopcroft [4]. If the graph is a tree (trees are a subfamily of the directed path graphs) our algorithm can be streamlined to the original procedure of Cockayne, Goodman and Hedetniemi [3].

The remainder of this section is devoted to notation and a brief review of chordal graphs. Further details are available in the cited references.

We make heavy use of the term clique. A *clique* is any maximal set of vertices which are all mutually adjacent. Some authors do not insist on maximality; our usage follows Harary [11].

An edge is a *chord* of a cycle if it connects two vertices of the cycle but is not itself an edge within the cycle. A graph is *chordal* if and only if every cycle of length greater than three has a *chord.* For our purposes a more useful definition of chordality is the characterization proven by Gavril. His result can be used to classify a hierarchy of chordal graphs in terms of intersection models.

A graph is an *intersection graph* if there is a correspondence between its vertices and a family of sets (the *intersection model)* such that two vertices are adjacent in the graph if and only if their two corresponding sets have a nonempty intersection. Restricting the sets to subtrees of a tree determines the class of chordal graphs [8].

If the intersection model is further restricted, so that each subtree must be a path, a proper subclass called the *undirected path graphs* results [10]. Further restricting the model to rooted trees with paths directed away from the root yields the *directed path graphs* [9]. Requiring that the tree itself be a path defines the class of *interval graphs* [12].

As Gavril has shown in his papers, the intersection models can always be chosen so that the nodes of the tree are the cliques of the original graph. Each vertex then corresponds to the subtree comprised of exactly those cliques to which it belongs. We call such an intersection model a *clique tree* for the graph. Because we want to extract as much structural information as possible from the graph, we insist that the clique tree have each vertex correspond to a subtree, undirected path, directed path or subpath depending upon whether the original graph is chordal, undirected path, directed path or interval.

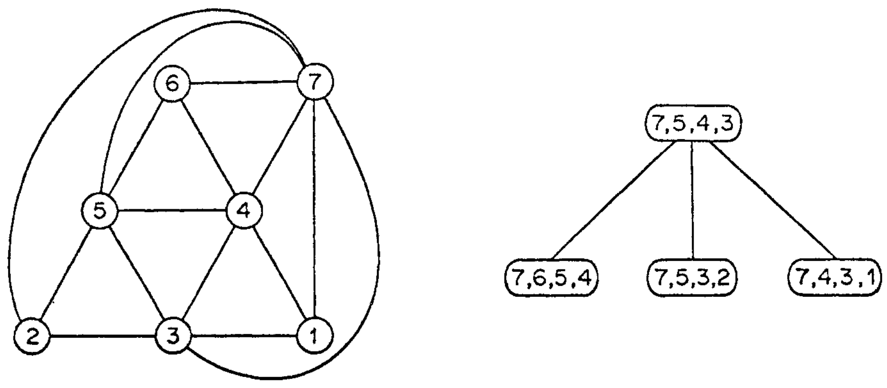

Figure 1(a). A chordal graph with its clique tree.

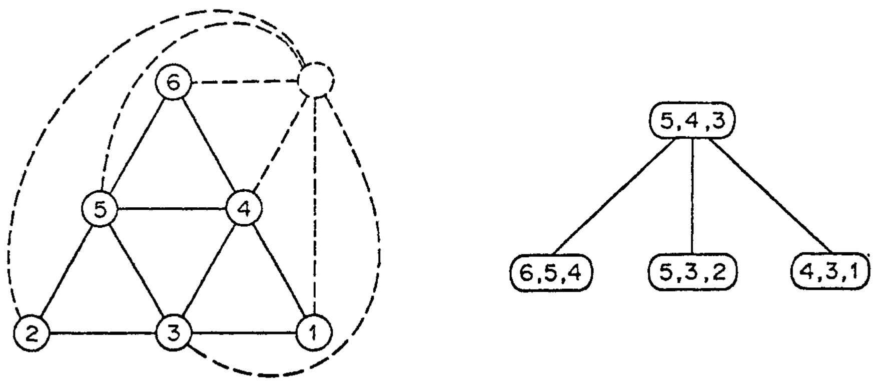

Figure 1(b). An undirected path graph with its clique tree.

The hierarchy of chordal graphs is illustrated in Figure 1 which shows (a) a chordal graph, (b) an undirected path graph, (c) a directed path graph and (d) an interval graph. Each belongs to its subclass but does not belong to the next more restrictive subclass. These four graphs are closely related. Each successive graph is obtained from the previous graph by vertex or edge deletion where the deleted elements are indicated with dashed lines. The examples are drawn from Gavril [8,9] and Lekkerkerker and Boland [12].

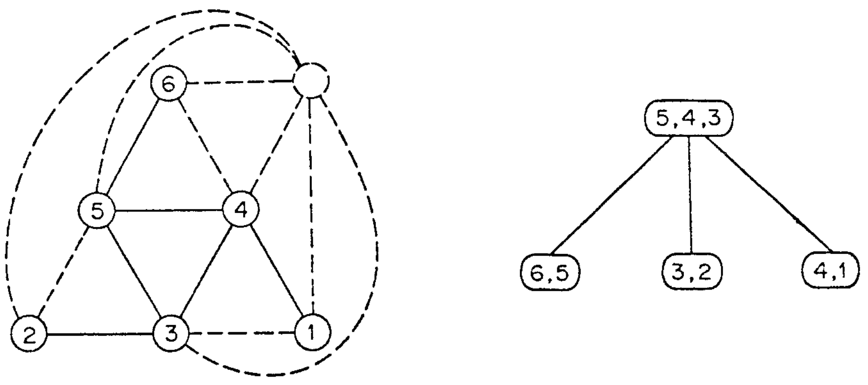

Figure 1(c). A directed path graph with its clique tree.

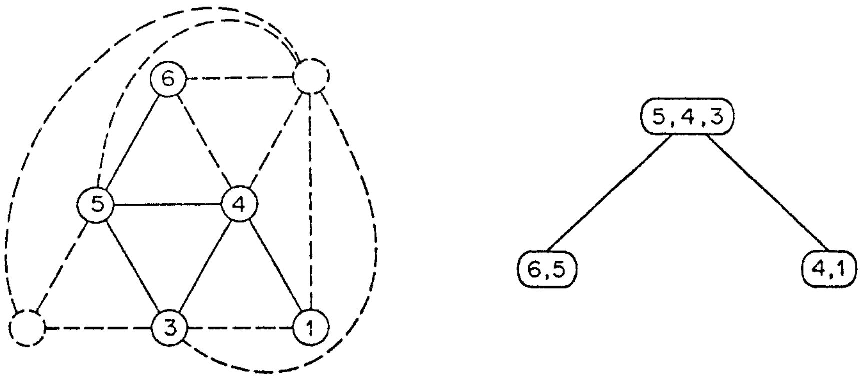

Figure 1(d). An interval graph with its clique tree.

A polynomial time algorithm for constructing the clique tree of an interval graph is easily obtained from an algorithm of Fulkerson and Gross [5]. Polynomial time algorithms for the other three classes of chordal graphs were first given by Gavril [8,9,10]. Linear time algorithms are now known for all but one of these classes. Lueker, Rose and Tarjan [16] have an algorithm for chordal graphs. Dietz, Furst and Hopcroft have a recognition algorithm for directed path graphs [4] which can be modified to produce a clique tree. Booth and Lueker [2] have a recognition algorithm for interval graphs which already produces the equivalent of a clique tree. To our knowledge there is no published linear time algorithm which builds a clique tree for undirected path graphs.

The notion of a path graph has been around for more than ten years. Renz introduced the idea in 1970 when he gave a partial characterization for undirected path graphs [15]. Dietz, Furst and Hopcroft base their work on a more recent paper by Truszczynski [4,17]. He refers to the problem as the generalized consecutive retrieval problem (see [2] for references to this and related concepts).

2. NP-Completeness for Chordal Graphs

No one has produced a polynomial time algorithm for finding a minimum dominating set of an arbitrary graph. The problem is NP-hard. As is usual for combinatorial minimization problems the NP-complete version is stated as a recognition problem: "Given a graph G and an integer k, is there a dominating set of size k for G?" We will use the latter formulation of the problem throughout this section.

The general dominating set problem was shown NP-complete using a reduction from the vertex cover problem [6, p. 190]. A slight variation of that reduction suffices to prove that even for the restricted case of chordal graphs the dominating set problem is NP-complete.

Theorem 1: The dominating set problem for chordal graphs is NP-complete.

Proof:

Let $G = (V,E)$ be an arbitrary graph and let k be an integer. We demonstrate that the vertex cover problem for G and k can be polynomially reduced to the dominating set problem for G' and k where G' is a chordal graph constructed from G.

Let $G' = (V',E')$ where $V' = V \cup E$. The vertices of G' which are in V are called *V-vertices* and those in E are called *E-vertices*. The edge set E' consists of edges between every pair of V-vertices and also between every E-vertex and its two incident V-vertices. Equivalently, G' is the union of the subdivision graph of G and a clique of the original vertices [11]. (The *subdivision graph* replaces each edge with a new vertex and two edges joining the new vertex to the old endpoints.) Figure 2 shows (a) the construction of G' from G and (b) the resulting clique tree. We claim that G has a vertex cover of size k if and only if G' has a dominating set of size k.

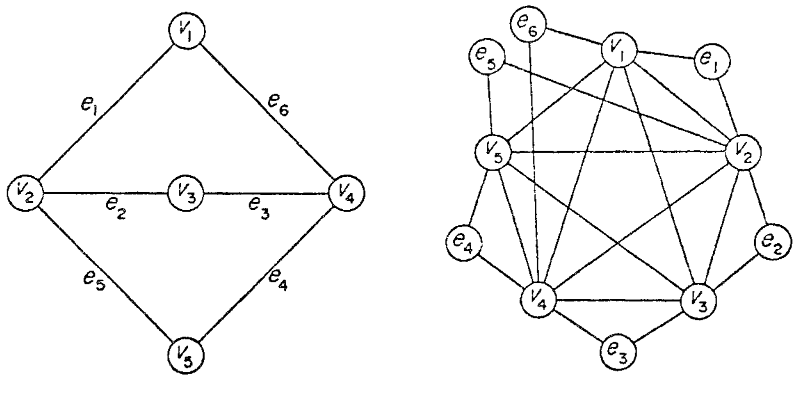

Figure 2(a). A graph G and its chordal graph G'.

Every vertex cover for G is also a dominating set for G'. This follows because every edge of G is incident to a vertex in the cover for G hence all vertices

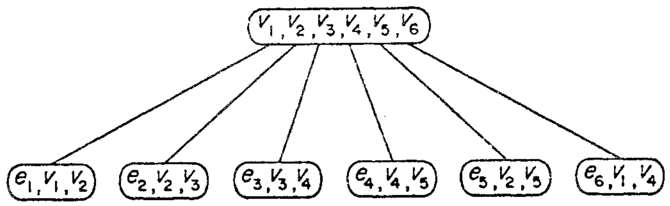

Figure 2(b). The clique tree for the chordal graph G'.

in G' are adjacent to some V-vertex in the cover: V-vertices because they form a clique and E-vertices because they are adjacent to their incident V-vertices from G.

Conversely, given a dominating set for G' we can assume without loss of generality that it contains no E-vertices; replacing such a vertex with either incident V-vertex maintains a dominating set. With this assumption every dominating set for G is a vertex cover for G because each E-vertex in G' is adjacent to a dominating V-vertex.

The proof is completed by noting that G' is the intersection graph of a clique tree whose root is the clique of all V-vertices and whose leaves are the cliques containing an E-vertex and its two incident V-vertices. Each E-vertex appears in precisely one clique (a subtree) and each V-vertex appears in the root plus each of the cliques for its incident edges (again a subtree). Thus G' is a chordal graph. □

3. A Linear Algorithm for Directed Path Graphs

This section describes a linear time algorithm for finding a minimum dominating set in a directed path graph. We will not give all of the details for our algorithm. Instead we will explain how to build it from pieces of other algorithms.

First we assume that a clique tree has been constructed for the directed path graph $G = (V,E)$. This requires only linear time using an easy modification to the recognition algorithm of Dietz, Furst and Hopcroft. The result is a rooted tree each of whose nodes is a clique of G. It has the property that the cliques containing any vertex form a contiguous path directed away from the root.

We next collect the following information, all in linear time, walking (or while constructing) the clique tree and using the original graph G.

[1] For each clique C find the integer *depth* $[C]$ which is the length of the path from the root of the clique tree to C. A list of all vertices contained in the clique is also generated.

[2] For each vertex v find the integer *high* $[v]$ which is the depth of the highest clique to which v belongs. A list of all adjacent vertices is also generated.

The rest of the algorithm is straightforward. Initialize the dominating set D to be empty, mark as "undominated" every vertex, then walk the clique tree in preorder (or any order in which all children are visited before their parent) performing the following operation.

[3] If there is a vertex v in the current clique C marked "undominated" and high[v]=C then choose a vertex x in C having the smallest value high[x] among all vertices in C. Add x to D and mark as "dominated" x and all vertices adjacent to x.

A linear running time for the algorithm is easily established. Each of the steps is linear either in the size of G, in the size of the clique tree (known to be linear in the size of G), or in the size of a clique. For any chordal graph the sum of the sizes of all cliques is linear in the size of G hence the entire algorithm is linear.

With the exception of the clique tree building, which uses the Dietz, Furst and Hopcroft algorithm, our solution is built entirely of standard algorithms from graph theory. Each of these is almost trivially linear time. Their algorithm is a bit more complicated and the proofs of linearity and correctness are quite involved. We remark that even without their algorithm we can substitute the polynomial time algorithm of Gavril to obtain a guaranteed polynomial running time for our algorithm.

Correctness of the algorithm follows from the next theorem. We introduce the notation *DOM(u)* to denote the set of vertices dominated by a vertex u and *DOM(U)* to denote the set of vertices dominated by a set U of vertices. A vertex dominates itself and any adjacent vertex.

Theorem 2: At the termination of the dominating set algorithm DOM(D)=V and $|D|$ is minimum among all dominating sets for G.

Proof:

We show by induction on the number of cliques visited that there is a minimum dominating set D' which contains D and that DOM(D) contains at least those vertices which appear only in visited cliques.

The basis is trivial. After zero iterations D is empty and clearly contained within any minimum dominating set. There are no vertices which D is required to dominate.

For the inductive step, if no v satisfies the condition of Step [3] in the algorithm the hypothesis continues to hold. Otherwise suppose that x is added to D while visiting C in the clique tree. By the induction hypothesis $D-\{x\}$ is contained within some minimum dominating set D'. If x is in D' we are done. Otherwise let v be the vertex of C which is not dominated by $D-\{x\}$.

The condition on v is that high[v] = depth[C]. Hence we know that v occurs only in cliques within the subtree rooted at C. Let y be any vertex in D' which dominates v. Consider the set $D'' = D'-\{y\}\cup\{x\}$. We claim that D'' is a dominating set for G.

Assume to the contrary that there is a vertex z not dominated by D''. Note that z appears in at least one clique which is not below C in the clique tree or it would be in DOM(D) and hence in DOM(D''). Similarly z cannot be in C or it would be in DOM(x) and hence in DOM(D''). The cliques containing z form a path so z appears in no clique of the subtree rooted at C.

We conclude that y must appear in a clique of the subtree rooted at C because it dominates v but that it must also appear in a clique not in that subtree because it dominates z. The requirement that the cliques containing y form a path forces us to conclude that y actually appears in C.

This produces the desired contradiction because the choice of x dictates that $\text{high}[x] \geqslant \text{high}[y]$ which in turn implies that for vertices appearing in C or in cliques above C, x can substitute for y as a dominator, whereas for vertices appearing only in cliques below C other vertices of D can be substituted. The path constraint on cliques containing z guarantees that these are the only two cases which arise. The induction is now complete. D'' is a dominating set for G, $|D'| = |D''|$ so D'' is a minimum dominating set, and D dominates every vertex appearing only in cliques which have been visited. □

Cockayne, Goodman and Hedetniemi actually solved a more general problem for trees. They allowed a set R of *required vertices* and a set F of *free vertices* to be specified. Required vertices must always be in D while free vertices are not required to be dominated by D. Initializing D to be R and marking all vertices in F plus all those in or adjacent to R as "dominated" converts our algorithm to this purpose.

If G is an interval graph we can certainly apply the directed path graph dominating set algorithm, but there is a substantially simpler method. Instead of building a directed path model, which uses the Booth-Lueker interval graph test as a subroutine, we can simply run the interval graph test and use the data structure it builds to directly find a linear arrangement of the cliques [2]. The bottom-up walk of the clique tree becomes a left-to-right scan of the cliques. In practice we would expect this algorithm to be easier to implement and more efficient at run time than the general directed path graph algorithm, although both are of course asymptotically linear in the size of G.

Finally, it should be noted that every tree is a directed path graph so our algorithm applies, but there is obviously no need to explicitly compute a clique tree. The cliques of a tree are its edges. Visiting the edges in a depth-first order corresponds to our algorithm and in fact is the original algorithm of Cockayne, Goodman and Hedetniemi.

4. A Possible Relationship with Graph Isomorphism

The construction used in Theorem 1 to reduce the general vertex cover problem to the dominating set problem for chordal graphs is the same construction used to prove that isomorphism testing of chordal graphs is complete with respect to graph isomorphism [2, Theorem 5]. This suggests a possible parallel between the two problems. There is such a parallel, but the picture is not quite complete.

The situation is summarized in Figure 3. For the dominating set problem we have just shown that the chordal case is NP-complete and that directed path and interval graphs have linear time algorithms. The status of the problem for undirected path graphs remains open. For graph isomorphism both the chordal and undirected path graphs are known to be isomorphism-complete [1, p. 13] whereas interval graphs have a linear time algorithm [2]. Only the question for directed path graphs remains open for isomorphism.

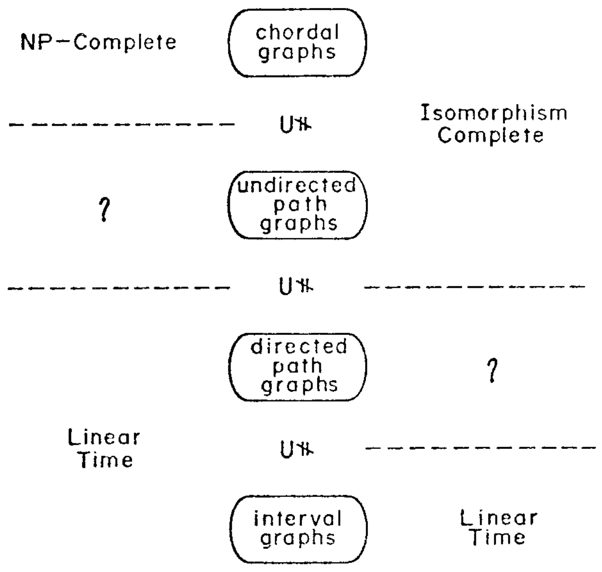

Figure 3. The current status of the dominating set and graph isomorphism problems for chordal graphs.

One possibility is that the two problems are somehow related and therefore the dominating set problem for undirected path graphs is NP-complete and the isomorphism problem for directed path graphs has a linear time (or at least polynomial time) solution. We have not been able to prove or disprove either conjecture, but the following observations may be useful.

If we assume that the two problems are in fact similar we want to prove that the dominating set problem for undirected path graphs is NP-complete and that directed path graphs have a linear time isomorphism test. The isomorphism-completeness proof for undirected path graphs doesn't seem to carry over to an NP-completeness proof because the isomorphism construction does not produce a vertex cover in the original graph, even though it is quite similar to Theorem 1 (the only difference is that the roles of vertices and edges in G are interchanged).

Similarly the linear time isomorphism test for interval graphs follows from the fact that a canonical representation of all possible clique trees is built for the interval graph; the algorithm of Dietz, Furst and Hopcroft does construct a clique tree but not a canonical one. The same is true of Gavril's recognition algorithm. Thus we currently see no obvious way of extending either result.

If the similarity between the two problems turns out to be a red herring we should actually be attempting to prove that the dominating set problem for undirected path graphs has a linear time (or at least polynomial time) algorithm or that directed path graph isomorphism is isomorphism-complete. Our intuition is that neither is the case.

The success of our current algorithm hinges upon the fact that clique paths can only descend from the root and that they are not allowed to travel up one branch and back down another. By way of contrast, the isomorphism-completeness proof for undirected path graphs demands precisely this property of the constructed paths. Again we see no way of extending the results.

Settling these two remaining cases looks like a fine topic for further research.

Acknowledgements

The problem of finding dominating sets for chordal graphs was first suggested to me by Sandra and Steve Hedetniemi who were also kind enough to point out a number of references in the literature.

References

[1] Kellogg S. Booth and Charles J. Colbourn, Problems polynomially equivalent to graph isomorphism, Department of Computer Science, University of Waterloo, CS-77-04 (June 1979).

[2] Kellogg S. Booth and George S. Lueker, Testing for the consecutive ones property, interval graphs, and graph planarity using PQ-tree algorithms, *Journal of Computer and System Sciences 13:3* (December 1976) pp. 335-379.

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

[4] Paul Dietz, Merrick Furst, and John Hopcroft, A linear time algorithm for the generalized consecutive retrieval problem, Department of Computer Science, Cornell University, TR 79-386 (July, 1979).

[5] D.R. Fulkerson and O.A. Gross, Incidence matrices and interval graphs, *Pacific Journal of Mathematics 15* (1965) pp. 835-855.

[6] Michael R. Garey and David S. Johnson, *Computers and Intractability, A Guide to the Theory of NP-Completeness*, W.H. Freeman and Company, San Francisco (1979).

[7] Fanica Gavril, Algorithms for minimum coloring, maximum clique, minimum covering by cliques, and maximum independent set of a chordal graph, *SIAM Journal on Computing 1:2* (1972) pp. 180-187.

[8] Fanica Gavril, The intersection graphs of subtrees in trees are exactly the chordal graphs, *Journal of Combinatorial Theory 16:1* (1974) pp. 47-56.

[9] Fanica Gavril, A recognition algorithm for the intersection graphs of directed paths in directed trees, *Discrete Mathematics 13* (1975) pp. 237-249.

[10] Fanica Gavril, A recognition algorithm for the intersection graphs of paths in trees, *Discrete Mathematics 23* (1978) pp. 221-227.

[11] Frank Harary, *Graph Theory*, Addison-Wesley, Reading, Massachusetts, (1969).

[12] C.G. Lekkerkerker and J.Ch. Boland, Representation of a finite graph by a set of intervals on the real line, *Fundamenta Mathematicae 51* (1962) pp. 45-64.

[13] George S. Lueker and Kellogg S. Booth, A linear time algorithm for deciding interval graph isomorphism, *Journal of the ACM 26:2* (April 1979) pp. 183-195.

[14] K.S. Natarajan and Lee J. White, Optimum domination in weighted trees, *Information Processing Letters 7:6* (October 1978) pp. 261-265.

[15] Peter L. Renz, Intersection representation of graphs by arcs, *Pacific Journal of Mathematics 34:2* (1970) pp. 501-510.

[16] Donald J. Rose, R. Endre Tarjan and George S. Lueker, Algorithmic aspects of vertex elimination on graphs, *SIAM Journal on Computing 5:2* (1976) pp. 266-283.

[17] M. Truszczynski, The theorem characterizing the acyclic families of sets, ICS Polish Academy of Sciences Reports 314 (1978).