Mathematical Social Sciences 16 (1988) 267-279
North-Holland

DOMINATING SETS IN SOCIAL NETWORK GRAPHS

Laura L. KELLEHER
Massachusetts Maritime Academy, Buzzards Bay, MA 02532, U.S.A.

Margaret B. COZZENS
Northeastern University, Boston, MA 02115, U.S.A.

Communicated by W.H. Batchelder
Received 19 June 1987
Revised 4 April 1988

This paper presents an approach to the study of the structure of social networks using dominating sets of graphs. A method is presented for partitioning the vertices of a graph using dominating vertices. For certain classes of graphs this is helpful in determining the underlying structure of the corresponding social network. An extension of this technique provides a method of studying the structure of directed graphs and directed social networks. Minimum dominating sets are related to statuses and structurally equivalent sets.

Key words: social networks; graph.

1. Introduction

This paper presents an approach to the study of the structure of social networks using dominating sets of graphs. A set of vertices in a simple, undirected graph is a *dominating set* if every vertex in the graph which is not in the dominating set is adjacent to one or more of the vertices in the dominating set. Dominating sets in graphs have applications in a wide variety of fields, including facility location problems and communication theory. For background in graph theory see Golumbic (1980), Harary (1969) and Roberts (1976). For more information on dominating sets in graphs see Cockayne et al. (1980), Cockayne and Hedetniemi (1977), Cozzens and Kelleher (to appear), Kelleher (1985) and Laskar and Hedetniemi (1983).

In studying the structure of a social network the concepts of status, clique and structurally equivalent set are important, and several variants of each have been defined in the literature. Throughout this paper the definitions of these concepts follow those used by Batchelder and Lefebvre (1982). Consider a social network that consists of a set of actors and a property that must either hold or not hold between a pair of actors. A social network graph can be constructed in which the vertices represent the actors in the social network and there is an edge between two vertices

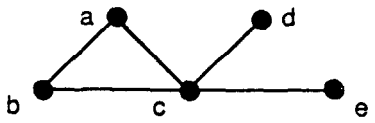

Fig. 1. A social network graph.

if and only if the property is present between the two actors. Let V denote the set of vertices of the graph and, for each $v \in V$, let $N(v)$ denote the *neighborhood* of vertex v, the set of vertices adjacent to v.

A *status* is a set of two or more actors in a network that have smilar ties or relationships to actors outside the set. Thus, in a social network graph, a status is a set S of two or more vertices such that for all $x, y \in S$, $N(x) \cap (V - S) = N(y) \cap (V - S)$. A *social network clique* is a set of actors in a network that have the same tie between all actors within the set. In the graph theoretic approach, then, a social network clique is a set of vertices that induces either a complete subgraph (a set of vertices with every pair of vertices adjacent) or an independent set (a set of vertices with no two vertices adjacent). Since in graph theory a clique induces a complete subgraph, social network cliques include cliques in the social network graph as well as independent sets in the social network graph. A pair of actors is *structurally equivalent* if they have identical ties to each of the actors in the network. Thus, if $x, y \in V$, then x and y are *structurally equivalent* if either $N(x) = N(y)$ or $N(x) \cup \{x\} = N(y) \cup \{y\}$. A set is a *structurally equivalent set* if its elements are pairwise structurally equivalent. A *maximal structurally equivalent set* is a structurally equivalent set that is not a subset of any other structurally equivalent set. A structurally equivalent set is both a status and a social network clique.

In the graph in Fig. 1, sets $\{a, b, d\}$ and $\{d, e\}$ are statuses, but $\{a, d\}$ is not a status. The set $\{a, d, e\}$ is an independent set while $\{a, b, c\}$ is a clique since it induces a complete subgraph. Sets $\{a, b, c\}$ and $\{a, d, e\}$ are both social network cliques. Sets $\{a, b\}$ and $\{d, e\}$ are maximal structurally equivalent sets.

The *complement*, $\bar{G}$, of a graph G has the same vertex set as G and two vertices are adjacent in $\bar{G}$ if and only if they are not adjacent in G. The graph in Fig. 2 is the complement of the graph in Fig. 1. It can easily be shown that the statuses, social network cliques and structurally equivalent sets of a graph are the same as those of the complement of the graph.

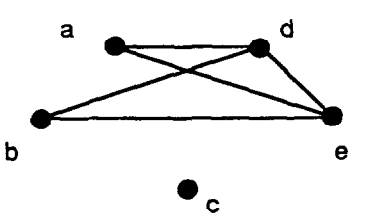

Fig. 2. The complement of a graph.

In the following section a method is presented for partitioning the vertices of a graph using dominating vertices. For certain classes of graphs this is helpful in determining the structure of the underlying social network. In § 3 an extension of this technique provides a method of studying the structure of directed graphs and directed social networks. The final section of the paper shows how minimum dominating sets are related to statuses and structurally equivalent sets.

2. Dominating vertices in social network graphs

Single vertices which dominate a graph can be helpful in determining the underlying structure of the graph. A *dominating vertex* in a graph is a vertex adjacent to all other vertices in the graph. Batchelder and Lefebvre (1982) presented a method of generating the set of all statuses and the set of all maximal structurally equivalent sets for a specific class of graphs. They first coupled a graph with its complement to form what they called a completed graph. Then, using the connected components of the completed graph and its subgraphs they decomposed the completed graph into a nesting of partitions, or a taxonomy, of completed subgraphs. However, by using dominating vertices it is possible to partition the vertices from the graph itself rather than from the completed graph. Further, the nesting of the partitions of the vertex set suggests that it may be helpful to consider an organization of these sets into a structure tree, called the S-tree of the graph. For many graphs this organization provides another method for determining the set of all statuses and the set of all maximal structurally equivalent sets.

Algorithm 1, given in pseudo-Pascal in Appendix A, uses dominating vertices to partition the vertex set and to construct the S-tree of a graph. The root of the S-tree is the vertex set of the graph and the sets of vertices from the partitioning of the graph become the vertices of the next level of the tree. The leaves of the tree are either singletons or sets whose induced subgraph is such that both it and its complement are connected.

Consider the graph shown in Fig. 1. The root of the S-tree for this graph is $\{a, b, c, d, e\}$, the vertex set of the graph. Since vertex c is the only dominating vertex of the graph, the next level of the S-tree consists of $\{c\}$ and $\{a, b, d, e\}$, the set of

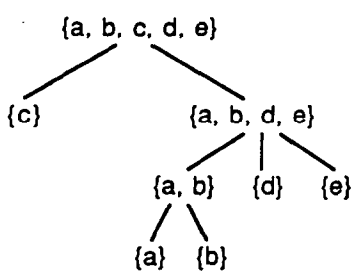

Fig. 3. The S-tree of a graph.

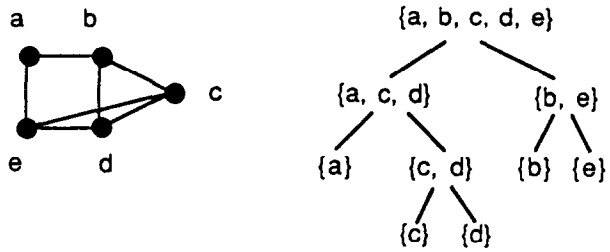

Fig. 4. A graph and its S-tree.

the remaining vertices of the graph. With vertex c and all its incident edges removed from the graph, the ramaining graph consists of three components. These components are included in the S-tree of the graph as the children of $\{a, b, d, e\}$. The subgraph induced by the set $\{a, b\}$, the only non-singleton component, has two dominating vertices. Thus, the S-tree is completed by adding $\{a\}$ and $\{b\}$ as the children of $\{a, b\}$. The S-tree constructed by this algorithm for the graph in Fig. 1 is shown in Fig. 3. Note that this tree is the S-tree of the graph in Fig. 2, the complement of the graph in Fig. 1.

The computational complexity of Algorithm 1 can be found by considering the number of levels of the tree to be constructed and the complexity of constructing each level of the tree. Using a standard method of finding the components of a graph, each level of the S-tree can be computed in n^2 time, where n is the number of vertices in the graph. Since there are at most $\log n$ levels of the tree, Algorithm 1 runs in $O(n^2 \log n)$ time.

Other examples of S-trees of graphs constructed by this algorithm are given in Figs. 4 and 5. Note that in the construction it is necessary to consider the complement of each of these graphs. The graph in Fig. 4 has no dominating vertex but its complement has two components. Thus the children of the root of the S-tree for this graph are the sets of the vertices of the components of the complement of the original graph. The graph in Fig. 5 has vertex a as a single dominating vertex. Therefore the children of the root of this S-tree are $\{a\}$ and the set of the remaining vertices, $\{b, c, d, e, f\}$. The subgraph induced by this set is a single component with no dominating vertex. Further, the complement of this subgraph is a single component with no dominating vertex. Thus the tree in Fig. 5 has a leaf which is not a singleton set.

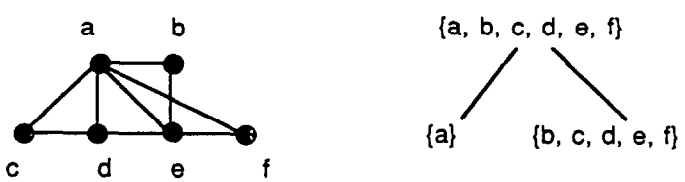

Fig. 5. Another graph and its S-tree.

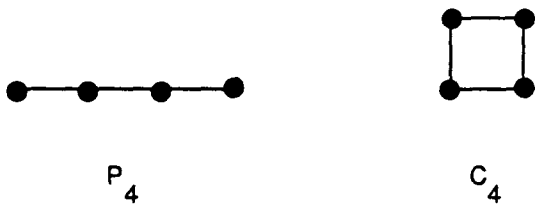

Fig. 6. The path and cycle on four vertices.

In implementing Algorithm 1 it is necessary to consider the complement of any component which does not have a dominating vertex. A forbidden subgraph characterization of a class of graphs which have a dominating vertex is presented in the following theorem. The notation P_n denotes the path on n distinct vertices and C_n denotes the cycle on n vertices. P_4 and C_4 are shown in Fig. 6.

Theorem 1. (Wolk, 1965.) *If G is a finite connected graph with no induced P_4 or C_4 then G has a dominating vertex.*

From this theorem, it follows that each component of a graph with no induced P_4 or C_4 has a dominating vertex. Thus, in the construction of the S-tree for graphs with no induced P_4 or C_4 it is not necessary to consider the complement of any component. Further, if a connected graph has no induced P_4 then the complement of the graph is not connected. Thus, the leaves of the S-tree of a graph with no induced P_4 are singleton sets. For further background on the class of graphs with no induced P_4 see Corneil et al. (1981). The following theorem shows how the set of statuses of a graph with no induced P_4 can be determined from the S-tree of the graph. This theorem corresponds directly to a theorem in Batchelder and Lefebvre (1982) for completed graphs.

Theorem 2. *Let G be a finite graph with no induced P_4 and S-tree formed by Algorithm 1. The set of all statuses of G is the set formed by taking the unions of two or more branches of the S-tree with a common parent.*

The following result for binary S-trees follows easily from Theorem 2.

Corollary 2.1. *Let G be a finite graph with no induced P_4 and S-tree formed by Algorithm 1. If the S-tree of G is a binary tree then the set of all statuses of G is the set of all vertices of the S-tree that are not leaves.*

The graphs in Figs. 1, 2 and 4 have no induced P_4. The set of all statuses of the graph in Fig. 1 (and the graph in Fig. 2) with non-binary S-tree shown in Fig. 3 is $\{\{a,b,c,d,e\},\{a,b,d,e\},\{a,b\},\{d,e\},\{a,b,d\},\{a,b,e\}\}$. The S-tree for the graph in Fig. 4 is a binary tree. The set of all statuses of this graph is

$\{\{a,b,c,d,e\},\{a,c,d\},\{c,d\},\{b,e\}\}$. The graph in Fig. 5 has an induced P_4, for example $\{c,d,e,f\}$. Therefore, the set of *all* statuses for this graph cannot be found by taking unions of two or more branches of the S-tree with a common parent. For a graph with an induced P_4, the set formed in this way consists of sets that are statuses of the graph, but there may be statuses of the graph that are not recorded in the set. The set $\{b,f\}$ is a status of the graph in Fig. 5 that is not found in this manner.

Batchelder and Lefebvre (1982) used an equivalence relation on V based on the set of all two element statuses of a graph to generate the set of maximal structurally equivalent sets of the graph. Since the set of all two element statuses of a graph with no induced P_4 can be found from the S-tree of the graph this method can be used to find the set of maximal structurally equivalent sets directly from such a graph. However, as indicated below, the structure of the S-tree itself provides another method of determining the set of all maximal structurally equivalent sets of a graph with no induced P_4.

Theorem 3. *Let G be a finite graph with no induced P_4 and with S-tree formed by Algorithm* 1. *The maximal structurally equivalent sets of G are formed by taking the union of all of the vertices which form singleton sets with a common parent in the S-tree.*

Proof. All of the vertices which form singleton sets with a common parent in the S-tree are adjacent to all (or none) of the other vertices in the parent set. Therefore, the union of these vertices is a social network clique. By Theorem 2, the union of these vertices is a status. Since every vertex which is adjacent to all (or none) of the other vertices in the parent set is added to the S-tree as a singleton set, the union of such vertices is a maximal structurally equivalent set. □

Corollary 3.1. *Let G be a finite graph with no induced P_4 and whose S-tree formed by Algorithm* 1 *is a binary tree. The maximal structurally equivalent sets of G are the vertices of the tree whose children are all singleton sets.*

From the S-tree in Fig. 3, $\{a,b\}$ and $\{d,e\}$ are maximal structurally equivalent sets for the graph in Fig. 1 (and the graph in Fig. 2). From the binary S-tree in Fig. 4, $\{b,e\}$ and $\{c,d\}$ are maximal structurally equivalent sets for the graph in Fig. 4.

The formation of the S-tree of a graph can be useful, therefore, in studying the structure of the graph and of the corresponding social network. Further, for a certain class of graphs this tree is helpful in determining the threshold dimension of the graph (Cozzens, to appear). An extension of these techniques provides a method of studying the structure of directed graphs and of directed social networks. This is considered in the following section.

3. Directed graphs

Social networks may involve a directed relationship between pairs of actors. The formation of the S-tree of a directed graph provides an approach to the study of the structure of such a network.

A *directed graph* consists of a finite set of vertices and a collection of ordered pairs of distinct vertices. Any such ordered pair (u, v) is called an *arc* of the directed graph. The arc (u, v) goes from u to v. Vertex u is *adjacent to* vertex v and vertex v is *adjacent from* vertex u. A directed graph is shown in Fig. 7.

The definitions of status, social network clique and structurally equivalent set of a social network graph can be extended for a directed graph. A *directed status* is a set S of vertices of a directed graph with vertex set V such that for all $x, y \in S$, and $z \in V - S$, if x is adjacent to z then y is adjacent to z and if x is adjacent from z then y is adjacent from z. Vertices in a *directed social network clique* form a set which must be adjacent both to and from all or none of the other vertices in the set. Vertices in a *directed structurally equivalent set* form a social network clique in which all of the vertices are adjacent to and from exactly the same vertices outside of the set. Consider the directed graph in Fig. 7. Set $\{a, b\}$ is a directed status but $\{a, b, d, e\}$ is not. Since vertices a and b also form a directed social network clique, $\{a, b\}$ is a directed structurally equivalent set.

A graph G can be associated with a directed graph D that has the same vertex set V as D and for $u, v \in V$, $\{u, v\}$ is an edge of G if and only if (u, v) or (v, u) or both is an arc of D. A graph G formed in this way from a directed graph D is called *the non-directed form of* D. A status (or structurally equivalent set) of G may not necessarily be a directed status (or directed structurally equivalent set) of D. However, by definition, any directed status of D must also be a status of G. Similarly, any directed structurally equivalent set of D must be a structurally equivalent set of G. Thus, one approach to finding the directed statuses and directed structurally equivalent sets of D is to consider the statuses and maximal structurally equivalent sets of G. Since for many graphs these can be found by using the S-tree of the graph, this method can be helpful in identifying the structure of the directed graph.

For a directed graph D, with non-directed form G, the formation of the S-tree for G is extended in Algorithm 2, given in Appendix B, to find a related S-tree for D. In Algorithm 2 the S-tree of the non-directed form of a directed graph is constructed. As each non-singleton vertex is added to the S-tree it is paired with the set

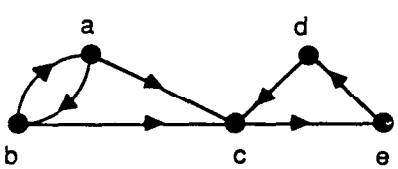

Fig. 7. A directed graph.

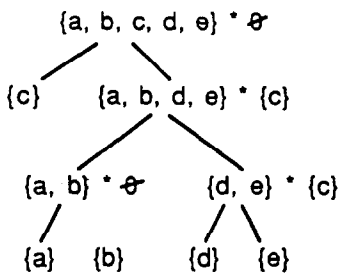

Fig. 8. The S-tree of a directed graph.

of vertices that are adjacent to and or from some but not all of the vertices in the vertex of the S-tree. The S-tree of the directed graph in Fig. 7 is shown in Fig. 8.

In implementing Algorithm 2, the formation of the set associated with each non-singleton vertex of the S-tree requires $O(n)$ comparisons, where n is the number of vertices in the graph. Since the formation of the S-tree runs in $O(n^2 \log n)$ time, Algorithm 2 runs in $O(n^2 \log n + n)$ or $O(n^2 \log n)$ time.

As the following theorem indicates, for some directed graphs, the S-tree can be used to find the set of directed statuses of the directed graph.

Theorem 4. *Let D be a directed graph whose non-directed form has no induced P_4 and whose S-tree formed by Algorithm 2 is a binary tree. The set of all directed statuses of D is the set of vertices of the S-tree that are not leaves and whose associated set is the empty set.*

Proof. By Corollary 2.1, the set of all statuses of the non-directed form of D is the set of all vertices in the S-tree of that graph that are not leaves. Let S be a vertex in the S-tree that is not a leaf and whose associated set is the empty set. Since there are no vertices in the directed graph that are adjacent to or from some but not all the vertices of S, all of the vertices in S must either be adjacent to or from all or none of the vertices not in S. Thus, S is a directed status of D. □

Further, the S-tree of some directed graphs can be used to find the set of all maximal directed structurally equivalent sets of the graphs. For a directed graph whose non-directed form has no induced P_4 the directed structurally equivalent vertices will be singleton sets with a common parent in the S-tree. Such vertices can be checked to see if their union forms a directed status and if they are mutually adjacent to and from all or none of the others. For the directed graph in Fig. 7 sets $\{a, b, c, d, e\}$ and $\{a, b\}$ are directed statuses while set $\{a, b\}$ is a directed maximal structurally equivalent set.

In the final section a further connection between dominating sets and the structure of social network graphs is explored.

4. Minimum dominating sets

The algorithms presented in this paper have used single vertices which dominate a graph or a directed graph to assist in identifying the structure of the corresponding social network. A set of a single dominating vertex is clearly a smallest possible dominating set of a graph. However, such a set cannot be a status of a graph since by definition a status must contain at least two vertices. A *minimum dominating set* is a dominating set that contains the least number of vertices of all of the dominating sets of the graph. A relationship between the minimum dominating set and the statuses of a connected graph is established in the following theorem.

Theorem 5. *If a minimum dominating set S of a finite connected graph G is a status of G then S is an independent dominating set of size two.*

Proof. For a graph G with n vertices and no isolated vertices the size of the smallest dominating set is less than or equal to $n/2$ (Ore, 1962). Since S is a minimum dominating set of a connected graph G, and a connected graph has no isolated vertices, G must contain some vertex that is not in S. Let v be such a vertex. Since S is a dominating set of G, there must be a vertex in S that is adjacent to v. However, by the definition of a status of a graph, if S, is a status of G and there is a vertex of S that is adjacent to $v \in V-S$, then every vertex of S is adjacent to v. Similarly, every vertex in S must be adjacent to every vertex in $V-S$. Suppose S is of size one. Since by definition a status must have at least two vertices S cannot be a status. Suppose S is of size greater than or equal to three. Let x be any vertex in S. Vertex x is adjacent to every vertex in $V-S$. Vertex v is adjacent to every vertex in S. Therefore, $\{x, v\}$ is a dominating set of G that is smaller than S, a contradiction to the assumption that S is a minimum dominating set of G. Thus, S must be of size two. If the two vertices of S are adjacent then each of these vertices is a dominating vertex of G, a contradiction to the assumption that S is a minimum dominating set of G. Therefore, S is an independent dominating set of size two. □

Since a structurally equivalent set is both a status and a social network clique, the following relationship between the minimum dominating sets and the structurally equivalent sets of a graph follows easily from the previous theorem.

Corollary 5.1. *Let G be a connected graph with n vertices. If a minimum dominating set S is also a structurally equivalent set then S consists of two independent vertices each of which has degree $n-1$.*

In this paper the study of the structure of social networks has been approached through the use of dominating sets of social network graphs. Questions for further research include: How can the set of all statuses and the maximal structurally equivalent sets of a graph with an induced P_4 be found? Are there other classes of

graphs for which the statuses and the maximal structurally equivalent sets can be found easily? How can the directed statuses and the maximal directed structurally equivalent sets of a directed graph with a non-binary S-tree be found?

Appendix A. Algorithm 1

Input. The adjacency lists of a graph.
Output. The S-tree of the graph.
Method. The root of the tree is the vertex set of the graph and this becomes the first parent vertex. The tree is built recursively by locating dominating vertices of components of the parent vertex. If a component does not have any dominating vertices the complement of the component it considered.

```
procedure component (v);
  begin
    Q ← {v};
    C ← ∅;
    while Q ≠ ∅
      begin
        choose x ∈ Q;
        Q ← Q − {x};
        C ← C ∪ {x};
        for all y ∈ A(x) if y ∉ C then Q ← Q ∪ {y}
      end;
    return C
  end;

procedure partition (parent);
  begin
    P1 ← parent;
    while P1 ≠ ∅
      begin
        choose v ∈ P1;
        component (v);
        if C ≠ parent then add C as a child of parent;
        P1 ← P1 − C;
        C1 ← C;
        R ← ∅;
        D ← ∅;
        while C1 ≠ ∅
          begin
            choose v ∈ C1;
```

```
            if A(v) = C − {v} then
              begin
                add {v} as a child of C;
                D ← D ∪ {v}
              end
            else R ← R ∪ {v}
          end;

        if D ≠ ∅ then
          begin
            if R ≠ ∅ then add R as a child of C;
            C ← R;
            for each v ∈ R
              A(v) ← A(v) − (D ∩ A(v))
          end
          else
            begin
              for each x ∈ C
                A(x) ← C − A(x);
                Com ← C;
                choose v ∈ Com;
                component (v);
                if C = Com then
                  begin
                    add C as a child of parent;
                    C ← ∅
                  end
            end;
          if C ≠ ∅ then partition (C)
      end
    end;

begin
  root ← V;
  parent ← root;
  partition (parent);
  return S-tree
end.
```

Appendix B. Algorithm 2

Input. The adjacency lists, $A(v) = A\,\text{in}(v) \cup A\,\text{out}(v)$, $v \in V$ of a directed graph with vertex set V.

Output. The S-tree of the directed graph.

```
procedure check (X);
  begin
    X' ← ∅;
    N ← (parent ∪ parent') − X;
    for each x ∈ N
      if (A in(x) ∩ X ≠ X or A in(x) ∩ X ≠ ∅)
      or (A out(x) ∩ X ≠ X or A out(x) ∩ X ≠ ∅)
      then X' ← X' ∪ {x};
      return X'
  end;

procedure partition (parent);
  begin
    follow procedure partition for Algorithm 1
    replacing 'add C(or R) as a child of parent' with:
      if |C| (or |R|) > 1 then
        begin
          check (C) (or R);
          add C*C' (or R*R') as a child of parent
        end;
  end;

begin
  root ← V*∅;
  parent ← root;
  partition (parent);
  return S-tree
end.
```

References

W. Batchelder and V. Lefebvre, A mathematical analysis of a natural class of partitions of a graph, J. Math. Psychol. 26 (1982) 124–148.

E.J. Cockayne, R.M. Dawes and S.T. Hedetniemi, Total domination in graphs, Networks 10 (1980) 211–219.

E.J. Cockayne and S.T. Hedetniemi, Toward a theory of domination in graphs, Networks 7 (1977) 247–261.

D.G. Corneil, H. Lerchs and L.S. Burlingham, Complement reducible graphs, Discrete Appl. Math. 3 (1981) 163–174.

M.B. Cozzens, Algorithms to compute the threshold dimension of some clique dominated graphs (to appear).

M.B. Cozzens and L.L. Kelleher, Dominating cliques in graphs, Ann. Discrete Math. (to appear).

M.C. Golumbic, Algorithmic Graph Theory and Perfect Graphs (Academic Press, New York, 1980).

F. Harary, Graph Theory (Addison Wesley, Reading, MA, 1969).

Kelleher, L.L. Domination in graphs and its application to social network theory, Ph.D. Thesis, Northeastern University, Boston, MA, 1985.

R. Laskar and S.T. Hedetniemi, Connected domination in graphs, Clemson University, Dept. Mathematical Sci., Tech. Report 414 (1983).

O. Ore, Theory of graphs, Amer. Math. Soc. Colloq. Publ., 38, Providence, RI (1962).

F.S. Roberts, Discrete Mathematical Models with Applications to Social Biological, and Environmental Problems (Prentice-Hall, Englewood Cliffs, NJ, 1976).

E.S. Wolk, A note on 'The Comparability Graph of a Tree,' Proc. A.M.S. 16 (1965) 17–20.