PREPRINT

國立臺灣大學 數學系 預印本 Department of Mathematics, National Taiwan University

Algorithmic Aspects of Domination in Graphs

Gerard Jennhwa Chang

October 6, 2011

Algorithmic Aspects of Domination in Graphs

Gerard Jennhwa Chang

Department of Mathematics
National Taiwan University
Taipei 10617, Taiwan

gjchang@math.ntu.edu.tw

Abstract

Domination in graph theory has many applications in the real world such as location problems. A dominating set of a graph $G = (V, E)$ is a subset D of V such that every vertex not in D is adjacent to at least one vertex in D. The domination problem is to determine the domination number $\gamma(G)$ of a graph G that is the minimum size of a dominating set of G. Although many theoretic theorems for domination and its variations have been established for a long time, the first algorithmic result on this topic was given by Cockayne, Goodman and Hedetniemi in 1975. They gave a linear-time algorithm for the domination problem in trees by using a labeling method. On the other hand, at about the same time, Garey and John constructed the first (unpublished) proof that the domination problem is NP-complete. Since then, many algorithmic results are studied for variations of the domination problem in different classes of graphs. This chapter is to survey the development on this line during the past 36 years. Polynomial-time algorithms using labeling method, dynamic programming method and primal-dual method are surveyed on trees, interval graphs, strongly chordal graphs, permutation graphs, cocomparability graphs and distance-hereditary graphs. NP-completeness results on domination are also discussed.

1 Introduction

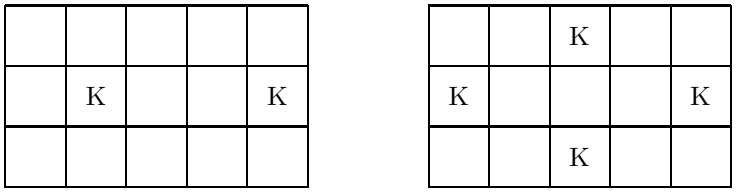
Graph theory was founded by Euler [90] in 1736 as a generalization of the solution to the famous problem of the Königsberg bridges. From 1736 to 1936, the same concept as graph, but under different names, was used in various scientific fields as models of real world problems, see the historic book by Biggs, Lloyd and Wilson [23]. This chapter intents to survey the domination problem in graph theory from an algorithmic point of view.

Domination in graph theory is a natural model for many location problems in operations research. As an example, consider the following fire station problem. Suppose a county has decided to build some fire stations, which must serve all of the towns in the county. The fire stations are to be located in some towns so that every town either has a fire station or is a neighbor of a town which has a fire station. To save money, the county wants to build the minimum number of fire stations satisfying the above requirements.

Domination has many other applications in the real world. The recent book by Haynes, Hedetniemi and Slater [114] illustrates many interesting examples, including dominating queens, sets of representatives, school bus routing, computer communication networks, (r, d)-configurations, radio stations, social network theory, land surveying, kernels of games, etc.

Among them, the classical problems of covering chessboards by the minimum number of chess pieces are important in stimulating the study of domination, which commenced in the early 1970's. These problems certainly date back to De Jaenisch [85] and have been mentioned in the literature frequently since that time.

A simple example is to determine the minimum number of kings dominating the entire chessboard. The answer to an $m \times n$ chessboard is $\lceil \frac{m}{3} \rceil \lceil \frac{n}{3} \rceil$. In the Chinese chess game, a king only dominates the four neighbor cells which have common sides with the cell the king lies in. In this case, the Chinese king domination problem for an $m \times n$ chessboard is harder. Figure 1 shows optimal solutions to both cases for a 3×5 board.



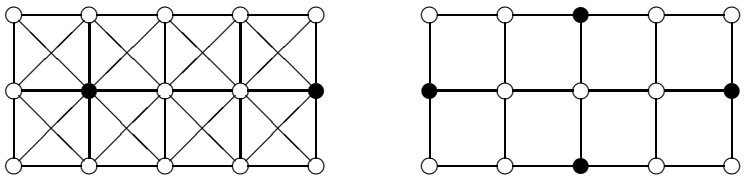
(a) chessboard (b) Chinese chessboard

Figure 1: King domination on a 3×5 chessboard.

The above problems can be abstracted into the concept of domination in terms of graphs as follows. A *dominating set* of a graph $G = (V, E)$ is a subset D of V such that every vertex not in D is adjacent to at least one vertex in D. The *domination number* $\gamma(G)$ of a graph G is the minimum size of a dominating set of G.

For the fire station problem, consider the graph G having all towns of the county as its vertices and a town is adjacent to its neighbor towns. The fire station problem is just the domination problem, as $\gamma(G)$ is the minimum number of fire stations needed.

For the king domination problem on an $m \times n$ chessboard, consider the king's graph G whose vertices correspond to the mn squares in the chessboard and two vertices are adjacent if and only if their corresponding squares have a common point. For the Chinese king domination problem, the vertex set is the same but two vertices are adjacent if and only if their corresponding squares have a common side. Figure 2 shows the corresponding graphs for the king and the Chinese king domination problems on a 3×5 chessboard. The king domination problem is just the domination problem, as $\gamma(G)$ is the minimum number of kings needed. Black vertices in the graph form a minimum dominating set.



(a) for chess (b) for Chinese chess

Figure 2: King's graphs for the chess and the Chinese chess.

Although many theoretic theorems for the domination problem have been established for a long time, the first algorithmic result on this topic was given by Cockayne, Goodman and Hedetniemi [58] in 1975. They gave a linear-time algorithm for the domination problem in trees by using a labeling method. On the other hand, at about the same time Garey and Johnson (see [101]) constructed the first (unpublished) proof that the domination problem is NP-complete for general graphs. Since then, many algorithmic results are studied for variations of the domination problem in different classes of graphs. The purpose of this chapter is to survey these results.

This chapter is organized as follows. Section 2 gives basic definitions and notation. In particular, the classes of graphs surveyed in this chapter are introduced. Among them, trees and interval graphs are two basic classes in the study of domination. While a tree can be viewed as many paths starting from a center with different branches, an interval graph is a "thick path" in the sense that a vertex of a path is replaced by a group of vertices tied together. Section 3 investigates different approaches for domination in trees, including labeling method, dynamic programming method, primal-dual approach and others. These techniques are used not only for trees, but also for many other classes of graphs in the study of domination as well as many other optimization problems. Section 4 is for domination in interval graphs. It is in general not clear to which classes of graphs the results in trees and interval graphs can be extended. For some classes of graphs the domination problem becomes NP-complete, while for some classes it is polynomially solvable. Section 5 surveys NP-completeness results on domination, for which chordal graphs play an important role. The remaining

sections are for classes of graphs in which the domination problem is solvable, including strongly chordal graphs, permutation graphs, cocomparability graphs and distance-hereditary graphs.

2 Definitions and notation

2.1 Graph terminology

A *graph* is an ordered pair $G = (V, E)$ consisting of a finite nonempty set V of *vertices* and a set E of 2-subsets of V, whose elements are called *edges*. Sometimes $V(G)$ is used for the vertex set and $E(G)$ for the edge set of a graph G. A graph is *trivial* if it contains only one vertex. For any edge $e = \{u, v\}$, it is said that vertices u and v are *adjacent*, and that vertex u (respectively, v) and edge e are *incident*. Two distinct edges are *adjacent* if they contain a common vertex. It is convenient to henceforth denote an edge by uv rather than $\{u, v\}$. Notice that uv and vu represent the same edge in a graph.

Two graphs $G = (V, E)$ and $H = (U, F)$ are *isomorphic* if there exists a bijection f from V to U such that $uv \in E$ if and only if $f(u)f(v) \in F$. Two isomorphic graphs are essentially the same as one can be obtained from the other by renaming vertices.

It is often useful to express a graph G diagrammatically. To do this, each vertex is represented by a point (or a small circle) in the plane and each edge by a curve joining the points (or small circles) corresponding to the two vertices incident to the edge. It is convenient to refer to such diagram of a graph as the graph itself. In Figure 3, a graph G with vertex set $V = \{u, v, w, x, y, z\}$ and edge set $E = \{uw, ux, vw, wx, xy, wz, xz\}$ is shown.

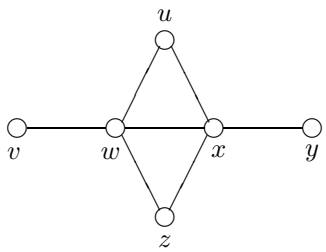


Figure 3: A graph G with 6 vertices and 7 edges.

Suppose A and B are two sets of vertices. The *neighborhood* $N_A(B)$ of B in A is the set of vertices in A that are adjacent to some vertex in B, i.e.,

$$N_A(B) = \{u \in A : uv \in E \text{ for some } v \in B\}.$$

The *closed neighborhood* $N_A[B]$ of B in A is $N_A(B) \cup B$. For simplicity, $N_A(v)$ stands for $N_A(\{v\})$, $N_A[v]$ for $N_A[\{v\}]$, $N(B)$ for $N_V(B)$, $N[B]$ for $N_V[B]$, $N(v)$ for $N_V(\{v\})$ and $N[v]$ for $N_V[\{v\}]$. The notion $u \sim v$ stands for $u \in N[v]$.

The *degree* $\deg(v)$ of a vertex v is the size of $N(v)$, or equivalently, the number of edges incident to v. An *isolated vertex* is a vertex of degree zero. A *leaf* (or *end vertex*) is a vertex of degree one. The minimum degree of a graph G is denoted by $\delta(G)$ and the maximum degree by $\Delta(G)$. A graph G is r-regular if $\delta(G) = \Delta(G) = r$. A 3-regular graph is also called a *cubic* graph.

A graph $G' = (V', E')$ is a *subgraph* of another graph $G = (V, E)$ if $V' \subseteq V$ and $E' \subseteq E$. In the case of $V' = V$, G' is called a *spanning subgraph* of G. For any nonempty subset S of V, the (*vertex*) *induced subgraph* $G[S]$ is the graph with vertex set S and edge set

$$E[S] = \{uv \in E : u \in S \text{ and } v \in S\}.$$

A graph is *H-free* if it does not contain H as an induced subgraph. The *deletion* of S from $G = (V, E)$, denoted by $G - S$, is the graph $G[V \setminus S]$. $G - v$ is a short notation for $G - \{v\}$ when v is a vertex in G. The *deletion* of a subset F of E from $G = (V, E)$ is the graph $G - F = (V, E \setminus F)$. $G - e$ is a short notation for $G - \{e\}$ if e is an edge of G. The *complement* of a graph $G = (V, E)$ is the graph $\overline{G} = (V, \overline{E})$, where

$$\overline{E} = \{uv \notin E : u, v \in V \text{ and } u \neq v\}.$$

Suppose $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ are two graphs with $V_1 \cap V_2 = \emptyset$. The *union* of G_1 and G_2 is the graph $G_1 \cup G_2 = (V_1 \cup V_2, E_1 \cup E_2)$. The *join* of G_1 and G_2 is the graph $G_1 + G_2 = (V_1 \cup V_2, E_+)$, where

$$E_+ = E_1 \cup E_2 \cup \{uv : u \in V_1 \text{ and } v \in V_2\}.$$

The *Cartesian product* of G_1 and G_2 is the graph $G_1 \square G_2 = (V_1 \times V_2, E_\square)$, where

$$V_1 \times V_2 = \{(v_1, v_2) : v_1 \in V_1 \text{ and } v_2 \in V_2\},$$

$$E_\square = \{(u_1, u_2)(v_1, v_2) : (u_1 = v_1, u_2v_2 \in E_2) \text{ or } (u_1v_1 \in E_1, u_2 = v_2)\}.$$

In a graph $G = (V, E)$, a *clique* is a set of pairwise adjacent vertices in V. An *i-clique* is a clique of size i. A 2-clique is just an edge. A 3-clique is called a *triangle*. A *stable* (or *independent*) *set* is a set of pairwise nonadjacent vertices in V.

For two vertices x and y of a graph, an *x-y walk* is a sequence $x = x_0$, x_1, …, $x_n = y$ such that $x_{i-1}x_i \in E$ for $1 \leq i \leq n$, where n is called the *length* of the walk. In a walk x_0, x_1, …, x_n, a *chord* is an edge x_ix_j with $|i - j| \geq 2$. A *trail* (*path*) is a walk in which all edges (vertices) are distinct. A *cycle* is an x-x walk in which all vertices are distinct except the first vertex is equal to the last. A graph is *acyclic* if it does not contain any cycle.

A graph is *connected* if for any two vertices x and y, there exists an x-y walk. A graph is *disconnected* if it is not connected. A (*connected*) *component* of a graph is a maximal subgraph which is connected. A *cut-vertex* is a vertex whose

deletion from the graph results in a disconnected graph. A *block* of a graph is a maximal connected graph which has no cut-vertices.

The *distance* $d(x, y)$ from a vertex x to another vertex y is the minimum length of an x-y path; and $d(x, y) = \infty$ when there is no x-y path.

Digraphs or *directed graphs* can be defined similar to graphs except that an edge (u, v) is now an ordered pair rather than a 2-subset. All terms in graphs can be defined for digraphs with suitable modifications by taking into account the directions of edges.

An *orientation* of a graph $G = (V, E)$ is a digraph (V, E') such that for each edge $\{u, v\}$ of E exactly one of (u, v) and (v, u) is in E' and the edges in E' all come from this way.

2.2 Variations of domination

Due to different requirements in the applications, people have studied many variations of the domination problem. For instance, in the queen's domination problem, one may ask the additional property that two queens don't dominate each other, or any two queens must dominate each other. The following are most commonly studied variants of domination.

Recall that a dominating set of a graph $G = (V, E)$ is a subset D of V such that every vertex not in D is adjacent to at least one vertex in D. This is equivalent to that $N[x] \cap D \neq \emptyset$ for all $x \in V$ or $\cup_{y \in D} N[y] = V$.

A dominating set D of a graph $G = (V, E)$ is *independent, connected, total* or *perfect* (*efficient*) if $G[D]$ has no edge, $G[D]$ is connected, $G[D]$ has no isolated vertex or $|N[v] \cap D| = 1$ for any $v \in V \setminus D$. An *independent* (respectively, *connected* or *total*) *perfect dominating set* is a perfect dominating set which is also independent (respectively, connected or total). A *dominating clique* (respectively, *cycle*) is a dominating set which is also a clique (respectively, cycle). For a fixed positive integer k, a *k-dominating set* of G is a subset D of V such that for every vertex v in V there exists some vertex u in D with $d(u, v) \leq k$. An *edge dominating set* of $G = (V, E)$ is a subset F of E such that every edge in $E \setminus F$ is adjacent to some edge in F. For the above variations of domination, the corresponding *independent, connected, total, perfect, independent perfect, connected perfect, total perfect, clique, cycle, k-* and *edge domination numbers* are denoted by $\gamma_{\rm i}(G)$, $\gamma_{\rm c}(G)$, $\gamma_{\rm t}(G)$, $\gamma_{\rm per}(G)$, $\gamma_{\rm iper}(G)$, $\gamma_{\rm cper}(G)$, $\gamma_{\rm tper}(G)$, $\gamma_{\rm cl}(G)$, $\gamma_{\rm cy}(G)$, $\gamma_k(G)$ and $\gamma_{\rm e}(G)$, respectively.

A dominating set D of a graph $G = (V, E)$ corresponds to a *dominating function* which is a function $f : V \to \{0, 1\}$ such that $\sum_{u \in N[v]} f(u) \geq 1$ for any $v \in V$. The *weight* of f is $w(f) = \sum_{v \in V} f(v)$. Then $\gamma(G)$ is equal to the minimum weight of a dominating function of G. Several variations of domination are defined in terms of functions as follows. A *signed dominating function* is a function $f : V \to \{+1, -1\}$ such that $\sum_{u \in N[v]} f(u) \geq 1$. The *signed domination number* $\gamma_{\rm s}(G)$ of G is the minimum weight of a signed dominating function.

A *Roman dominating function* is a function $f : V \to \{0, 1, 2\}$ such that any vertex v with $f(v) = 0$ is adjacent to some vertex u with $f(u) = 2$. The *Roman domination number* $\gamma_{\rm Rom}(G)$ of G is the minimum weight of a Roman dominating function.

A set-valued variation of domination is as follows. For a fixed positive integer k, a *k-rainbow dominating function* is a function $f : V \to 2^{\{1,2,\ldots,k\}}$ such that $f(v) = \emptyset$ implies $\cup_{u \in N(v)} f(u) = \{1, 2, \ldots, k\}$. The *weight* of f is $w(f) = \sum_{v \in V} |f(v)|$. The *$k$-rainbow domination number* $\gamma_{k\rm rain}(G)$ of G is the minimum weight of a k-rainbow dominating function. Notice that 1-rainbow domination is the same as the ordinary domination.

A quite different variation motivated from the power system monitoring is as follows. For a positive integer k, suppose D is a vertex subset of a graph $G = (V, E)$. The following two observation rules are applied iteratively.

- **Observation Rule 1 (OR1)** A vertex in D observes itself and all of its neighbors.
- **Observation Rule 2 (OR2)** If an observed vertex is adjacent to at most k unobserved vertices, then these vertices become observed as well.

The set D is a *k-power dominating set* of G if all vertices of the graph are observed after repeatedly applying the above two observation rules. Alternatively, let $\mathcal{S}_0(D) = N[D]$ and

$$\mathcal{S}_{i+1}(D) = \cup\{N[v] : v \in \mathcal{S}_i(D), |N(v) \backslash \mathcal{S}_i(D)| \leq k\}$$

for $i \geq 0$. Notice that $\mathcal{S}_0(D) \subseteq \mathcal{S}_1(D) \subseteq \mathcal{S}_2(D) \subseteq \ldots$ and there is a t such that $\mathcal{S}_i(D) = \mathcal{S}_t(D)$ for $i \geq t$. Using this notation, D is a k-power dominating set if and only if $\mathcal{S}_t(D) = V$. The *k-power domination number* $\gamma_{k\rm pow}(G)$ of G is the minimum size of a k-power dominating set. For the case of $k = 1$, 1-power domination is called power domination.

Figure 4 shows a graph G of 13 vertices and 19 edges, whose values of $\gamma_\pi(G)$ for variations of domination and the corresponding optimal sets/functions are given below.

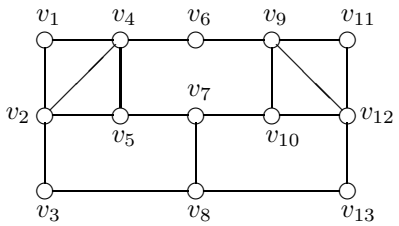


Figure 4: A graph G of 13 vertices and 19 edges.

$\gamma(G) = 3$, $D^* = \{v_2, v_8, v_9\}$;
$\gamma_{\rm i}(G) = 3$, $D^* = \{v_2, v_8, v_9\}$;
$\gamma_{\rm c}(G) = 6$, $D^* = \{v_2, v_4, v_5, v_7, v_{10}, v_{12}\}$;
$\gamma_{\rm t}(G) = 5$, $D^* = \{v_2, v_4, v_7, v_{10}, v_{12}\}$;
$\gamma_{\rm per}(G) = 4$, $D^* = \{v_2, v_3, v_8, v_9\}$;
$\gamma_{\rm iper}(G) = \infty$, infeasible;
$\gamma_{\rm cper}(G) = 10$, $D^* = \{v_1, v_2, v_4, v_5, v_6, v_7, v_9, v_{10}, v_{11}, v_{12}\}$;
$\gamma_{\rm tper}(G) = 6$, $D^* = \{v_1, v_2, v_4, v_5, v_{12}, v_{13}\}$;
$\gamma_{\rm cl}(G) = \infty$, infeasible;
$\gamma_{\rm cy}(G) = 8$, $D^* = \{v_2, v_4, v_6, v_9, v_{12}, v_{10}, v_7, v_5\}$;
$\gamma_2(G) = 2$, $D^* = \{v_6, v_7\}$;
$\gamma_k(G) = 1$, $D^* = \{v_7\}$ for $k \geq 3$;
$\gamma_{\rm e}(G) = 3$, $D^* = \{v_2v_4, v_9v_{12}, v_7v_8\}$;
$\gamma_{\rm s}(G) = 5$, $f^*(v_i) = -1$ for $i = 1, 6, 8, 11$ and $f^*(v_i) = +1$ for other i;
$\gamma_{\rm Rom}(G) = 6$, $f^*(v_i) = 2$ for $i = 2, 8, 9$ and $f^*(v_i) = 0$ for other i;
$\gamma_{1\rm rain}(G) = 3$, $f^*(v_i) = \{1\}$ for $i = 2, 8, 9$ and $f^*(v_i) = \emptyset$ for other i;
$\gamma_{2\rm rain}(G) = 6$, $f^*(v_i) = \{1, 2\}$ for $i = 2, 8, 9$ and $f^*(v_i) = \emptyset$ for other i;
$\gamma_{3\rm rain}(G) = 8$, $f^*(v_i) = \{1\}$ for $i = 3, 6, 8, 13$, $f^*(v_1) = f^*(v_{10}) = \{2\}$,
$f^*(v_5) = f^*(v_{11}) = \{3\}$ and $f^*(v_i) = \emptyset$ for other i;
$\gamma_{4\rm rain}(G) = 10$, $f^*(v_i) = \{1\}$ for $i = 3, 6, 8, 13$, $f^*(v_1) = \{2\}, f^*(v_{10}) = \{2, 4\}$,
$f^*(v_5) = \{3, 4\}, f^*(v_{11}) = \{3\}$ and $f^*(v_i) = \emptyset$ for other i;
$\gamma_{5\rm rain}(G) = 12$, $f^*(v_i) = \{1\}$ for $i = 3, 6, 8, 13$, $f^*(v_1) = \{2\}, f^*(v_{10}) = \{2, 4, 5\}$,
$f^*(v_5) = \{3, 4, 5\}, f^*(v_{11}) = \{3\}$ and $f^*(v_i) = \emptyset$ for other i;
$\gamma_{k\rm rain}(G) = 13$, $f^*(v_i) = \{1\}$ for all i for $k \geq 6$;
$\gamma_{k\rm pow}(G) = 1$, $D^* = \{v_2\}$ for $k \geq 1$.

The (*vertex-*)*weighted* versions of all of the above vertex-subset variations of domination can also be considered. Now, every vertex v has a weight $w(v)$ of real number. The problem is to find a dominating set D in a suitable variation such that

$$w(D) = \sum\nolimits_{v \in D} w(v)$$

is as small as possible. Denote this minimum value by $\gamma_\pi(G, w)$, where π stands for a variation of the domination problem. When $w(v) = 1$ for all vertices v, the weighted cases become the cardinality cases.

For some variations of domination, the vertex weights may assume to be non-negative as the following lemma shows.

Lemma 2.1 *Suppose $G = (V, E)$ is a graph in which every vertex is associated with a weight $w(v)$ of real number. If $w'(v) = \max\{w(v), 0\}$ for all vertices $v \in V$, then for any $\pi \in \{\emptyset, {\rm c}, {\rm t}, k, k{\rm pow}\}$ we have*

$$\gamma_\pi(G, w) = \gamma_\pi(G, w') + \sum\nolimits_{w(v) < 0} w(v).$$

Proof. Denote by A the set of all vertices v with $w(v) < 0$. Suppose D is a π-dominating set of G with $\sum_{v\in D} w(v) = \gamma_\pi(G, w)$. Then

$$\begin{aligned}\gamma_\pi(G, w') &\leq \sum\nolimits_{v\in D} w'(v)\\ &= \sum\nolimits_{v\in D} w(v) - \sum\nolimits_{v\in D\cap A} w(v)\\ &\leq \gamma_\pi(G, w) - \sum\nolimits_{w(v)<0} w(v).\end{aligned}$$

On the other hand, for any π-dominating set D of G with $\sum_{v\in D} w'(v) = \gamma_\pi(G, w')$, $D \cup A$ is also a π-dominating set of G and so

$$\begin{aligned}\gamma_\pi(G, w) &\leq \sum\nolimits_{v\in D\cup A} w(v)\\ &= \sum\nolimits_{v\in D} w'(v) + \sum\nolimits_{v\in A} w(v)\\ &= \gamma_\pi(G, w') + \sum\nolimits_{w(v)<0} w(v).\end{aligned}$$

The lemma then follows. ∎

More generally, one may consider *vertex-edge-weighted* cases of the domination problems as follows. Now, besides the weights of vertices, each edge e has a weight $\overline{w}(e)$. The object is then to find a dominating set D in a suitable variation such that

$$\overline{w}(D) = \sum\nolimits_{v\in D} w(v) + \sum\nolimits_{u\in V\setminus D} \overline{w}(uu')$$

is as small as possible, where u' is a vertex in D that is adjacent to u. Note that there are many choices of u' except for the perfect domination and its three variations. Denote this minimum value by $\gamma_\pi(G, w, \overline{w})$, where π stands for a variation of domination. When $\overline{w}(e) = 0$ for all edges e, the vertex-edge-weighted cases become the vertex-weighted cases.

Another parameter related to domination is as follows. The *domatic number* $d(G)$ of a graph G is the maximum number r such that G has r pairwise disjoint dominating sets $D_1, D_2, \ldots, D_r$. One can also define *independent, connected, total, perfect, independent perfect, connected perfect, total perfect, clique, cycle, k-, edge, k-power domatic numbers* $d_{\rm i}(G), d_{\rm c}(G), d_{\rm t}(G), d_{\rm per}(G), d_{\rm iper}(G), d_{\rm cper}(G)$, $d_{\rm tper}(G), d_{\rm cl}(G), d_{\rm cy}(G), d_k(G), d_{\rm e}(G), d_{k\rm pow}(G)$, respectively, according to above variations of domination in similar ways.

2.3 Special classes of graphs

In this subsection, special classes of graphs are introduced. They are not only important in the study of domination, but also fundamental in graph theory.

A *complete graph* is a graph whose vertex set is a clique. The complete graph with n vertices is denoted by K_n. The complement $\overline{K}_n$ of the complete graph K_n is then a graph with no edge.

The *n-path*, denoted by P_n, is a graph with n vertices that contains a chordless path of length n. The *n-cycle*, denoted by C_n, is a graph with n vertices that contains a chordless cycle of length n.

An *r-partite graph* is a graph whose vertex set can be partitioned into r stable sets, which are called its *partite sets*. A 2-partite graph is usually called a *bipartite* graph. A *complete r-partite graph* is a r-partite graph in which vertices in different partite sets are adjacent. A complete r-partite graph with partite sets having $n_1, n_2, \ldots, n_r$ vertices, respectively, is denoted by $K_{n_1,n_2,\ldots,n_r}$.

A *tree* is a connected graph without any cycle. A *directed tree* is an orientation of a tree. A *rooted tree* is a directed tree in which there is a special vertex r, called the *root*, such that for every vertex v there is a directed r-v path. Trees are probably the simplest structures in graph theory. Problems looking hard in general graphs are often investigated in trees first as a warm up. Domination in trees is introduced in Section 3. Many ideas for domination in trees are then generalized to other classes of graphs.

Suppose $\mathcal{F}$ is a family of sets. The *intersection graph* of $\mathcal{F}$ is the graph obtained by representing each set in $\mathcal{F}$ as a vertex and joining two distinct vertices with an edge if their corresponding sets intersect. It is well-known that any graph is the intersection graph of some family $\mathcal{F}$. The problem of characterizing the intersection graphs of families of sets having some specific topological or other pattern is often interesting and frequently has applications in the real world. A typical example is the class of interval graphs. An *interval graph* is the intersection graph of intervals in the real line. They play important roles in many applications. Domination in interval graphs is investigated in Section 4.

A graph is *chordal* (or *triangulated*) if every cycle of length greater than three has a chord. The class of chordal graphs is one of the classical classes in the *perfect graph theory*, see the book by Golumbic [103]. It turns out to be also very important in the domination theory. It is well-known that a graph is chordal if and only if it is the intersection graph of some subtrees of a certain tree. If these subtrees are paths, this chordal graph is called an *undirected path graph*. If these subtrees are directed paths of a rooted tree, the graph is called a *directed path graph*. If these subtrees are paths in some n-path, the graph is just an interval graph.

Most variations of the domination problem are NP-complete even for chordal graphs, see Section 5. As an important subclass of chordal graphs, the class of strongly chordal graphs is a star of the domination theory. Strongly chordal graphs include directed path graphs, which in turn include trees and interval graphs. Domination in strongly chordal graphs is studied in Section 6.

A permutation diagram consists of n points on each of two parallel lines and n straight line segments matching the points. The intersection graph of the line

segments is called a *permutation graph*. Domination in permutation graphs is introduced in Section 7.

A *comparability graph* is a graph $G = (V, E)$ that has a *transitive* orientation $G' = (V, E')$, i.e., $uv \in E'$ and $vw \in E'$ imply $uw \in E'$. A *cocomparability graph* is the complement of a comparability graph. Cocomparabity graphs are generalizations of permutation graphs and intervals graphs. Domination in cocomparability graphs is investigated in Section 8.

A graph is *distance-hereditary* if the distance between any two vertices is the same in any connected induced subgraph containing them. Domination in distance-hereditary graphs is studied in Section 9.

For more detailed discussions of these classes of graphs, see the remaining sections of this chapter.

3 Trees

3.1 Basic properties of trees

Recall that a tree is an acyclic connected graph. The following characterizations are well-known, see the textbook by West [194].

Theorem 3.1 *The following statements are equivalent for any graph $G = (V, E)$.*

(1) *G is a tree.*

(2) *G is connected and $|V| = |E| + 1$.*

(3) *G is acyclic and $|V| = |E| + 1$.*

(4) *For any two vertices u and v, there is a unique u-v path.*

(5) *The vertices of G have an ordering $[v_1, v_2, \ldots, v_n]$ such that v_i is a leaf of $G_i = G[\{v_i, v_{i+1}, \ldots, v_n\}]$ for $1 \leq i \leq n-1$, or equivalently*

$$\text{for each } 1 \leq i \leq n-1,\ v_i \text{ is adjacent to } \textit{exactly one } v_j \text{ with } j > i. \qquad \text{(TO)}$$

The ordering in Theorem 3.1 (5) is called a *tree ordering* of the tree, where the only neighbor v_j of v_i with $j > i$ is called the *parent* of v_i and v_i is a *child* of v_j. The tree ordering plays an important role in many algorithms dealing with trees. Many algorithms on trees process from leaves by passing information to their parents iteratively, or equivalently, doing a loop according to a tree ordering. From an algorithmic point of view, the testing of a tree and finding a tree ordering can be done in linear time. Figure 5 shows a tree of 11 vertices and a tree ordering.

For some algorithms in trees, it is necessary to process simultaneously a group of leaves which is adjacent to a vertex with only one non-leaf neighbor. This corresponds to a tree ordering, which is called a *strong tree ordering*, with the property that all children of a vertex are consecutive in the ordering. The tree order $[v_2, v_3, v_1, v_4, v_6, v_7, v_5, v_8, v_9, v_{11}, v_{12}, v_{10}, v_{13}, v_{14}, v_{15}]$ is strong for the tree in Figure 5.

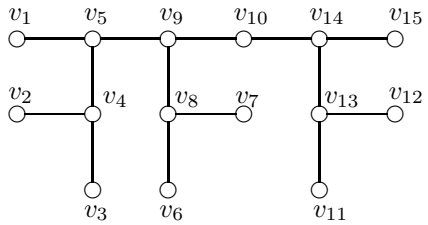


Figure 5: An example of the tree ordering.

3.2 Labeling algorithm for trees

Cockayne, Goodman and Hedetniemi [58] gave the first linear-time algorithm for the domination problem in trees by a labeling method, which is a naive but useful approach.

The algorithm starts processing a leaf v of a tree T, which is adjacent to a unique vertex u. To dominate v, a minimum dominating set D of T must contain u or v. However, since $N[v] \subseteq N[u]$, it is better to have u in D rather than v in D. So one can keep an information "required" in u and delete v from T. At some iteration, if a "required" leaf v adjacent to u which is not labeled by "required" is processed, it is necessary to put v into D and delete it from T. But now, there is a vertex in D that dominates u, so u is labeled by "free". For convenience, all vertices are labeled by "bound" initially.

More precisely, suppose the vertex set of a graph $G = (V, E)$ is partitioned into three sets F, B and R, where F consists of *free* vertices, B consists of *bound* vertices and R consists of *required* vertices. A *mixed dominating set* of G (*with respect to* F, B, R) is a subset $D \subseteq V$ such that

$$R \subseteq D \text{ and every vertex in } B \setminus D \text{ is adjacent to some vertex in } D.$$

Free vertices need not be dominated by D but may be included in D in order to dominate bound vertices. The *mixed domination number* $\gamma^{\mathrm{m}}(G)$ is the minimum size of a mixed dominating set in G, such a set is called an *md-set* of G. Note that mixed domination is the ordinary domination when $B = V$ and $F = R = \emptyset$.

The construction and correctness of the algorithm is based on the following theorem.

Theorem 3.2 *Suppose T is a tree having free, bound and required vertices F, B and R, respectively. Let v be a leaf of T, which is adjacent to u. Then the following statements hold.*

(1) *If $v \in F$, then $\gamma^{\mathrm{m}}(T) = \gamma^{\mathrm{m}}(T - v)$.*

(2) *If $v \in B$ and T' is the tree which results from T by deleting v and relabeling u as "required", then $\gamma^{\mathrm{m}}(T) = \gamma^{\mathrm{m}}(T')$.*

(3) *If $v \in R$ and $u \in R$, then $\gamma^{\mathrm{m}}(T) = \gamma^{\mathrm{m}}(T - v) + 1$.*

(4) *If $v \in R$, $u \notin R$ and T' is the tree which results from T by deleting v and relabeling u as "free", then $\gamma^{\mathrm{m}}(T) = \gamma^{\mathrm{m}}(T') + 1$.*

Proof. (1) Since v is free, any md-set D' of $T - v$ is also a mixed dominating set of T. Thus, $\gamma^{\mathrm{m}}(T) \leq |D'| = \gamma^{\mathrm{m}}(T - v)$. On the other hand, suppose D is an md-set of T. If $v \notin D$, then D is also a mixed dominating set of $T - v$. If $v \in D$, then $(D \setminus \{v\}) \cup \{u\}$ is a mixed dominating set of $T - v$, whose size is at most $|D|$. Thus, in either case, $\gamma^{\mathrm{m}}(T - v) \leq |D| = \gamma^{\mathrm{m}}(T)$.

(2) Since u is required in T', any md-set D' of T' always contains u and hence is also a mixed dominating set of T. Thus, $\gamma^{\mathrm{m}}(T) \leq |D'| = \gamma^{\mathrm{m}}(T')$. On the other hand, suppose D is an md-set of T. Since v is bound in T, either u or v is in D. In any case, $D' = (D \setminus \{v\}) \cup \{u\}$ is a mixed dominating set of T', in which u is considered as a required vertex. So, $\gamma^{\mathrm{m}}(T') \leq |D'| \leq |D| = \gamma^{\mathrm{m}}(T)$.

(3) If D' is an md-set of T', then $D' \cup \{v\}$ is a mixed dominating set of T. Thus, $\gamma^{\mathrm{m}}(T) \leq |D' \cup \{v\}| = \gamma^{\mathrm{m}}(T') + 1$. On the other hand, any md-set D of T contains both u and v. Then $D \setminus \{v\}$ is a mixed dominating set of T'. So, $\gamma^{\mathrm{m}}(T') \leq |D \setminus \{v\}| = \gamma^{\mathrm{m}}(T) - 1$.

(4) If D' is an md-set of T', then $D' \cup \{v\}$ is a mixed dominating set of T. Thus, $\gamma(T) \leq |D' \cup \{v\}| = \gamma^{\mathrm{m}}(T') + 1$. On the other hand, any md-set D of T contains v. Since u is free in T', $D \setminus \{v\}$ is a mixed dominating set in T'. So, $\gamma^{\mathrm{m}}(T') \leq |D \setminus \{v\}| = \gamma^{\mathrm{m}}(T) - 1$. ∎

The above theorem then gives the following algorithm for the mixed domination problem in trees.

Algorithm DomTreeL. Find a minimum mixed dominating set of a tree.
Input. A tree T whose vertices are labeled by free, bound or required. A tree ordering $[v_1, v_2, \ldots, v_n]$ of T.
Output. A minimum mixed dominating set D of T.
Method.

$D \leftarrow \phi$;
for $i = 1$ **to** $n - 1$ **do**
    let v_j be the parent of v_i;
    if (v_i is bound) **then**
        relabel v_j as required;
    else if (v_i is required) **then**
        $D \leftarrow D \cup \{v_i\}$;
        if v_j is bound **then** relabel v_j as free;
        end if;
    end for;
if v_n is not free **then** $D \leftarrow D \cup \{v_n\}$.

Slater [184] generalized the above idea to solve the k-domination problem in trees. In fact he solved a slightly more general problem called R-domination.

Now, each vertex v is associated with an ordered pair $R_v = (a_v, b_v)$, where a_v is a nonnegative integer and b_v a positive integer. The dominating set D is chosen so that each vertex v is within distance a_v from some vertex in D. The integer b_v indicates that there is a vertex in the current D that is at distance b_v from v. More precisely, an R-dominating set of $G = (V, E)$ is a vertex subset D such that for any vertex v in G either there is some $u \in D$ with $d(u, v) \leq a_v$ or else there is some $u \in V$ with $b_u + d(u, v) \leq a_v$. The *R-domination number* $\gamma^R(G)$ of G is the minimum size of an R-dominating set. Notice that R-domination with each $R_v = (k, k + 1)$ is the same as the k-domination.

The construction and correctness of Slater's algorithm is based on the following theorem whose proof is omitted here.

Theorem 3.3 *Suppose T is a tree in which each vertex v has a label $R_v = (a_v, b_v)$, where a_v is a nonnegative integer and b_v a positive integer. Let v be a leaf of T, which is adjacent to u. Then the following statements hold.*

(1) *If $a_v \geq b_v$ and T' is the tree which results from T by deleting v and resetting b_u by $\min\{b_u, b_v + 1\}$, then $\gamma^R(T) = \gamma^R(T')$.*

(2) *If $a_v = 0$ and T' is the tree which results from T by deleting v and resetting b_u by 1, then $\gamma^R(T) = \gamma^R(T') + 1$.*

(3) *If $0 < a_v < b_v$ and T' is the tree which results from T by deleting v and resetting a_u by $\max\{a_u, a_v - 1\}$ and b_u by $\min\{b_u, b_v + 1\}$, then $\gamma^R(T) = \gamma^R(T')$.*

This then gives the following algorithm for R-domination in trees.

Algorithm RDomTreeL. Find a minimum R-dominating set of a tree.
Input. A tree T with a tree ordering $[v_1, v_2, \ldots, v_n]$, in which each vertex v has a label $R_v = (a_v, b_v)$, where $a_v \geq 0$ and $b_v > 0$ are integers.
Output. A minimum R-dominating set D of T.
Method.

$D \leftarrow \phi$;
for $i = 1$ **to** $n - 1$ **do**
    let v_j be the parent of v_i;
    if $(a_{v_i} \geq b_{v_i})$ **then**
        $b_{v_j} \leftarrow \min\{b_{v_j}, b_{v_i} + 1\}$;
    else if $(a_{v_i} = 0)$ **then**
        $D \leftarrow D \cup \{v_i\}$;
        $b_{v_j} \leftarrow 1$;
        end if;
    else if $(0 < a_{v_i} < b_{v_i})$ **then**
        $a_{v_j} \leftarrow \max\{a_{v_j}, a_{v_i} - 1\}$;
        $b_{v_j} \leftarrow \min\{b_{v_j}, b_{v_i} + 1\}$;
        end if;
    end for;
if $(a_{v_n} < b_{v_n})$ **then** $D \leftarrow D \cup \{v_n\}$.

The labeling algorithm is also used for many variations of domination. For instance, Mitchell and Hedetniemi [163] and Yannakakis and Gavril [198] gave labeling algorithms for the edge domination problem in trees. Laskar, Pfaff, Hedetniemi and Hedetniemi [150] gave a labeling algorithm for the total domination problem in trees.

Next, an example of labeling algorithm using strong tree orderings is demonstrated. Chang, Wu and Zhu [49] investigated the k-rainbow domination problem on trees. For technical reasons, they in fact dealed with a more general problem. A *k-rainbow assignment* is a mapping L that assigns each vertex v a label $L(v) = (a_v, b_v)$ with $a_v, b_v \in \{0, 1, \ldots, k\}$. A *$k$-$L$-rainbow dominating function* is a function $f : V(G) \to 2^{\{1,2,\ldots,k\}}$ such that for every vertex v in G the following conditions hold.

(L1) $|f(v)| \geq a_v$.

(L2) $|\cup_{u \in N(v)} f(u)| \geq b_v$ whenever $f(v) = \emptyset$.

The *k-L-rainbow domination number* $\gamma_{kL\mathrm{rain}}(G)$ of G is the minimum weight of a k-L-rainbow dominating function. A k-L-rainbow dominating function f of G is *optimal* if $w(f) = \gamma_{kL\mathrm{rain}}(G)$. Notice that k-rainbow domination is the same as k-L-rainbow domination if $L(v) = (0, k)$ for each $v \in V(G)$.

Theorem 3.4 *Suppose v is a leaf adjacent to u in a graph G with a k-rainbow assignment L. Let $G' = G - v$ and L' be the restriction of L on $V(G')$, except that when $a_v > 0$ we let $b'_u = \max\{0, b_u - a_v\}$. Then the following hold.*

(1) *If $a_v > 0$, then $\gamma_{kL\mathrm{rain}}(G) = \gamma_{kL'\mathrm{rain}}(G') + a_v$.*

(2) *If $a_v = 0$ and $a_u \geq b_v$, then $\gamma_{kL\mathrm{rain}}(G) = \gamma_{kL'\mathrm{rain}}(G')$.*

Theorem 3.5 *Suppose $N(u) = \{z, v_1, v_2, \ldots, v_s\}$ such that $v_1, v_2, \ldots, v_s$ are leaves in a graph G with a k-rainbow assignment L. Assume $a_{v_i} = 0$ for $1 \leq i \leq s$ and $b_{v_1} \geq b_{v_2} \geq \ldots \geq b_{v_s} > a_u$. Let $b^* = \min\{b_{v_i} + i - 1 : 1 \leq i \leq s + 1\} = b_{v_{i^*}} + i^* - 1$, where $b_{v_{s+1}} = a_u$ and i^* is chosen as small as possible. If $G' = G - \{v_1, v_2, \ldots, v_s\}$ and L' is the restriction of L on $V(G')$ with modifications that $a'_u = b_{v_{i^*}}$ and $b'_u = \max\{0, b_u - i^* + 1\}$, then $\gamma_{kL\mathrm{rain}}(G) = \gamma_{kL'\mathrm{rain}}(G') + i^* - 1$.*

Remark that for the case when the component of G containing u is a star, the vertex z does not exist. There is in fact no vertex v_{s+1}. The assignment of $b_{v_{s+1}} = a_u$ is for the purpose of convenience. In the case of $i^* = s + 1$, it just means that a'_u is the same as a_u.

The theorems above then give the following linear-time algorithm for the k-L-rainbow domination problem in trees.

Algorithm RainbowDomTreeL. Find the k-L-domination number of a tree.
Input. A tree $T = (V, E)$ in which each vertex v is labeled by $L(v) = (a_v, b_v)$.
Output. The minimum k-L-rainbow dominating number r of T.
Method.

$r \leftarrow 0$;
get a Breadth First Search ordering $x_1, x_2, \ldots, x_n$ for the tree T rooted at x_1;
for $j = 1$ **to** n **do** $s_j \leftarrow 0$; {number of children x_i with $a_{x_i} = 0$ and $b_{x_i} > a_{x_j}$}
for $j = n$ **to** 2 **step by** -1 **do**
    $s \leftarrow s_j$;
    $v \leftarrow x_j$;
    if $s > 0$ **then** {apply Theorem 3.5}
        let $u = v$ and $z, v_1, v_2, \ldots, v_s, b^*, i^*$ be as described in Theorem 3.5;
        $r \leftarrow r + i^* - 1$;
        $a_u \leftarrow b_{v_{i^*}}$;
        $b_u \leftarrow \max\{0, b_u - i^* + 1\}$;
        end if;
    else {apply Theorem 3.4}
        let $u = x_{j'}$ be the parent of v;
        if $a_v > 0$ **then** { $b_u \leftarrow \max\{0, b_u - a_v\}$; $r \leftarrow r + a_v$ };
        else if $a_u < b_v$ **then** $s_{j'} \leftarrow s_{j'} + 1$;
        end else;
    end do;
if $a_{x_1} > 0$ **then** $r \leftarrow r + a_{x_1}$; **else if** $b_{x_1} > 0$ **then** $r \leftarrow r + 1$.

As these algorithms suggest, the labeling algorithm may only work for problems whose solutions have "local property". For an example of problem without local property, the independent domination problem in the tree T of Figure 6 is considered. The only minimum independent dominating set of T is $\{v_1, v_3\}$. If the tree ordering $[v_1, v_2, v_4, v_3, v_5]$ is given, the algorithm must be clever enough to put the leaf v_1 into the solution at the first iteration. If another tree ordering $[v_5, v_4, v_3, v_2, v_1]$ is given, the algorithm must be clever enough not to put the leaf v_5 at the first iteration. So, the algorithm must be one that not only looks at a leaf and its only neighbor, but also has some idea about the whole structure of the tree. This is the meaning that the solution does not have a "local property".

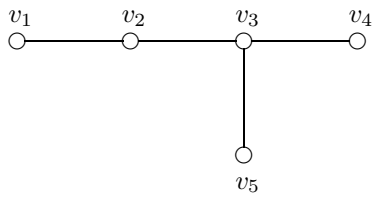


Figure 6: A tree T with a unique minimum independent dominating set.

3.3 Dynamic programming for trees

Dynamic programming is a powerful method for solving many discrete optimization problems; see the books by Bellman and Dreyfus [15], Dreyfus and Law [87] and Nemhauser [168]. The main idea of the dynamic programming approach for domination is to turn the "bottom-up" labeling method into "top-down". Now a specific vertex u is chosen from G. A minimum dominating set D of G either contains or does not contain u. So it is useful to consider the following two domination problems which are the ordinary domination problem with boundary conditions.

$$\gamma^1(G,u) = \min\{|D| : D \text{ is a dominating set of } G \text{ and } u \in D\}.$$

$$\gamma^0(G,u) = \min\{|D| : D \text{ is a dominating set of } G \text{ and } u \notin D\}.$$

Lemma 3.6 $\gamma(G) = \min\{\gamma^1(G,u),\ \gamma^0(G,u)\}$ *for any graph* G *with a specific vertex* u.

Suppose H is another graph with a specific vertex v. Let I be the graph with the specific vertex u, which is obtained from the disjoint union of G and H by joining a new edge uv; see Figure 7.

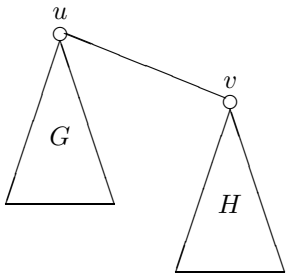


Figure 7: The composition of two trees G and H.

The aim is to use $\gamma^1(G,u), \gamma^0(G,u), \gamma^1(H,v)$ and $\gamma^0(H,v)$ to find $\gamma^1(I,u)$ and $\gamma^0(I,u)$. Suppose D is a dominating set of I with $u \in I$. Then $D = D' \cup D''$, where D' is a dominating set of G with $v \in D'$ and D'' is a subset of $V(H)$ which dominates $V(H) - \{v\}$. There are two cases. In the case of $v \in D''$, D'' is a dominating set of H. On the other hand, if $v \notin D''$ then D'' is a dominating set of $H - v$. In order to cover the latter case, the following new problem is introduced.

$$\gamma^{00}(G,u) = \min\{|D| : D \text{ is a dominating set of } G - u\}.$$

Note that $\gamma^{00}(G,u) \leq \gamma^0(G,u)$, since a dominating set D of G with $u \notin D$ is also a dominating set of $G - u$.

Theorem 3.7 *Suppose G and H are graphs with specific vertices u and v, respectively. Let I be the graph with the specific vertex u, which is obtained from the disjoint union of G and H by joining a new edge uv. Then the following statements hold.*

(1) $\gamma^1(I,u) = \gamma^1(G,u) + \min\{\gamma^1(H,v), \gamma^{00}(H,v)\}$.
(2) $\gamma^0(I,u) = \min\{\gamma^0(G,u) + \gamma^0(H,v), \gamma^{00}(G,u) + \gamma^1(H,v)\}$.
(3) $\gamma^{00}(I,u) = \gamma^{00}(G,u) + \gamma(H) = \gamma^{00}(G,u) + \min\{\gamma^1(H,v), \gamma^0(H,v)\}$.

Proof. (1) This follows from the fact that D is a dominating set of I with $u \in D$ if and only if $D = D' \cup D''$, where D' is a dominating set of G with $u \in D'$ and D'' is a dominating set of H with $v \in D''$ or a dominating set of $H - v$.

(2) This follows from the fact that D is a dominating set of I with $u \notin D$ if and only if $D = D' \cup D''$, where either D' is a dominating set of G with $u \notin D'$ and D'' is a dominating set of H with $v \notin D''$, or D' is a dominating set of $G - u$ and D'' is a dominating set of H with $v \in D''$.

(3) This follows from the fact that D is a dominating set of $I - u$ if and only if $D = D' \cup D''$, where D' is a dominating set of $G - u$ and D'' is a dominating set of H. ∎

The lemma and the theorem above then give the following dynamic programming algorithm for the domination problem in trees.

Algorithm DomTreeD. Determine the domination number of a tree.
Input. A tree T with a tree ordering $[v_1, v_2, \ldots, v_n]$.
Output. The domination number $\gamma(T)$ of T.
Method.

for $i = 1$ **to** n **do**
    $\gamma^1(v_i) \leftarrow 1$;
    $\gamma^0(v_i) \leftarrow \infty$;
    $\gamma^{00}(v) \leftarrow 0$;
    end do;
for $i = 1$ **to** $n - 1$ **do**
    let v_j be the parent of v_i;
    $\gamma^1(v_j) \leftarrow \gamma^1(v_j) + \min\{\gamma^1(v_i), \gamma^{00}(v_i)\}$;
    $\gamma^0(v_j) \leftarrow \min\{\gamma^0(v_j) + \gamma^0(v_i), \gamma^{00}(v_j) + \gamma^1(v_i)\}$;
    $\gamma^{00}(v_j) \leftarrow \gamma^{00}(v_j) + \min\{\gamma^1(v_i), \gamma^0(v_i)\}$;
    end do;
$\gamma(T) \leftarrow \min\{\gamma^1(v_n), \gamma^0(v_n)\}$.

The advantage of the dynamic programming method is that it also works for problems whose solutions have no local property. As an example, Beyer, Proskurowski, Hedetniemi and Mitchell [22] solved the independent domination problem by this method. Moreover, the method can be used to solve

the vertex-edge-weighted cases. The following derivation for the vertex-edge-weighted domination in trees is slightly different from that given by Natarajan and White [167].

Define $\gamma^1(G,u,w,\overline{w}), \gamma^0(G,u,w,\overline{w})$ and $\gamma^{00}(G,u,w,\overline{w})$ in the same way as $\gamma^1(G,u)$, $\gamma^0(G,u)$ and $\gamma^{00}(G,u)$, except that $|D|$ is replaced by $\overline{w}(D)$.

Lemma 3.8 *$\gamma(G,w,\overline{w}) = \min\{\gamma^1(G,u,w,\overline{w}), \gamma^0(G,u,w,\overline{w})\}$ for any graph G with a specific vertex u.*

Theorem 3.9 *Suppose G and H are graphs with specific vertices u and v, respectively. Let I be the graph with the specific vertex u, which is obtained from the disjoint union of G and H by joining a new edge uv. The following statements hold.*

(1) $\gamma^1(I,u,w,\overline{w}) = \gamma^1(G,u,w,\overline{w}) + \min\{\gamma(H,v,w,\overline{w}), \overline{w}(uv) + \gamma^{00}(H,v,w,\overline{w})\}$.

(2) $\gamma^0(I,u,w,\overline{w}) = \min\{\gamma^0(G,u,w,\overline{w}) + \gamma(H,w,\overline{w}), \gamma^{00}(G,u,w,\overline{w}) + \overline{w}(uv) + \gamma^1(H,v,w,\overline{w})\}$.

(3) $\gamma^{00}(I,u,w,\overline{w}) = \gamma^{00}(G,u,w,\overline{w}) + \gamma(H,w,\overline{w})$.

The lemma and the theorem above then give the following algorithm for the vertex-edge-weighted domination problem in trees.

Algorithm VEWDomTreeD. Determine the vertex-edge-weighted domination number of a tree.
Input. A tree T with a tree ordering $[v_1, v_2, \ldots, v_n]$, and each vertex v has a weight $w(v)$ and each edge e has a weight $\overline{w}(e)$.
Output. The vertex-edge-weighted domination number $\gamma(T,w,\overline{w})$ of T.
Method.

for $i = 1$ **to** n **do**
    $\gamma(v_i) \leftarrow \gamma^1(v_i) \leftarrow w(v_i)$;
    $\gamma^0(v_i) \leftarrow \infty$;
    $\gamma^{00}(v_i) \leftarrow 0$;
    end do;
for $i = 1$ **to** $n-1$ **do**
    let v_j be the parent of v_i;
    $\gamma^1(v_j) \leftarrow \gamma^1(v_j) + \min\{\gamma(v_i), \overline{w}(uv) + \gamma^{00}(v_i)\}$;
    $\gamma^0(v_j) \leftarrow \min\{\gamma^0(v_j) + \gamma(v_i), \gamma^{00}(v_j) + \overline{w}(uv) + \gamma^1(v_i)\}$;
    $\gamma^{00}(v_j) \leftarrow \gamma^{00}(v_j) + \gamma(v_i)$;
    $\gamma(v_j) \leftarrow \min\{\gamma^1(v_j), \gamma^0(v_j)\}$;
    end do;
$\gamma(T,w,\overline{w}) \leftarrow \gamma(v_n)$.

The dynamic programming method is also used in several papers for solving variations of the domination problems in trees, see [13, 98, 116, 124, 185, 203].

3.4 Primal-dual approach for trees

The most beautiful method used in domination may be the primal-dual approach. In this method, besides the original domination problem, the following dual problem is also considered. In a graph $G = (V, E)$, a 2-*stable set* is a subset $S \subseteq V$ in which every two distinct vertices u and v have distance $d(u, v) > 2$. The 2-*stability number* $\alpha_2(G)$ of G is the maximum size of a 2-stable set in G. It is easy to see the following inequality.

Weak Duality Inequality: $\alpha_2(G) \leq \gamma(G)$ for any graph G.

Note that the above inequality can be strict, as shown by the n-cycle C_n that $\alpha_2(C_n) = \lfloor \frac{n}{3} \rfloor$ but $\gamma(C_n) = \lceil \frac{n}{3} \rceil$.

For a tree T, an algorithm which outputs a dominating set D^* and a 2-stable set S^* with $|D^*| \leq |S^*|$ is designed. Then

$$|S^*| \leq \alpha_2(T) \leq \gamma(T) \leq |D^*| \leq |S^*|,$$

which imply that all inequalities are equalities. Consequently, D^* is a minimum dominating set, S^* is a maximum 2-stable and the *strong duality equality* $\alpha_2(T) = \gamma(T)$ holds.

The algorithm starts from a leaf v adjacent to u. It also uses the idea as in the labeling algorithm that u is more powerful than v since $N[v] \subseteq N[u]$. Instead of choosing v, u is put into D^*. Besides, v is also put into S^*. More precisely, the algorithm is as follows.

Algorithm DomTreePD. Find a minimum dominating set and a maximum 2-stable set of a tree.
Input. A tree T with a tree ordering $[v_1, v_2, \ldots, v_n]$.
Output. A minimum dominating set D^* and a maximum 2-stable set S^* of T.
Method.

$D^* \leftarrow \phi$;
$S^* \leftarrow \phi$;
for $i = 1$ **to** n **do**
    let v_j be the parent of v_i;
    (assume $v_j = v_n$ for $v_i = v_n$)
    if $(N[v_i] \cap D^* = \phi)$ **then**
        $D^* \leftarrow D^* \cup \{v_j\}$;
        $S^* \leftarrow S^* \cup \{v_i\}$;
        end if:
    end do.

To verify the algorithm, it is sufficient to prove that D^* is a dominating set, S^* is a 2-stable set and $|D^*| \leq |S^*|$.

D^* is clearly a dominating set as the if-then statement does.

Suppose S^* is not a 2-stable set, i.e., there exist v_i and $v_{i'}$ in S^* such that $i < i'$ but $d_T(v_i, v_{i'}) \leq 2$. Let $T_i = T[\{v_i, v_{i+1}, \ldots, v_n\}]$. Then T_i contains v_i,

v_j and $v_{i'}$. Since $d(v_i, v_{i'}) \leq 2$, the unique v_i-$v_{i'}$ path in T (and also in T') is either $v_i, v_{i'}$ or $v_i, v_j, v_{i'}$. In any case, $v_j \in N[v_{i'}]$. Thus, at the end of iteration i, D^* contains v_j. When the algorithm processes $v_{i'}$, $N[v_{i'}] \cap D^* \neq \phi$ which causes that S^* does not contain $v_{i'}$, a contradiction.

$|D^*| \leq |S^*|$ follows from that when v_j is added into D^*, which may or may not already be in D^*, a new vertex v_i is always added into S^*.

Theorem 3.10 *Algorithm DomTreePD gives a minimum dominating set D^* and a maximum 2-stable set S^* of a tree T with $|D^*| = |S^*|$ in linear time.*

Theorem 3.11 (Strong Duality) *$\alpha_2(T) = \gamma(T)$ for any tree T.*

The primal-dual approach was in fact used by Farber [92] and Kolen [144] for the weighted domination problem in strongly chordal graphs. It was also used by Cheston, Fricke, Hedetniemi and Jacobs [57] for upper fraction domination in trees.

3.5 Power domination in trees

Power domination is a most different variation of domination. The observation rules make it so different from the ordinary domination as well as other variations. Haynes, Hedetniemi, Hedetniemi and Henning [113] gave a linear-time algorithm for the power domination problem in trees. This subsection demonstrates a neat linear-time algorithm offered by Guo, Niedermeier and Raible [105].

Algorithm PowerDomTree. Find a minimum power dominating set of a tree.
Input. A tree T rooted at vertex r.
Output. A minimum power dominating set D of T.
Method.

sort the non-leaf vertices of T in a list L according to a post-order traversal of T;
$D \leftarrow \emptyset$;
while $L \neq \{r\}$ **do**
    $v \leftarrow$ the first vertex in L;
    $L \leftarrow L \setminus \{v\}$;
    if v has at least two unobserved children **then**
        $D \leftarrow D \cup \{v\}$;
        exhaustively apply the two observation rules to T;
        end if;
    end while;
if r is unobserved **then** $D \leftarrow D \cup \{r\}$.

Theorem 3.12 *Algorithm PowerDomTree gives a minimum power dominating set of a tree in linear time.*

Proof. First is to prove that the output D of the algorithm is a power dominating set. The proof is based on an induction on the depth of the vertices u in T, denoted by $\text{depth}(u)$. Note that the proof works top-down whereas the algorithm works bottom-up. For $\text{depth}(u) = 0$, it is clear that u, which is r, is observed due to the second "if"-condition of the algorithm. Suppose that all vertices u with $\text{depth}(u) < k$ with $k > 0$ are observed. Consider a vertex u with $\text{depth}(u) = k$, which is a child of the vertex v. If $v \in D$, then u is observed; otherwise, due to the induction hypothesis, v is observed. Moreover, if $\text{depth}(v) \geq 1$, then the parent of v is observed as well. Because of the first "if"-condition of the algorithm, v is not in D only if v has at most one unobserved child during the "while"-loop of the algorithm processing v. If vertex u is this only unobserved child, then it gets observed by applying OR2 to v, because v itself and all its neighbors (including the parent of v and its children) with the only exception of u are observed by the vertices in D. In summary, it is concluded that all vertices in T are observed by D.

Next is to prove the optimality of D by showing a more general statement:

Claim. Given a tree $T = (V, E)$ rooted at r, Algorithm PowerDomTree outputs a power dominating set D such that $|D \cap V_u| \leq |D' \cap V_u|$ for any minimum power dominating set D' of T and any tree vertex u, where V_u denotes the vertex set of the subtree of T rooted at u.

Proof of the Claim. Let $T_u = (V_u, E_u)$ denote the subtree of T rooted at vertex u. Let ℓ denote the depth of T, that is, $\ell := \max_{v \in V} \text{depth}(v)$. For each vertex $u \in V$, define $d_u := |D \cap V_u|$ and $d'_u := |D' \cap V_u|$. The claim is to be proved by an induction on the depth of tree vertices, starting with the maximum depth ℓ and proceeding to 0.

Since vertices u with $\text{depth}(u) = \ell$ are leaves and the algorithm adds no leaf to D, it is the case that $d_u = 0$ and so $d_u \leq d'_u$ for all u with $\text{depth}(u) = \ell$.

Suppose that $d_u \leq d'_u$ holds for all u with $\text{depth}(u) > k$ with $k < \ell$. Consider a vertex u with $\text{depth}(u) = k$. Let C_u denote the set of children of u. Then, for all $v \in C_u$, by the induction hypothesis, $d_v \leq d'_v$. In order to show $d_u \leq d'_u$, one only has to consider the case that

$$\text{(a) } u \in D, \quad \text{(b) } u \notin D', \quad \text{and (c) } \textstyle\sum_{v \in C_u} d_v = \sum_{v \in C_u} d'_v. \tag{1}$$

In all other cases, $d_u \leq d'_u$ always holds. In the following, it will be shown that this case does not apply. Assume that $u \neq r$. The argument works also for $u = r$.

From (c) and that $d_v \leq d'_v$ for all $v \in C_u$ (induction hypothesis), $d_v = d'_v$ for all $v \in C_u$. Moreover, (a) is true only if u has two unobserved children v_1 and v_2 during the "while"-loop of the algorithm processing u (the first "if"-condition). In other words, this means that vertices v_1 and v_2 are not observed by the vertices in $(D \cap V_u) \setminus \{u\}$. In the following, it is shown that u has to be included in D' in order for D' to be a valid power dominating set. To this end,

the following statement is needed:

$$\text{for all } x \in D' \cap V_{v_i} \text{ with } 1 \leq i \leq 2 \text{ there is some } x' \in W_{(v_i,x)} \cap D, \qquad (2)$$

where $W_{(v_i,x)}$ denotes the path between v_i and x (including v_i and x).

Without loss of generality, consider only $i = 1$. Assume that statement (2) is not true for a vertex $x \in D' \cap V_{v_1}$. Since $x \notin D$, one can infer that $d_x < d'_x$. Since no vertex from $W_{(v_1,x)}$ is in D and, by induction hypothesis, $d_y \leq d'_y$ for all $y \in V_{v_1}$, it follows that $d_{x'} < d'_{x'}$ for all vertices $x' \in W_{(v_1,x)}$, in particular, $d_{v_1} < d'_{v_1}$. This contradicts the fact that $d_{v_i} = d'_{v_i}$ for all children v_i of u. Thus, statement (2) is true.

By statement (2), for each vertex $x \in D' \cap V_{v_i}$ ($i \in \{1, 2\}$), there exists a vertex $x' \in D$ on the path $W_{(v_i,x)}$. Thus, if vertex v_i is observed by a vertex $x \in D' \cap V_{v_i}$, then there is a vertex $x' \in D \cap W_{(v_i,x)}$ that observes v_i. However, (a) in (1) is true only if v_1 and v_2 are unobserved by the vertices in $D \cap V_{v_1}$ and $D \cap V_{v_2}$. Altogether, this implies that v_1 and v_2 cannot be observed by the vertices in $D' \cap V_{v_1}$ and $D' \cap V_{v_2}$. Since D' is a power dominating set of T, vertex u is taken into D'; otherwise, u has two unobserved neighbors v_1 and v_2. OR2 can never be applied to u whatever vertices from $V \setminus V_u$ are in D'. It concludes that the case that $u \in D, u \notin D'$, and $\sum_{v \in C_u} d_v = \sum_{v \in C_u} d'_v$ does not apply. This completes the proof of the claim. □

Concerning the running time, the following explains how to implement the exhaustive applications of OR1 and OR2. Observe that, if OR2 is applicable to a vertex u during the bottom-up process, then all vertices in T_u can be observed at the current stage of the bottom-up process. In particular, after adding vertex u to D, all vertices in T_u can be observed. Thus, exhaustive applications of OR1 and OR2 to the vertices of T_u can be implemented as pruning T_u from T which can be done in constant time. Next, consider the vertices in $V \setminus V_u$. After adding u to D, the only possible application of OR1 is that u observes its parent v. Moreover, the applicability of OR2 to the vertices x lying on the path from u to r is checked, in the order of their appearance, the first v, and the last r. As long as OR2 is applicable to a vertex x on this path, x is deleted from the list L that is defined in the first line of the algorithm, and prune T_x from T.

The linear running time of the algorithm is then easy to see: The vertices to which OR2 is applied are removed from the list L immediately after the application of OR2 and are never processed by the instruction in the first line inside the "while"-loop of the algorithm. With proper data structures such as integer counters storing the number of observed neighbors of a vertex, the applications of the observation rules to a single vertex can be done in constant time. Post-order traversal of a rooted tree is clearly doable in linear time. ■

3.6 Tree related classes

Besides the methods demonstrated in the previous subsections, the "transformation method" sometimes is also used in the study of domination. Roughly

speaking, the method transforms the domination problem in certain graphs to another well-known problem, which is solvable. As this method depends on the variation of domination and the type of graphs, it will be mentioned only when it is used in the problem surveyed in this chapter.

There are some classes of graphs which have tree-like structures, including block graphs, (partial) k-trees and cacti.

A *block graph* is a graph whose blocks are complete graphs. For results on variations of domination in block graphs, see [41, 45, 128, 132, 204].

A *cactus* is a graph whose blocks are cycles. Hedetniemi [120] gave a linear-time algorithm for the domination problem in cacti.

For a positive integer k, *k-trees* are defined recursively as follows: (i) a complete graph of $k+1$ vertices is a k-tree; (ii) the graph obtained from a k-tree by adding a new vertex adjacent to a clique of k vertices is a k-tree. *Partial k-trees* are subgraphs of k-trees. For results on variations of domination in k-trees and partial k-trees, see [3, 68, 97, 173, 181, 190, 191].

4 Interval graphs

4.1 Interval orderings of interval graphs

Recall that an interval graph is the intersection graph of a family of intervals in the real line.

In many papers, people design algorithms or prove theorems for interval graphs by using the interval models directly. This very often is accomplished by ordering the intervals according to some nondecreasing order of their right (or left) endpoints.

For instance, suppose $G = (V, E)$ is an interval graph with an interval model

$$\{I_i = [a_i, b_i] : 1 \leq i \leq n\},$$

where $b_1 \leq b_2 \leq \ldots \leq b_n$. One can solve the domination problem for G by using exactly the same primal-dual algorithm for trees except replacing the 4th line of Algorithm DomTreePD by

$$\text{let } j \text{ be the largest index such that } v_j \in N[v_i];$$

In order to prove that the revised algorithm works for interval graphs, it is only necessary to show that S^* is a 2-stable set. Suppose to the contrary that S^* contains two vertices v_i and $v_{i'}$ with $i < i'$ and $d(v_i, v_{i'}) \leq 2$, say there is a vertex $v_k \in N[v_i] \cap N[v_{i'}]$. Consider the largest indexed vertex v_j of $N[v_i]$ as chosen in iteration i of the algorithm. Since $v_k \in N[v_i]$, $k \leq j$. Consider two cases.

Case 1. $j \leq i'$. In this case, $k \leq j \leq i'$. Then $b_k \leq b_j \leq b_{i'}$. Since $v_k \in N[v_{i'}]$, I_k intersects $I_{i'}$ and so $a_{i'} \leq b_k$. Therefore, $a_{i'} \leq b_j \leq b_{i'}$ and so I_j intersects $I_{i'}$.

Case 2. $i' < j$. In this case, $i < i' < j$. Then $b_i \leq b_{i'} \leq b_j$. Since $v_j \in N[v_i]$, I_i intersects I_j and so $a_j \leq b_i$. Therefore, $a_j \leq b_{i'} \leq b_j$ and so I_j intersects $I_{i'}$.

In any case, $v_j \in N[v_{i'}]$. As v_j is put into D^* in iteration i, when the algorithm processes $v_{i'}$, $N[v_{i'}] \cap D^* \neq \emptyset$ so that S^* does not contain $v_{i'}$, a contradiction. Therefore, S^* is a 2-stable set.

As one can see, the arguments in Cases 1 and 2 are quite similar. This is also true in many other proofs for interval graphs. One may expect that a unified property can be applied. This is in fact the so called interval ordering in the following theorem (see [176]). Notice that once property (IO) below holds, the conclusion $v_j \in N[v_{i'}]$ above follows immediately.

Theorem 4.1 *$G = (V, E)$ is an interval graph if and only if G has an interval ordering which is an ordering $[v_1, v_2, \ldots, v_n]$ of V satisfying*

$$i < j < k \text{ and } v_i v_k \in E \text{ imply } v_j v_k \in E. \tag{IO}$$

Proof. ($\Rightarrow$) Suppose G is the intersection graph of

$$\{I_i = [a_i, b_i] : 1 \leq i \leq n\}.$$

Without loss of generality, assume that $b_1 \leq b_2 \leq \ldots \leq b_n$. Suppose $i < j < k$ and $v_i v_k \in E$. Then $b_i \leq b_j \leq b_k$. Since $v_i v_k \in E$, it is the case that $I_i \cap I_k \neq \emptyset$ which implies that $a_k \leq b_i$. Thus, $a_k \leq b_j \leq b_k$ and so $I_j \cap I_k \neq \emptyset$, i.e., $v_j v_k \in E$.

($\Leftarrow$) On the other hand, suppose (IO) holds. For any $v_i \in V$, let i^* be the minimum index such that $v_{i^*} \in N[v_i]$ and let interval $I_i = [i^*, i]$. If $v_i v_k \in E$ with $i < k$, then $k^* \leq i < k$ and so $I_i \cap I_k \neq \emptyset$. If $I_i \cap I_k \neq \emptyset$ with $i < k$, say $j \in I_i \cap I_k$, then $k^* \leq j \leq i < k$. Since $v_{k^*} v_k \in E$, by (IO), $v_i v_k \in E$. Therefore, G is an interval graph with

$$\{I_i = [i^*, i] : 1 \leq i \leq n\}$$

as an interval model. ∎

The problem of recognizing interval graphs and giving their interval ordering is a fundamental problem. Several linear-time recognition algorithms for interval graphs have been developed in the literature, see Booth and Leuker [27], Korte and Mohring [145], Hsu and his coauthors [125, 126, 127, 182], Habib et al. [106] among many others. Note that a linear time algorithm may not be easily implementable and so there are still some efforts to search for new (simple) interval graph recognition algorithms and new ways to reconstruct an interval representation or just the interval ordering of a given interval graph. The 6-sweep LBFS algorithm by Corneil, Olariu and Stewart [72] is a such one. It is believed that a 3-sweep LBFS algorithm is possible.

4.2 Domatic numbers of interval graphs

Another interesting usage of the interval ordering is the following rewriting for the result on the domatic numbers of interval graphs obtained by Bertossi [25]. He transformed the domatic number problem on interval graphs to a network flow problem as follows. This is a typical example of the transformation method.

First, Bertossi's method is described as follows. Suppose $G = (V, E)$ is an interval graph, whose vertex set $V = \{1, 2, \ldots, n\}$, with an interval model $\{[a_i, b_i] : 1 \leq i \leq n\}$. Without loss of generality, assume that

$$\text{no two intervals share a common endpoint and } a_1 < a_2 < \ldots < a_n.$$

Two "dummy" vertices 0 and $n + 1$ are added to the graph with $b_0 < a_1$ and $b_n < a_{n+1}$. Then an acyclic directed network H is constructed as follows. The vertex set of H is $\{0, 1, 2, \ldots, n, n + 1\}$, and there is a directed edge (i, j) in H if and only if $j \in P(i) \cup Q(i)$, where

$P(i) = \{k : a_i < a_k < b_i < b_k\}$ and

$Q(i) = \{k : b_i < a_k$ and there is no h with $b_i < a_h < b_h < a_k\}$.

Figure 8 shows an example of H.

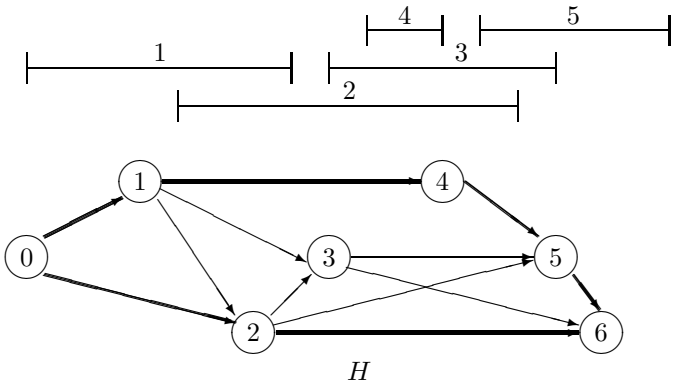


Figure 8: The construction of H for an interval graph of domatic number 2.

Bertossi then proved that any path from vertex 0 to vertex $n + 1$ in H corresponds to a proper dominating set of G and vice versa. His arguments have a flaw. In fact this statement is not true as $0, 2, 3, 5, 6$ is a path in the directed network H in Figure 8, but its corresponding dominating set $\{2, 3, 5\}$ has a proper subset $\{2\}$ that is also a dominating set. To be precise, he only showed that

(P1) a 0-$(n + 1)$ path in H corresponds to a dominating set of G, and

(P2) a proper dominating set of G corresponds to a 0-$(n + 1)$ path in H.

Besides, the entire arguments can be treated in terms of the interval ordering as follows. Now assume that $[0, 1, 2, \ldots, n, n+1]$ is an interval ordering of the graph $G = (V, E)$ with two isolated vertices 0 and $n+1$ added. Then construct a directed network H' with

vertex set $\{0, 1, 2, \ldots, n, n+1\}$ and

edge set $\{ij : i < j$ and $(i < h < j$ imply $ih \in E$ or $hj \in E)\}$.

Notice that H' is not the same as H. However, statements (P1) and (P2) remain true if H is replaced by H'. Also, there is a simpler proof using property (IO), which is different from their original proof using the endpoints of the intervals. The argument is as follows.

First, a 0-$(n+1)$ path $P : 0 = i_0, i_1, \ldots, i_r, i_{r+1} = n+1$ in H certainly corresponds to a dominating set $D = \{i_0, i_1, \ldots, i_r, i_{r+1}\}$ of G by the definition of the edge set of H'. This proves (P1). Conversely, for a proper dominating set D of G, consider the corresponding path P. For any $0 \leq s \leq r$, suppose $i_s < h < i_{s+1}$. Since D is a dominating set of G, there exists some $i_j \in D$ such that $hi_j \in E$.

Case 1. $j \leq s$. Then $i_s h \in E$ by (IO).

Case 2. $j = s+1$. Then $hi_{s+1} \in E$.

Case 3. $j > s+1$. If $N[i_{s+1}] \subseteq N[i_j]$, then $D \setminus \{v_{s+1}\}$ is a dominating set of G, violating that D is a proper dominating set. So, there is some vertex $k \in N[i_{s+1}] \setminus N[i_j]$. The case of $i_{s+1} < i_j < k$ or $h < k < i_j$ implies $k \in N[i_j]$ by (IO), a contraction. The case of $i_j < h < i_{s+1}$ implies $hi_{s+1} \in E$.

In any case, $i_s i_{s+1}$ is an arc in H. This proves (P2).

Primal-dual approaches were also used in [157, 188] to solve the domatic number problem in interval graphs. Manacher and Mankus [159] made it possible to get an $O(n)$ algorithm for the problem. Peng and Chang [169] used the primal-dual method to get a linear-time algorithm for the problem in strongly chordal graphs, see Section 6.3.

4.3 Weighted independent domination in interval graphs

There are many algorithms for variants of domination in interval graphs, see [17, 20, 50, 52, 55, 175, 176]. Among them, Ramalingam and Pandu Rangan [176] gave a unified approach to the weighted independent domination, the weighted domination, the weighted total domination and the weighted connected domination problems in interval graphs by using the interval orderings. Their algorithms are demonstrated in this and the following subsections.

Now, suppose $G = (V, E)$ is an interval graph with vertex set $V = \{1, 2, \ldots, n\}$, where $[1, 2, \ldots, n]$ is an interval ordering of G. Assume that each vertex is associated with a real number as its weight. Notice that except for independent domination, according to Lemma 2.1, assume that the weights are nonnegative. Consider following notation.

$V_i = \{1, 2, \ldots, i\}$ and G_i denotes the subgraph $G[V_i]$ induced by V_i.

V_0 is the empty set.

$\text{low}(i)$ = minimum element in $N[i]$.

$\text{maxlow}(i) = \max\{\text{low}(j) : \text{low}(i) \leq j \leq i\}$.

$L_i = \{\text{maxlow}(i), \text{maxlow}(i) + 1, \ldots, i\}$.

$M_i = \{j : j > i \text{ and } j \text{ is adjacent to } i\}$.

For any family X of sets of vertices, $\min\{X\}$ denotes a minimum-weighted set in X. If X is the empty set, then $\min\{X\}$ denotes a set of infinite weight.

Notice that the vertices in the set $\{1, 2, \ldots, \text{low}(i) - 1\}$ are not adjacent to i and the vertices in $\{\text{low}(i), \text{low}(i) + 1, \ldots, i\}$ are adjacent to i. The vertices in L_i form a maximal clique in the graph G_i. Let j be the vertex such that $\text{low}(i) \leq j \leq i$ and $\text{maxlow}(i) = \text{low}(j)$. Then, $\text{low}(i) \leq \text{low}(j) \leq j \leq i$. It can easily be seen that $N[j]$ is a subset of $L_i \cup M_i$. Furthermore, in G_i, j is adjacent only to the vertices in L_i.

Having all of these, it is ready to establish the solutions to weighted independent domination problem in interval graphs.

Let ID_i denote an independent dominating set of the graph G_i and let MID_i denote the minimum weighted ID_i.

Notice that, in G_i, the set L_i is a maximal clique and that there is a vertex in L_i which is not adjacent to any vertex in $V_i \setminus L_i$. Hence, any ID_i contains exactly one vertex j in L_i. Furthermore, it is necessary and sufficient that $ID_i \setminus \{j\}$ dominates $V_{\text{low}(j)-1}$ and contains no vertex adjacent to j. Hence, $ID_i \setminus \{j\}$ is an independent dominating set of $G_{\text{low}(j)-1}$. In other words, a set is an ID_i if and only if it is of the form $ID_{\text{low}(j)-1} \cup \{j\}$ for some j in L_i.

These give the following lemma.

Lemma 4.2 (a) $MID_0 = \emptyset$. (b) *For* $1 \leq i \leq n$,
$MID_i = \min\{MID_{\text{low}(j)-1} \cup \{j\} : j \in L_i\}$.

A linear-time algorithm for the weighted independent domination problem in interval graphs then follows. The detailed description is omitted as it is easy. Similarly, in the following three subsections, only recursive formulas for variations of domination in interval graphs are presented.

4.4 Weighted domination in interval graphs

Let D_i denote a subset of V that dominates V_i. Unlike independent domination, it is not necessary to restrict D_i as a subset of V_i. Let MD_i denote a minimum weighted D_i.

Since there is a vertex in L_i whose neighbors are all in $L_i \cup M_i$, the set D_i contains some vertex j in $L_i \cup M_i$. It is necessary and sufficient that the remaining set $D_i \setminus \{j\}$ dominates $V_{\text{low}(j)-1}$ since j dominates all vertices in

$V_i \setminus V_{\text{low}(j)-1}$ and no vertex in $V_{\text{low}(j)-1}$. (Note that if $j \in L_i \cup M_i$, then $\text{low}(j) \leq i$.) In other words, a set is a D_i if and only if it is of the form $D_{\text{low}(j)-1} \cup \{j\}$ for some j in $L_i \cup M_i$.

These give the following lemma.

Lemma 4.3 (a) $MD_0 = \emptyset$. (b) *For* $1 \leq i \leq n$,

$MD_i = \min\{MD_{\text{low}(j)-1} \cup \{j\} : j \in L_i \cup M_i\}$.

4.5 Weighted total domination in interval graphs

Let TD_i denote a subset of V that totally dominates V_i and let MTD_i be a minimum weighted TD_i. Let PD_i denote a subset of V that totally dominates $\{i\} \cup V_{\text{low}(i)-1}$ and let MPD_i be a minimum weighted PD_i.

As in domination, TD_i also includes some vertex j in $L_i \cup M_i$. If $j \in L_i$, then it is necessary and sufficient that the set $TD_i \setminus \{j\}$ totally dominates $V_{\text{low}(j)-1} \cup \{j\}$. If $j \in M_i$, then it is necessary and sufficient that the set $TD_i \setminus \{j\}$ totally dominates $V_{\text{low}(j)-1}$.

Similarly, any PD_i includes some vertex j adjacent to i. By the definition, $j \geq \text{low}(i)$. Hence, it is necessary and sufficient that the $PD_i \setminus \{j\}$ totally dominates $V_{\min\{\text{low}(i)-1,\text{low}(j)-1\}}$.

These give the following lemma.

Lemma 4.4 (a) $MTD_0 = \emptyset$. (b) *For* $1 \leq i \leq n$,

$MTD_i = \min\Big(\{MPD_j \cup \{j\} : j \in L_i\} \bigcup \{MTD_{\text{low}(j)-1} \cup \{j\} : j \in M_i\}\Big)$.

$MPD_i = \min\{MTD_{\min\{\text{low}(j)-1,\text{low}(i)-1\}} \cup \{j\} : j \in N(i)\}$.

Note that in the original paper by Ramalingam and Pandu Rangan [176], there is a typo that using $j \in N[i]$ rather than $j \in N(i)$ in the formula for MPD_i.

4.6 Weighted connected domination in interval graphs

Let CD_i denote a connected dominating set of G_i that contains the vertex i and let MCD_i denote a minimum weighted CD_i.

If $\text{low}(i) = 1$, then MCD_i is $\{i\}$ since all vertices have nonnegative weights. If $\text{low}(i) > 1$, then any CD_i contains vertices other than i, and hence some vertex adjacent to i in G_i. Let j be the maximum vertex in $CD_i \setminus \{i\}$. Assume that $\text{low}(j) < \text{low}(i)$. Otherwise j is removed to get a CD_i of the same or lower weight. If $\text{low}(j) < \text{low}(i)$, then any other vertex of G_j adjacent to i is also adjacent to j. Thus, it is necessary and sufficient that $CD_i \setminus \{i\}$ is a CD_j.

These give the following lemma.

Lemma 4.5 (a) *If* $\text{low}(i) = 1$, *then* $MCD_i = \{i\}$. (b) *For* $\text{low}(i) > 1$,

$MCD_i = \min\{MID_j \cup \{i\} : j \in N[i]$ *and* $j < i$ *and* $\text{low}(j) < \text{low}(i)\}$.

(c) $\min\{MCD_n : i \in L_n\}$ *is a minimum weighted connected dominating set of the graph* G.

5 Chordal graphs and NP-completeness results

5.1 Perfect elimination orderings of chordal graphs

It was seen in previous sections that the domination problem is well solved for trees and interval graphs. People then try to generalize the results for general graphs. However, the NP-completeness theory raised by Cook suggests that this is in general quite impossible. Garey and Johnson, in an unpublished paper (see [101]), pointed out that the dominating problem is NP-complete by transforming the vertex cover problem to it.

VERTEX COVER
INSTANCE: A graph $G = (V, E)$ and a positive integer $k \leq |V|$.
QUESTION: Is there a subset $C \subseteq V$ of size at most k such that each edge xy of G has either $x \in C$ or $y \in C$?

Their idea can in fact be modified to prove many NP-complete results for the domination problem and its variations in many classes of graphs. Among these classes, the class of chordal graphs is most interesting in the study of many graph optimization problems. Chordal graphs are raised in the theory of perfect graphs, see [103]. It contains trees, interval graphs, directed path graphs, split graphs, undirected path graphs, etc., as subclasses.

Recall that a graph is chordal if every cycle of length at least four has a chord. The following property is an important characterization of chordal graphs. The proof presented here is from Theorem 4.3 in [104], except that the Maximum Cardinality Search is not introduced explicitly.

Theorem 5.1 *A graph* $G = (V, E)$ *is chordal if and only if it has a perfect elimination ordering which is an ordering* $[v_1, v_2, \ldots, v_n]$ *of* V *such that*

$$i < j < k \text{ and } v_iv_j, v_iv_k \in E \text{ imply } v_jv_k \in E. \qquad \text{(PEO)}$$

Proof. ($\Rightarrow$) For any ordering $\sigma = [v_1, v_2, \ldots, v_n]$, define the vector

$$d(\sigma) = (d_n, d_{n-1}, \ldots, d_1),$$

where each $d_s = |\{t : t > s \text{ and } v_t \text{ is adjacent to } v_s\}|$. Choose an ordering σ such that $d(\sigma)$ is lexicographically largest.

Suppose $p < q < r$ and $v_r \in N(v_p) \setminus N(v_q)$. Consider the ordering σ' obtained from σ by interchanging v_p and v_q. Then $d'_s = d_s$ for all $s > q$ and

$$|\{t : t > q \text{ and } v_t \in N(v_p)\}| = d'_q \leq d_q = |\{t : t > q \text{ and } v_t \in N(v_q)\}|.$$

However, r is a vertex such that $r > q$ and $v_r \in N(v_p) \setminus N(v_q)$. Then, there exists some $s > q$ such that $v_s \in N(v_q) \setminus N(v_p)$. This gives

$$p < q < r, v_r \in N(v_p) \setminus N(v_q) \text{ imply } v_s \in N(v_q) \setminus N(v_p) \text{ for some } s > q. \quad (*)$$

Next, $(*)$ is used to prove the following claim.

Claim. There does not exist any chordless path $P : v_{i_1}, v_{i_2}, \ldots, v_{i_x}$ with $x \geq 3$ and $i_y < i_x < i_1$ for $1 < y < x$.

(Notice that the claim implies (PEO) as v_k, v_i, v_j is a chordless path with $i < j < k$ whenever $v_j v_k \notin E$.) Suppose to the contrary that such a path P exists. Choose one with a largest i_x. Since $i_2 < i_x < i_1$ and $v_{i_1} \in N(v_{i_2}) \setminus N(v_{i_x})$, by $(*)$, there exists some $i_{x+1} > i_x$ such that $v_{i_{x+1}} \in N(v_{i_x}) \setminus N(v_{i_2})$. Let z be the minimum index such that $z \geq 2$ and $v_{i_{x+1}} v_z \in E$. Note that z exists and $z \geq 3$. For the case when $v_{i_1} v_{i_{x+1}} \notin E$, $P' : v_{i_1}, v_{i_2}, \ldots, v_{i_{z-1}}, v_{i_z}, v_{i_{x+1}}$ (or its inverse) is a chordless path of length at least three with $i_y < i_{x+1} < i_1$ (or $i_y < i_1 < i_{x+1}$) for $1 < y \leq z$. In this case, $i_x < i_{x+1}$ is a contradiction to the choice of P. For the case when $v_{i_1} v_{i_{x+1}} \in E$, P' together with the edge $v_{i_1} v_{i_{x+1}}$ form a chordless cycle of length at least four, a contradiction to the fact that G is chordal.

($\Leftarrow$) On the other hand, suppose (PEO) holds. For any cycle of length at least four, choose the vertex of the cycle with the least index. By (PEO), the two neighbors of this vertex in the cycle are adjacent. ∎

5.2 NP-completeness for domination

This section first demonstrates Garey and Johnson's proof that the domination problem is NP-complete. The proof is adapted for split graphs, which are special chordal graphs. A *split graph* is a graph whose vertex set is the disjoint union of a clique C and a stable set S. Notice that a split graph is chordal as the ordering with the vertices in S first and the vertices in C next gives a perfect elimination ordering.

Theorem 5.2 *The domination problem is NP-complete for split graphs.*

Proof. The proof is given by transforming the vertex cover problem in general graphs to the domination problem in split graphs. Given a graph $G = (V, E)$, construct the graph $G' = (V', E')$ with

vertex set $V' = V \cup E$ and

edge set $E' = \{v_1 v_2 : v_1 \neq v_2 \text{ in } V\} \cup \{ve : v \in e\}$.

Notice that G' is a split graph whose vertex set V' is the disjoint union of the clique V and the stable set E.

If G has a vertex cover C of size at most k, then C is a dominating set of G' of size at most k, by the definition of G'. On the other hand, suppose G' has a

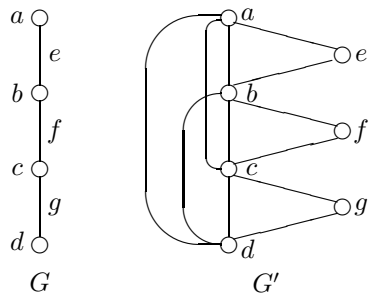


Figure 9: A transformation to a split graph.

dominating set D of size at most k. If D contains any $e \in E$, say $e = uv$, then replace e with u to get a new dominating set of size at most k. In this way, assume that D is a subset of V. It is then clear that D is a vertex cover of G of size at most k.

Since the vertex cover problem is NP-complete, the domination problem is also NP-complete for split graphs. ∎

Note that the dominating set of G' in the proof above in fact induces a connected subgraph.

Corollary 5.3 *The total and the connected domination problems are NP-complete for split graphs.*

In fact, the proof can be modified to get

Theorem 5.4 *The domination problem is NP-complete for bipartite graphs.*

Proof. The vertex cover problem in general graphs is transformed to the domination problem in bipartite graphs as follows. Given a graph $G = (V, E)$, construct the graph $G' = (V', E')$ with

vertex set $V' = \{x, y\} \cup V \cup E$ and

edge set $E' = \{xy\} \cup \{yv : v \in V\} \cup \{ve : v \in e\}$.

Notice that G' is a bipartite graph whose vertex set V' is the disjoint union of two stable sets $\{x\} \cup V$ and $\{y\} \cup E$.

If G has a vertex cover C of size at most k, then $\{y\} \cup C$ is a dominating set of G' of size at most $k + 1$. On the other hand, suppose G' has a dominating set D of size at most $k + 1$. Since $N_{G'}[x] = \{x, y\}$, D must contain x or y. One may assume that D contains y but not x, as $(D \setminus \{x\}) \cup \{y\}$ is also a dominating set of size at most $k + 1$. Since $y \in D$, if D contains any $e \in E$, say $e = uv$, one can replace e with u to get a new dominating set of size at most $k + 1$. In this way, one may assume that $D \setminus \{y\}$ is a subset of V. It is then clear that $D \setminus \{y\}$ is a vertex cover of G of size at most k.

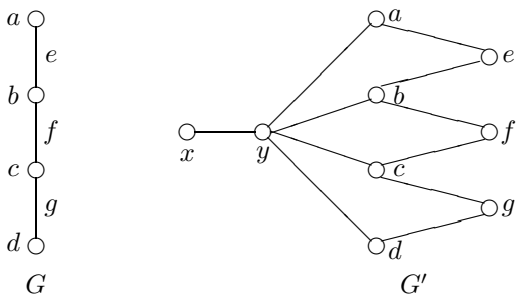


Figure 10: A transformation to a bipartite graph.

Since the vertex cover problem is NP-complete, the domination problem is NP-complete for bipartite graphs. ∎

Corollary 5.5 *The total and the connected domination problems are NP-complete for bipartite graphs.*

There are many other NP-complete results for variations of domination, see [13, 16, 26, 68, 78, 82, 98, 116, 128, 166, 203, 204] and [101, 133, 134]. Most of the proofs are more or less similar to the above two. Only very few are proved by using different methods. As an example, Booth and Johnson [26] proved that the domination problem is NP-complete for undirected path graphs, which is another subclass of chordal graphs, by reducing the 3-dimensional matching problem to it.

Theorem 5.6 *The domination problem is NP-complete for undirected path graphs.*

Proof. Consider an instance of the 3-dimensional matching problem, in which there are three disjoint sets W, X and Y each of size q and a subset

$$M = \{m_i = (w_r, x_s, y_t) : w_r \in W, x_s \in X \text{ and } y_t \in Y \text{ for } 1 \leq i \leq p\}$$

of $W \times X \times Y$ having size p. The problem is to find a subset M' of M having size exactly q such that each $w_r \in W$, $x_s \in X$ and $y_t \in Y$ occurs in precisely one triple of M'.

Given an instance of the 3-dimensional matching problem, construct a tree $\mathcal{T}$ having $6p+3q+1$ vertices from which an undirected path graph G is obtained. The vertices of the tree, which are represented by sets, are explained below.

For each triple $m_i \in M$ there are six vertices depend only upon the triple itself and not upon the elements within the triple:

$\{A_i, B_i, C_i, D_i\}$
$\{A_i, B_i, D_i, F_i\}$

$\{C_i, D_i, G_i\}$
$\{A_i, B_i, E_i\}$
$\{A_i, E_i, H_i\}$
$\{B_i, E_i, I_i\}$ for $1 \leq i \leq p$.

These six vertices form the subtree of $\mathcal{T}$ corresponding to m_i, which is illustrated in Figure 11. Next, there is a vertex for each element of W, X and Y that depends upon the triples of M to which each respective element belongs:

$\{R_r\} \cup \{A_i : w_r \in m_i\}$ for $w_r \in W$,
$\{S_s\} \cup \{B_i : x_s \in m_i\}$ for $x_s \in X$,
$\{T_t\} \cup \{C_i : y_t \in m_i\}$ for $y_t \in Y$.

Finally, $\{A_i, B_i, C_i : 1 \leq i \leq p\}$ is the last vertex of $\mathcal{T}$. The arrangement of these vertices in the tree $\mathcal{T}$ is shown in Figure 11. This then results in an undirected path graph G with vertex set

$$\{A_i, B_i, C_i, D_i, E_i, F_i, G_i, H_i, I_i : 1 \leq i \leq p\} \cup \{R_j, S_j, T_j : 1 \leq j \leq q\}$$

of size $9p + 3q$, where the undirected path in $\mathcal{T}$ corresponding to a vertex v of G consists of those vertices (sets) containing v in the tree $\mathcal{T}$.

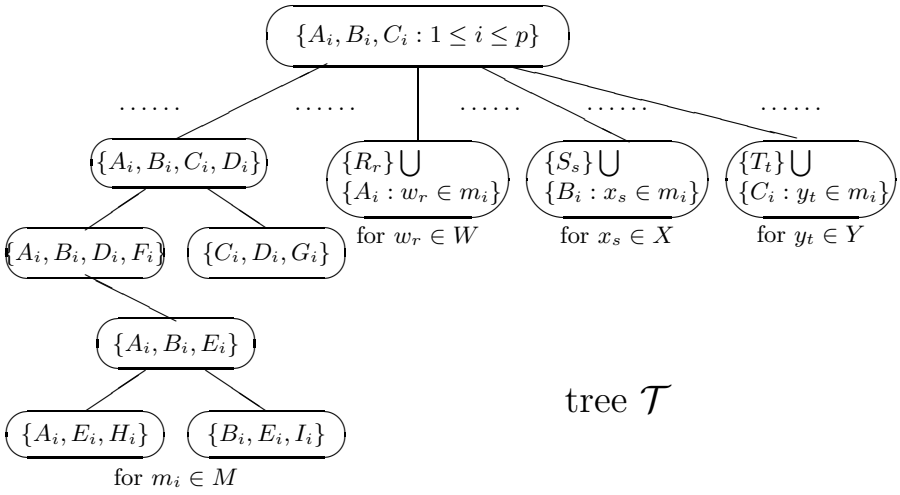


Figure 11: A transformation to an undirected path graph.

The theorem then follows from the claim that G has a dominating set of size $2p + q$ if and only if the 3-dimensional matching problem has a solution.

Suppose D is a dominating set of G of size $2p + q$. Observe that for any i, the only way to dominate the vertex set $\{A_i, B_i, C_i, C_i, D_i, E_i, F_i, G_i, H_i, I_i\}$ corresponding to m_i with two vertices is to choose D_i and E_i, and that any larger dominating set might just as well consist of A_i, B_i and C_i, since none of the other possible vertices dominate any vertex outside of the set. Consequently,

D consists of A_i, B_i and C_i for t m_i's, and D_i and E_i for $p-t$ other m_i's, and at least $\max\{3(q-t), 0\}$ R_r, S_s, T_t. Then,

$$2p + q = |D| \geq 3t + 2(p-t) + 3(q-t) = 2p + 3q - 2t$$

and so $t \geq q$. Picking q triples m_i for which A_i, B_i and C_i are in D form a matching M' of size q.

Conversely, suppose the 3-dimensional matching problem has a solution M' of size q. Let

$$D = \{A_i, B_i, C_i : m_i \in M'\} \cup \{D_i, E_i : m_i \in M \setminus M'\}.$$

It is straightforward to check that D is a dominating set of G of size $3q + 2(p-q) = 2p + q$. ∎

5.3 Independent domination in chordal graphs

Farber [93] showed a surprising result that the independent domination problem is solvable by using a linear programming method. On the other hand, it is known [66] that the weighted independent domination problem is NP-complete. The proof has the same spirit of the proof of Theorem 5.2.

Theorem 5.7 *The weighted independent domination problem is NP-complete for chordal graphs.*

Proof. The vertex cover problem in general graphs is transformed to the weighted independent domination problem in chordal graphs as follows. Given a graph $G = (V, E)$, construct the following chordal graph $G' = (V', E')$ with

vertex set $V' = \{v'', v', v : v \in V\} \cup E$ and

edge set $E' = \{v''v', v'v : v \in V\} \cup \{ve : v \in e\} \cup \{e_1e_2 : e_1 \neq e_2 \text{ in } E\}$.

The weight of each $e \in E$ is $2|V| + 1$ and the weight of each vertex in V' is 1.

If G has a vertex cover C of size at most k, then $\{v'', v : v \in C\} \cup \{v' : v \in V \setminus C\}$ is an independent dominating set of G' with weight at most $|V| + k$. On the other hand, suppose G' has an independent dominating set D of weight at most $|V| + k$. As $k \leq |V|$, D contains no elements in E. Let $C = D \cap V$. It is clear that C is a vertex cover of G. Also, for each $v \in V$ the set D contains exactly one vertex in $\{v'', v'\}$. Thus C is of size at most k.

Since the vertex cover problem is NP-complete, the weighted independent domination problem is NP-complete for chordal graphs. ∎

Farber's algorithm for the independent domination problem in chordal graphs is by means of a linear programming method. Suppose $G = (V, E)$ is a chordal

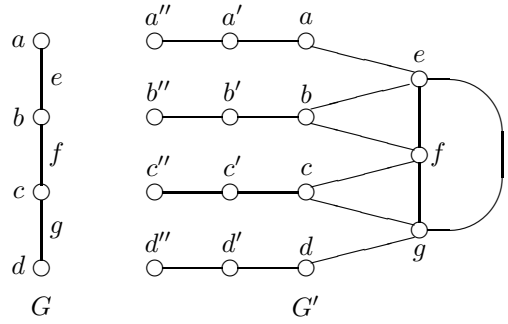


Figure 12: A transformation to a weighted chordal graph.

graph with a perfect elimination ordering $[v_1, v_2, \ldots, v_n]$ and vertex weights $w_1, w_2, \ldots, w_n$ of real numbers. Write

$$\begin{aligned} i \sim j \quad & \text{for} \quad v_i \in N[v_j], \\ i \widetilde{<} j \quad & \text{for} \quad i \sim j \text{ and } i \leq j, \\ i \widetilde{>} j \quad & \text{for} \quad i \sim j \text{ and } i \geq j. \end{aligned}$$

It follows from the definition of a perfect elimination ordering that each

$$C_j = \{v_i : i \widetilde{>} j\}$$

is a clique. Thus a set S of vertices is independent if and only if each C_j contains at most one vertex of S. Also, S is a dominating set if and only if for each j, S contains at least one vertex v_i with $i \sim j$. Consider the following linear problem:

$$\begin{array}{lll} P_1(G,w): \text{ Minimize} & \sum_{i=1}^{n} w_i x_i, & \\ \text{subject to} & \sum_{i \sim j} x_i \geq 1 & \text{for each } j, \\ & \sum_{i \widetilde{>} j} x_i \leq 1 & \text{for each } j, \\ & x_i \geq 0 & \text{for each } i. \end{array}$$

It follows from the above comments that there is a one-to-one correspondence between independent dominating sets of G and feasible 0-1 solutions to $P_1(G,w)$. Moreover, an optimal 0-1 solution to $P_1(G,w)$ corresponds to a minimum weighted independent dominating set in G. Notice that a set of vertices of G is an independent dominating set in G if and only if it is a maximal independent set in G. Consequently, there exist independent dominating sets, and $P_1(G,w)$ is a feasible linear program. It will follow from the algorithm presented below that if every vertex of G has a weight in $\{1, 0, -1, -2, \ldots\}$ then $P_1(G,w)$ has an optimal 0-1 solution, and that the following dual program has an integer optimal solution:

$$
\begin{array}{lll}
D_1(G,w): \quad \text{Maximize} & \sum_{j=1}^{n}(y_j - z_j), & \\
\text{subject to} & \sum_{j\sim i} y_j - \sum_{j\widetilde{<}i} z_j \le w_i & \text{for each } i, \\
& y_j, z_j \ge 0 & \text{for each } j.
\end{array}
$$

If, however, not all vertex weights are in $\{1, 0, -1, -2, \ldots\}$ (even though they are all integers), then it may be the case that neither $P_1(G,w)$ nor $D_1(G,w)$ has an integer solution. For the reminder of this section, assume that each vertex has a weight in $\{1, 0, -1, -2, \ldots\}$.

First define two functions which simplify the presentation of the algorithm. For each i, let

$$f(i) = w_i + \sum\nolimits_{j\widetilde{<}i} z_j - \sum\nolimits_{j\sim i} y_j,$$

$$g(i) = w_i + \sum\nolimits_{j\widetilde{<}i} z_j - \sum\nolimits_{j\widetilde{<}i} y_j.$$

Note that $f(i)$ is the slack in the dual constraint associated with vertex v_i.

The algorithm to locate a minimum weighted dominating set in G has two stages. Stage 1 finds a feasible solution to $D_1(G,w)$ by scanning the vertices in the order $v_1, v_2, \ldots, v_n$; and stage 2 uses this solution to find a feasible 0–1 solution to $P_1(G,w)$ by scanning the vertices in the order $v_n, v_{n-1}, \ldots, v_1$. Initially, each $y_j = 0$, each $z_j = 0$ and each $x_i = 2$. (The interpretation of $x_i = 2$ is that x_i has not yet been assigned a value.) If, at the time v_j is scanned in stage one, the dual constraint associated with v_j is violated, i.e., if $f(j) < 0$, then add just enough to z_j to bring that constraint into feasible. Otherwise, add as much as possible to y_j without violating the dual constraint associated with v_j or with any previously scanned vertex. In stage two, if $x_i = 2$ and $g(i) = 0$ when v_i is scanned, then let $x_i = 1$ and let $x_j = 0$ for each v_j adjacent to v_i.

A formal description of the algorithm is given below.

Algorithm IndDomChordal. Determine $\gamma_{\mathrm{i}}(G,w)$ for a chordal graph G with weights $w_1, w_2, \ldots, w_n$ in $\{1, 0, -1, -2, \ldots\}$.
Input. A chordal graph G with a perfect elimination ordering $[v_1, v_2, \ldots, v_n]$ and vertex weights $w_1, w_2, \ldots, w_n$ in $\{1, 0, -1, -2, \ldots\}$.
Output. Optimal solutions to $P_1(G,w)$ and $D_1(G,w)$.
Method.

each $y_j \leftarrow 0$, each $z_j \leftarrow 0$ and each $x_i \leftarrow 2$;
Stage 1: **for** $j = 1$ **to** n **do**
  if $f(j) < 0$ **then** $z_j \leftarrow -f(j)$
  else $y_j \leftarrow \min\{f(k) : k\widetilde{<}j\}$;
Stage 2: **for** $i = n$ **to** 1 **by** -1 **do**
  if $x_i = 2$ and $g(i) = 0$ **then**
    $x_i \leftarrow 1$;
    $x_j \leftarrow 0$ for each $v_j \in N(v_i)$;
    end if.

The validity of the algorithm is established below.

Theorem 5.8 *Algorithm IndDomChordal finds a minimum weighted independent dominating set of a chordal graph with vertex weights in* $\{1, 0, -1, -2, \ldots\}$.

Several lemmas are needed. Note that since the weights are integral, all $x_i, y_j, z_j, f(i), g(i)$ are integral at any time.

Lemma 5.9 *For each* j, $f(j) \geq 0$ *at all times after scanning* v_j *in stage* 1.

Proof. The fact that $f(j) \geq 0$ immediately after scanning v_j in stage 1 follows from the choice of z_j and y_j. The fact that $f(j) \geq 0$ after scanning each v_k where $k > j$ follows from the choice of y_k. ∎

Lemma 5.10 *At the end of stage one,* $y_j \geq 0$ *for any* j.

Proof. This is an immediate consequence of Lemma 5.9. ∎

Lemma 5.11 *At the end of stage* 1, $0 \leq f(j) \leq g(j)$ *for any* j.

Proof. This follows from Lemmas 5.9 and 5.10. ∎

Lemma 5.12 *At the end of stage* 1, *for each* i, *there is a* $j \widetilde{<} i$ *such that* $g(j) = 0$.

Proof. According to Lemma 5.11, $g(i) \geq 0$. If $g(i) = 0$, then the lemma is true as j can be chosen to be i. Suppose $g(i) > 0$. Then $f(i)$ was positive immediately after scanning v_i, and so, by the choice of z_i and y_i, $z_i = 0$ and y_i was chosen to force $f(j)$ to be 0 for some $j \widetilde{<} i$. Thus there is some $j \widetilde{<} i$ such that

$$\sum\nolimits_{k \sim j, k \leq i} y_k - \sum\nolimits_{k \widetilde{<} j} z_j = w_j.$$

If $y_k = 0$ for each k such that $k \widetilde{>} j$ and $k \leq i$, then $g(j) = 0$. Otherwise, choose k such that $y_k > 0$, $k \widetilde{>} j$ and $k \leq i$. Then $i \sim k$ since $i \widetilde{>} j$, $k \widetilde{>} j$ and $[v_1, v_2, \ldots, v_n]$ is a perfect elimination ordering. Hence $k \widetilde{<} i$ since $i \geq k$ and $i \sim k$. Since all vertex weights are integral, y_k is an integer. Thus $y_k \geq w_i$, since $w_i \in \{1, 0, -1, -2, \ldots\}$. Now, $g(i) > 0$, $y_k \geq w_i$ and $k \widetilde{<} i$, and hence there is some $\ell \widetilde{<} i$ such that $z_\ell > 0$. By the choice of z_ℓ, it is the case that $g(\ell) = 0$. ∎

Note that the proof of Lemma 5.12 relies upon the fact that G has vertex weights in $\{1, 0, -1, -2, \ldots\}$ to show that if $y_k > 0$ then $y_k \geq w_i$.

Lemma 5.13 *At the end of stage* 2, *for each* j, *if* $g(j) = 0$ *then* $x_k = 1$ *for some* $k \widetilde{>} j$.

Proof. It is clear from the instructions in stage two that, for each i, if $x_i = 1$ at any time during the algorithm then $x_i = 1$ at the end of the algorithm. Suppose $g(j) = 0$. If x_j was 2 just prior to scanning v_j in stage two then x_j was assigned the value of 1 when v_j was scanned, and hence $x_j = 1$ at the end of the algorithm. In this case, choose $k = j$ as desired. Otherwise, x_j was 0 prior to scanning v_j. In that case, $x_j = 0$ by virtue of the fact that $x_k = 1$ for some previously scanned neighbor v_k of v_j, i.e., $x_k = 1$ for some $k \widetilde{>} j$. ∎

Proof of Theorem 5.8. It is easy to see that Algorithm IndDomChordal halts after $O(|V|+|E|)$ operations. Next to show that the final values of $x_1, x_2, \ldots, x_n$ and $y_1, y_2, \ldots, y_n, z_1, z_2, \ldots z_n$ are feasible solutions to $P_1(G, w)$ and $D_1(G, w)$ respectively, and then verify that these solutions satisfy the conditions of complementary slackness. It then follows that they are optimal.

(i) Feasibility of dual solution: Clearly $z_j \geq 0$ for each j. By Lemmas 5.9 and 5.10, $y_j \geq 0$ and $f(j) \geq 0$ for each j.

(ii) Feasibility of primal solution: The instructions in stage two guarantee that if $x_i = x_j = 1$ then $v_i v_j \notin E$, and if $x_i = 0$ then $x_j = 1$ for some $j \sim i$. Since a 0–1 primal solution is feasible if and only if the set $\{v_j : x_j = 1\}$ is an independent dominating set in G, it suffices to show that each x_i is either 0 or 1. According to Lemma 5.12, for each i, there is a $j \widetilde{<} i$ such that $g(j) = 0$. According to Lemma 5.13, there is a $k \widetilde{>} j$ such that $x_k = 1$. Since $i \widetilde{>} j$, $k \widetilde{>} j$ and $[v_1, v_2, \ldots, v_n]$ is a perfect elimination ordering, it follows that $i \sim k$. If $i = k$ then $x_i = 1$; otherwise $x_i = 0$.

(iii) Complementary slackness: If $x_i > 0$ then $g(i) = 0$ by the choice of x_i, and hence $f(i) = 0$, by Lemma 5.11. Thus $\sum_{j \sim i} y_j - \sum_{j \widetilde{<} i} z_j = w_i$.

If $z_j > 0$ then $g(j) = 0$ by the choice of z_j, and so $x_k = 1$ for some $k \widetilde{>} j$, by Lemma 5.13. Hence $\sum_{i \widetilde{>} j} x_i \geq 1$. Equality follows from the fact that the primal solution is feasible.

Suppose $y_j > 0$ but $x_i = x_k = 1$ for $i \sim j$ and $k \sim j$. Since $x_i = x_k = 1$, $g(i) = g(k) = 0$, and so $j \leq \min\{i, k\}$ by Lemma 5.9, Lemma 5.11 and the choice of y_j. Thus $i \widetilde{>} j$ and $k \widetilde{>} j$, which imply $i \sim k$ since $[v_1, v_2, \ldots, v_n]$ is a perfect elimination ordering. On the other hand, $v_i v_k \notin E$ since $x_i = x_k = 1$. Consequently $i = k$. Hence, if $y_j > 0$ then $\sum_{i \sim j} x_i \leq 1$. Equality follows from the fact that the primal solution is feasible. ∎

6 Strongly chordal graphs

6.1 Strong elimination orderings of strongly chordal graphs

Strongly chordal graphs were introduced by several people [46, 95, 122] in the study of domination. In particular, most variations of the domination problem are solvable in this class of graphs. There are many equivalent ways to define them. This section adapts the notation from Farber's paper [95].

A graph $G = (V, E)$ is *strongly chordal* if it admits a *strong elimination ordering* which is an ordering $[v_1, v_2, \ldots, v_n]$ of V such that the following two conditions hold.

(a) If $i < j < k$ and $v_i v_j, v_i v_k \in E$, then $v_j v_k \in E$.

(b) If $i < j < k < \ell$ and $v_i v_k, v_i v_\ell, v_j v_k \in E$, then $v_j v_\ell \in E$.

Notice that an ordering satisfying condition (a) is a perfect elimination ordering. Hence, every strong elimination ordering is a perfect elimination ordering, and every strongly chordal graph is chordal. The notion $i \sim j$ stands for $v_i \in N[v_j]$.

The following equivalent definition of the strong elimination ordering is more convenient in many arguments for strongly chordal graph.

Lemma 6.1 *An ordering $[v_1, v_2, \ldots, v_n]$ of the vertices of G is a strong elimination ordering of G if and only if*

$$i \le j, k \le \ell, i \sim k, i \sim \ell \text{ and } j \sim k \text{ imply } j \sim \ell. \qquad \text{(SEO)}$$

Proof. Suppose $[v_1, v_2, \ldots, v_n]$ is a strong elimination ordering of G. Suppose that $i \le j$, $k \le \ell$, $i \sim k$, $i \sim \ell$ and $j \sim k$. If $i = j$ or $k = \ell$ or $j = \ell$, then clearly $j \sim \ell$. Suppose, on the other hand, that $i < j$, $k < \ell$ and $j \ne \ell$. By symmetry, assume that $i \le k$. Now consider three cases.

Case 1. Suppose $j = k$. Then $i < j < \ell$ and $v_i v_j, v_i v_\ell \in E$, whence $v_j v_\ell \in E$, by (a) in the definition of a strong elimination ordering.

Case 2. Suppose $j < k$. Then $i < j < k < \ell$ and $v_i v_k, v_i v_\ell, v_j v_k \in E$, whence $v_j v_\ell \in E$, by (b) in the definition of a strong elimination ordering.

Case 3. Suppose $k < j$. If $i = k$, then $v_k v_\ell \in E$. Otherwise, $i < k < \ell$ and $v_i v_k, v_i v_\ell \in E$, whence $v_k v_\ell \in E$. Consequently, $v_k v_\ell, v_k v_j \in E$ and either $k < \ell < j$ or $k < j < \ell$. In either case, $v_j v_\ell \in E$.

This completes the proof of necessity. The proof of sufficiency is trivial and thus omitted. ∎

6.2 Weighted domination in strongly chordal graphs

There are quite a few algorithms designed for variants of the domination problem in strongly chordal graphs, see [36, 40, 46, 47, 95, 122, 146, 195]. This section presents the linear algorithm, given by Fraber [95], for locating a minimum weighted dominating set in a strongly chordal graph.

Let $G = (V, E)$ be a strongly chordal graph with a strong elimination ordering $[v_1, v_2, \ldots, v_n]$ and vertex weights $w_1, w_2, \ldots, w_n$. According to Lemma 2.1, assume that these vertex weights are nonnegative. Consider the following linear problem:

$$\begin{array}{lllll} P_2(G, w): & \text{Minimize} & \sum_{i=1}^{n} w_i x_i, & & \\ & \text{subject to} & \sum_{i \sim j} x_i \ge 1 & \text{for each } j, \\ & & x_i \ge 0 & \text{for each } i. \end{array}$$

By definition, a set S of vertices of G is a dominating set if and only if, for each j, S contains some vertex v_i such that $i \sim j$. Consequently, there is a one-to-one correspondence between feasible 0–1 solutions to $P_2(G, w)$ and dominating sets in G. Moreover, an optimal 0–1 solution to $P_2(G, w)$ corresponds to a minimum weighted dominating set in G. It follows from the algorithm below that $P_2(G, w)$ has an optimal 0–1 solution and that the following dual program has an optimal solution:

$$\begin{array}{lll} D_2(G,w): \text{ Maximize} & \sum_{j=1}^{n} y_j, & \\ \text{subject to} & \sum_{j\sim i} y_j \le w_i & \text{for each } i, \\ & y_j \ge 0 & \text{for each } j. \end{array}$$

The algorithm presented below solves the linear programs $P_2(G, w)$ and $D_2(G, w)$. To simplify the presentation of the algorithm, define a function h and a family of sets. For each i, let

$$h(i) = w_i - \sum\nolimits_{j\sim i} y_j \quad \text{and} \quad T_i = \{j : i \sim j \text{ and } y_j > 0\}.$$

Note that $h(i)$ is the slack in the dual constraint associated with vertex v_i, and T_i is the set of constraints in $P_2(G, w)$ containing x_i which must be at equality to satisfy the conditions of complementary slackness.

The algorithm has two stages. Stage 1 finds a feasible solution to $D_2(G, w)$ by scanning the vertices in the order $v_1, v_2, \ldots, v_n$; and stage 2 uses this solution to find a feasible 0–1 solution to $P_2(G, w)$ by scanning the vertices in the order $v_n, v_{n-1}, \ldots, v_1$. This algorithm uses a set T to assure that the conditions of complementary slackness are satisfied. Initially, $T = \{1, 2, \ldots, n\}$, each $y_j = 0$ and each $x_i = 0$. When v_j is scanned in stage 1, y_j increases as much as possible without violating the dual constraint. In stage 2, if $h(i) = 0$ and $T_i \subseteq T$ when v_i is scanned then let $x_i = 1$ and replace T by $T \setminus T_i$. Otherwise x_i remains 0. A more formal description of the algorithm follows.

Algorithm WDomSC. Determine $\gamma(G, w)$ for a strongly chordal graph G with nonnegative vertex weights $w_1, w_2, \ldots, w_n$.
Input. A strongly chordal graph G with a strong elimination ordering [v_1, v_2, …, v_n] and nonnegative vertex weights $w_1, w_2, \ldots, w_n$.
Output. Optimal solutions to $P_2(G, w)$ and $D_2(G, w)$.
Method.

$T \leftarrow \{1, 2, \ldots, n\}$; each $y_j \leftarrow 0$; each $x_i \leftarrow 0$;
Stage 1: **for** $j = 1$ **to** n **do**
$y_j \leftarrow \min\{h(k) : k \sim j\}$;
Stage 2: **for** $i = n$ **to** 1 **by** -1 **do**
if ($h(i) = 0$ and $T_i \subseteq T$) **then**
$x_i \leftarrow 1$;
$T \leftarrow T \setminus T_i$;
end if.

The algorithm is verified as follows.

Theorem 6.2 *Algorithm WDomSC finds a minimum weighted dominating set of a strongly chordal graph with nonnegative vertex weights in linear time provided that a strong elimination ordering is given.*

Proof. It is easy to see that the algorithm halts after $O(|V|+|E|)$ operations. In order to show that the final values of $x_1, x_2, \ldots, x_n$ and $y_1, y_2, \ldots, y_n, z_1, z_2, \ldots z_n$ are optimal solutions to $P_2(G, w)$ and $D_2(G, w)$, respectively, it suffices to show that these solutions are feasible and that they satisfy the conditions of complementary slackness.

(i) Feasibility of dual solution: The instructions in stage 1 guarantee that $h(i) \geq 0$ for each i and $y_j \geq 0$ for each j.

(ii) Feasibility of primal solution: Clearly, each x_i is either 0 or 1. Thus it suffices to show that for each j, $x_i = 1$ for some $i \sim j$. By the choice of y_j, there is a $k \sim j$ such that $h(k) = 0$ and $\max T_k \leq j$. If $x_k = 1$, then the claim holds. Otherwise, by the algorithm, T_k was not contained in T when v_k was scanned in stage 2. Since, in stage 2, the vertices are scanned in the order $v_n, v_{n-1}, \ldots, v_1$, there is some $\ell > k$ such that $x_\ell = 1$ and $T_\ell \cap T_k \neq \emptyset$. Let $i \in T_\ell \cap T_k$. Then $i \leq j$ since $\max T_k \leq j$. Thus $i \leq j$, $k < \ell$, $i \sim k$, $i \sim \ell$ and $j \sim k$, whence $\ell \sim j$ by Lemma 6.1, since $[v_1, v_2, \ldots, v_n]$ is a strong elimination ordering. Hence $\ell \sim j$ and $x_\ell = 1$. Consequently, the primal solution is feasible.

(iii) Complementary slackness: If $x_i > 0$, then $x_1 = 1$ and so $h(i) = 0$, i.e., $\sum_{j \sim i} y_j = w_i$.

Suppose $y_j > 0$. It is clear from the instructions that if $x_i = x_k = 1$, then $T_i \cap T_k = \emptyset$. Thus, $\sum_{i \sim j} x_i \leq 1$. Equality follows from the feasibility of the primal solution. ∎

Farber in fact also gave an algorithm for the weighted independent domination problem with arbitrary real weights for strongly chordal graphs. The approach is similar to that for chordal graphs (with restricted weights) in Section 5.2. The only difference is that the function $g(i)$ is replaced by a set $S_i = \{j : i \sim j \text{ and } y_j > 0\}$ which is similar to T_i in this section. The development is similar to that in Section 5.2 and thus omitted.

6.3 Domatic partition in strongly chordal graphs

Peng and Chang [170] gave an elegant algorithm for the domatic partition problem in strongly chordal graphs.

Their algorithm uses a primal-dual approach. Suppose $[v_1, v_2, \ldots, v_n]$ is a strong elimination ordering of $G = (V, E)$ with the minimum degree $\delta(G)$. Choose a vertex x of degree $\delta(G)$. As any dominating set D_i in a domatic partition of G contains at least one vertex v_i in $N[x]$ and two distinct D_i have different corresponding v_i, it is easy to see the following inequality.

Weak Duality Inequality: $d(G) \leq \delta(G) + 1$.

Their algorithm maintains $\delta(G)+1$ disjoint sets. Initially, these sets are empty. The algorithm scans the vertices in the reverse order of the strong elimination ordering. A vertex is included in a set when it is scanned. When the algorithm terminates, these $\delta(G)+1$ sets are dominating sets.

A vertex v is *completely dominated* if v is dominated by all of these $\delta(G)+1$ dominating sets.

Algorithm DomaticSC. Determine a domatic partition of a strongly chordal graph G of size $\delta(G)+1$.
Input. A strongly chordal graph $G=(V,E)$ with a strong elimination ordering $[v_1, v_2, \ldots, v_n]$.
Output. A partition of V into $\delta(G)+1$ disjoint dominating sets of G.
Method.

$S_i \leftarrow \emptyset$ for $1 \le i \le \delta(G)+1$;
for $i = n$ **to** 1 **step** -1 **do**
find the largest $k \sim i$ such that v_k is not completely dominated;
let S_ℓ be a set that does not dominate v_k;
$S_\ell \leftarrow \{v_i\} \cup S_\ell$;
if no such set exists then include v_i to an arbitrary S_ℓ;
end do.

Before proving the correctness of the algorithm, two lemmas are needed.

Lemma 6.3 *Assume $S_\ell \subseteq \{v_{i+1}, v_{i+2}, \ldots, v_n\}$ and $k \sim i$, where $1 \le i \le n$. If S_ℓ does not dominate v_k, then S_ℓ does not dominate v_j for all $j \le k$ with $j \sim i$.*

Proof. Suppose to the contrary that S_ℓ has a vertex v_p dominating v_j, i.e., $i < p$ and $p \sim j$. Then $i < p$, $j \le k$, $i \sim j$, $i \sim k$ and $p \sim j$ imply $p \sim k$ by (SEO), which contradicts that S_ℓ does not dominate v_k. ∎

Let $r(v) = |\{x \in N[v]\colon x \text{ is not in any of the } \delta(G)+1 \text{ sets}\}|$ and $\mathrm{ndom}(v)$ be the number of sets that do not dominate v during the execution of Algorithm DomaticSC.

Lemma 6.4 *Algorithm DomaticSC maintains the following invariant:*

$$r(v_j) \ge \mathrm{ndom}(v_j) \text{ for all } j \in \{1, 2, \ldots, n\}.$$

Proof. The lemma is proved by induction. Initially,

$$r(v_j) = \deg(v_j) + 1 \ge \delta(G) + 1 = \mathrm{ndom}(v_j)$$

for all $v_j \in V$. During iteration i, only values of $r(v_j)$ and $\mathrm{ndom}(v_j)$, where $j \sim i$, may be altered when v_i is included in a set S_ℓ. Notice that the algorithm determines the largest index $k \sim i$ such that v_k is not completely dominated. It then finds a set S_ℓ that does not dominate v_k (S_ℓ is chosen arbitrarily when v_k does not exist).

For any $j \sim i$ with $j \leq k$, by Lemma 6.3, v_j was not dominated by S_ℓ. Therefore, $r(v_j)$ and $\mathrm{ndom}(v_j)$ are decremented by one after v_i is included in S_ℓ.

On the other hand, for any $j \sim i$ with $j > k$ (or non-existence of such v_k), by the choice of the vertex v_k in the algorithm, vertex v_j is completely dominated, i.e., $\mathrm{ndom}(v_j) = 0$. Thus the invariant is maintained. ∎

Theorem 6.5 *Algorithm DomaticSC partitions the vertex set of a strongly chordal graph $G = (V, E)$ into $d(G) = \delta(G) + 1$ disjoint dominating sets in linear time provided that a strong elimination ordering is given.*

Proof. Upon termination of the algorithm, $r(v_j) = 0$ for all $j \in \{1, 2, \ldots, n\}$. According to Lemma 6.4, $\mathrm{ndom}(v_j) = 0$ for all v_j in V. That is, these $\delta(G) + 1$ sets are dominating sets of G. The strong duality equality

$$d(G) = \delta(G) + 1$$

then follows from the weak duality inequality.

To implement that algorithm efficiently, each vertex v_i is associated with a variable $\mathrm{ndom}(i)$ and an array $L(i)$ of size $\delta(G)+1$. Initially, $\mathrm{ndom}(i) = \delta(G)+1$ and the values of entries in L_i are all zero. Thus, for each vertex it takes $O(d_i)$ time to test $\mathrm{ndom}(i)$ to determine v_k, where d_i is the degree of v_i. It then takes $O(\delta(G) + 1)$ time to decide which set v_i should go. Finally, for each $v_j \in N[v_i]$, it takes $O(1)$ time to update $\mathrm{ndom}(j)$ and L_j. Therefore, the algorithm takes

$$O\left(\sum\nolimits_{i=1}^{n} (d_i + \delta(G) + 1)\right) = O(|V| + |E|) \quad \text{time.}$$ ∎

7 Permutation graphs

Given a permutation $\pi = (\pi(1), \pi(2), \ldots, \pi(n))$ on the set $I_n = \{1, 2, \ldots, n\}$, the *permutation graph* of π is the graph $G(\pi) = (I_n, E(\pi))$ with

$$E(\pi) = \{jk : (j - k)(\pi^{-1}(j) - \pi^{-1}(k)) < 0\}.$$

Note that $\pi^{-1}(j)$ is the position of j in the permutation π. Figure 13 illustrates a permutation and its corresponding permutation graph. If a line between each integer i and its position in π is drown, then n lines are created, each with an associated integer. In this way, two vertices j and k are adjacent in $G(\pi)$ if and only if their corresponding lines cross. That is, $G(\pi)$ is the intersection graph of these n lines. Notice that an independent set in $G(\pi)$ corresponds to an increasing subsequence of π, and a clique in $G(\pi)$ corresponds to a decreasing subsequence of π.

Permutation graphs were first introduced by Pnueli, Lempel and Even in [172, 89]. Since that time quite a few polynomial time algorithms have been

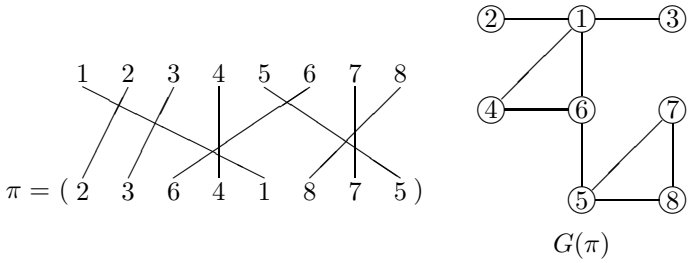


Figure 13: A permutation and its corresponding permutation graph.

constructed on permutation graphs. For example, Atallah, Manacher and Urrutia [9], Brandstädt and Kratsch [32], Farber and Keil [96] and Tsai and Hsu [192] have constructed polynomial domination and independent domination algorithms, Brandstädt and Kratsch [32] and Colbourn and Stewart [80] have constructed polynomial connected domination algorithms, and Brandstädt and Kratsch [32] and Corneil and Stewart [80] have constructed polynomial total domination algorithms.

This section presents a simple $O(n^2)$-time algorithm, due to Brandstädt and Kratsch [33] for finding a minimum weighted independent dominating set in a permutation graph. Assume that the defining permutation π of the permutation graph is given as part of the input. Spinrad [187] has shown that π can be constructed in $O(n^2)$ time, given the graph G.

This algorithm takes the advantage of the observation that a set is an independent dominating set if and only if it is a maximal independent set. Since maximal independent sets in permutation graphs correspond to maximal increasing subsequences in π, all that is necessary is to search for such a sequence in π of minimum weight. In particular, it determines, for every j, $1 \leq j < n$, the minimum weight $\gamma_i(j, w)$ of an independent dominating set in the subsequence $\pi(1), \pi(2), \ldots, \pi(j)$, which contains $\pi(j)$ as the rightmost element. Let $w(j)$ denote the weight of vertex j.

Algorithm WIndDomPer. Solve the weighted independent domination problem for permutation graphs.
Input. A permutation graph G with its corresponding permutation π on the set $\{1, 2, \ldots, n\}$ and vertex weights $w(1), w(2), \ldots, w(n)$ of real numbers.
Output. The weighted independent domination number $\gamma_i(G, w)$ of G.
Method.

for $j = 1$ **to** n **do**
    $p(j) \leftarrow 0$;
    $\gamma_i(j) = w(\pi(j))$;
    end do;
for $k = j - 1$ **to** 1 **step** -1 **do**

if $(\pi(j) > \pi(k)$ **and** $p(j) = 0)$ **then**
$\gamma_i(j) \leftarrow w(\pi(j)) + \gamma_i(k)$;
$p(j) \leftarrow \pi(k)$;
end if;
else if $\pi(j) > \pi(k) > p(j) > 0$ **then**
$\gamma_i(j) \leftarrow \min\{\gamma_i(j), w(\pi(j)) + \gamma_i(k)\}$;
$p(j) \leftarrow \pi(k)$;
end if
end do;
$m \leftarrow p(n)$;
$\gamma_i(G, w) \leftarrow \gamma_i(n)$;
for $j = n - 1$ **to** 1 **step** -1 **do**
if $\pi(j) > m$ **then**
$\gamma_i(G, w) \leftarrow \min\{\gamma_i(G, w), \gamma_i(j)\}$;
$m \leftarrow \pi(j)$;
end if.

The algorithm is illustrated by the permutation graph $G(\pi)$ in Figure 13, where $\pi = (2\,3\,6\,4\,1\,8\,7\,5)$ and all weights are equal to 1: $\gamma_i(j) = (1\,2\,3\,3\,1\,2\,2\,2)$, and thus the minimum size of an independent dominating set is 2, for example the set $\{1, 5\}$.

8 Cocomparability graphs

A graph $G = (V, E)$ is a *comparability graph* if G has an orientation $H = (V, F)$ such that $xy, yz \in F$ imply $xz \in F$. In other words, G has a *comparability ordering* which is an ordering $[v_1, v_2, \ldots, v_n]$ of V satisfying

$$i < j < k \text{ and } v_iv_j, v_jv_k \in E \text{ imply } v_iv_k \in E. \tag{CO}$$

A graph $G = (V, E)$ is a *cocomparability graph* if its complement $\overline{G}$ is a comparability graph, or equivalently, G has a *cocomparability ordering* which is an ordering $[v_1, v_2, \ldots, v_n]$ of V satisfying

$$i < j < k \text{ and } v_iv_k \in E \text{ imply } v_iv_j \in E \text{ or } v_jv_k \in E. \tag{CCO}$$

There is an $O(n^{2.376})$-time recognition algorithm for comparability graphs and thus for cocomparability graphs [187]. This has been improved by McConnell and Spinrad who gave an $O(n + m)$-time algorithm constructing an orientation of any given graph G such that the orientation is a transitive orientation of G if and only if G has a transitive orientation [161]. Unfortunately, the best algorithm for testing whether the orientation is indeed transitive has running time $O(n^{2.376})$.

The class of cocomparability graphs is a well studied superclass of the classes of interval graphs, permutation graphs and trapezoid graphs. Domination problems on cocomparability graphs were considered for the first time by Kratsch

and Stewart [149]. They obtained polynomial time algorithms for the domination/total domination/connected domination and the weighted independent domination problems in cocomparability graphs. These algorithms are designed by dynamic programming using cocomparability orderings. Breu and Kirkpatrick [35] (see [4]) improved this by giving $O(nm^2)$-time algorithms for the domination and the total domination problems and an $O(n^{2.376})$-time algorithm for the weighted independent domination problem in cocomparability graphs.

On the other hand, the weighted domination, total domination and connected domination problems are NP-complete in cocomparability graphs [51]. Also, the problem "Given a cocomparability graph G, does G have a dominating clique?" is NP-complete [149].

An $O(n^3)$-time algorithm computing a minimum cardinality connected dominating set of a connected cocomparability graph has been given in [149]. The following is the algorithm for this problem given by Breu and Kirkpatrick [35] (see also [4]).

Let $[v_1, v_2, \ldots, v_n]$ be a cocomparability ordering of a cocomparability graph $G = (V, E)$. For vertices $u, w \in V$, write $u < w$ if u appears before w in the ordering, i.e., $u = v_i$ and $w = v_j$ implies $i < j$. For $i \leq j$ the set $\{v_k : i \leq k \leq j\}$ is denoted by $[v_i, v_j]$. Then $v_iv_j \in E$ implies that every vertex v_k with $i < k < j$ is adjacent to v_i or to v_j, thus $\{v_i, v_j\}$ dominates $[v_i, v_j]$. This can be generalized as follows: let $S \subseteq V$ where $G[S]$ is connected. Then S dominates $[\min(S), \max(S)]$ where $\min(S)$ (respectively, $\max(S)$) is the vertex of S with the smallest (respectively, largest) index in the ordering.

The following theorem and lemma are given in [149].

Theorem 8.1 *Any connected cocomparability graph G has a minimum connected dominating set S such that the induced subgraph $G[S]$ is a chordless path $p_1, p_2, \ldots, p_k$.*

Lemma 8.2 *Suppose $S \subseteq V$ is a minimum connected dominating set of a cocomparability graph $G = (V, E)$ with a cocomparability ordering $[v_1, v_2, \ldots, v_n]$. If $G[S]$ is a chordless path $p_1, p_2, \ldots, p_k$, then every vertex $x < \min(S)$ is dominated by $\{p_1, p_2\}$ and every vertex $y > \max(S)$ is dominated by $\{p_{k-1}, p_k\}$.*

The following approach enables an elegant way of locating a chordless path of minimum size that dominates the cocomparability graph. A *source vertex* is a vertex v_i such that $v_kv_i \in E$ for all $k < i$ and a *sink vertex* is a vertex v_j such that $v_jv_k \in E$ for all $k > j$. Then $[v_1, v_2, \ldots, v_n]$ is a *canonical cocomparability ordering* if $[v_1, v_2, \ldots, v_r]$, $1 \leq r < n$, are the source vertices and $v_s, v_{s+1}, \ldots, v_n$, $1 < s \leq n$, are the sink vertices. Note that every cocomparability graph G has a canonical cocomparability ordering. Furthermore, given any cocomparability ordering, a canonical one can be computed in time $O(n + m)$.

From now on, assume that $[v_1, v_2, \ldots, v_n]$ is a canonical cocomparability ordering. Since the source vertices of G form a clique, any source vertex v_i

dominates $[v_1, v_i]$. Analogously, since the sink vertices of G form a clique, any sink vertex v_j dominates $[v_j, v_n]$. Therefore the vertex set of every path between a source vertex and a sink vertex is dominating.

The following theorem given in [35] enlights the key property.

Theorem 8.3 *Every connected cocomparability graph $G = (V, E)$ satisfying $\gamma_c(G) \geq 3$ has a minimum connected dominating set which is the vertex set of a shortest path between a source and a sink vertex of G.*

Proof. Let $[v_1, v_2, \ldots, v_n]$ be a canonical cocomparability ordering of G. According to Theorem 8.1, there is a minimum connected dominating set S of G such that $G[S]$ is a chordless path $P : p_1, p_2, \ldots, p_k$, $k \geq 3$. Construct below a chordless path P'' between a source vertex and a sink vertex of G, that has the same number of vertices as the path P.

Let $p_1 = v_i$ and $p_2 = v_j$. First observe that $p_2 = v_j$ cannot be a source vertex, otherwise $N[p_2] \supseteq [v_1, v_j]$ implying that $\{p_2, p_3, \ldots, p_k\}$ is also a connected dominating set of G, a contradiction. If p_1 is a source vertex then P starts at a source vertex. In this case, proceed with the path $P' = P$ (possibly) rearranging p_{k-1}, p_k.

Suppose $p_1 = v_i$ is not a source vertex. Then there is a source vertex u of G with $up_1 \notin E$. Since $[v_1, v_2, \ldots, v_n]$ is a canonical cocomparability ordering and since p_1 and p_2 are not source vertices, $u < p_1$ and $u < p_2$. Since $v_iv_j \in E$ and by Lemma 8.2, $\{v_i, v_j\}$ dominates $[1, \max\{v_i, v_j\}]$. Consequently $up_2 \in E$.

Consider the set $S' = \{u, p_2, \ldots, p_k\}$. Since $\{u, p_2\}$ dominates $[v_1, v_j]$, S' is a dominating set. Since $P : p_1, p_2, \ldots, p_k$ is a chordless path, $t \geq 3$ implies $p_tu \notin E$. Thus S' induces the chordless path $P' : u, p_2, \ldots, p_k$.

Similarly, starting from P' the vertex p_k can be replaced, if necessary. Vertex p_{k-1} is not a sink vertex. If p_k is a sink vertex then $S'' = S'$ and $P'' = P'$. Otherwise, replace p_k by a sink vertex v satisfying $vp_k \notin E$ to obtain S'' and P''.

$G[S'']$ induces a chordless path between a sink and a source vertex. The vertex set of any path between a sink and a source vertex is a dominating set. By construction S'' is a minimum connected dominating set. Consequently S'' is the vertex set of a shortest path between a source vertex and a sink vertex of G. ∎

According to Theorem 8.3, when $\gamma_c(G) \geq 3$, computing a minimum connected dominating set of a connected cocomparability graph G reduces to computing a shortest path between a source and a sink vertex of G.

Algorithm ConDomCC. Solve the connected domination problem in cocomparabiliy graphs.
Input: A connected cocomparability graph $G = (V, E)$ and a canonical cocomparability ordering $[v_1, v_2, \ldots, v_n]$ of G.
Output: A minimum connected dominating set of G.
Methods.

1. Check whether G has a minimum connected dominating set D of size at most 2. If so, output D and stop.
2. Construct a new graph G' by adding two new vertices s and t to G such that s is adjacent exactly to the source vertices of G and t is adjacent exactly to the sink vertices of G.
3. Compute a shortest path $P : s, p_1, p_2, \ldots, p_k, t$ between s and t in G' by the breadth-first-search.
4. Output $\{p_1, p_2, \ldots, p_k\}$.

The correctness of Algorithm ConDomCC follows immediately from Theorem 8.3. The "almost linear" running time of the algorithm follows from the well-known fact that breadth-first-search is a linear-time procedure.

Theorem 8.4 *For any connected cocomparability graph $G = (V, E)$ with a canonical cocomparability ordering $[v_1, v_2, \ldots, v_n]$, Algorithm ConDomCC outputs a minimum connected dominating set of G in $O(nm)$ time. In fact, all parts of the algorithm except checking for a connected dominating set of size two can be done in time $O(n + m)$.*

It is clearly unsatisfactory that the straightforward test for a connected dominating set of size two dominates the overall running time. The crux is that there are even permutation graphs for which each minimum connected dominating set of size two contains neither a source nor a sink vertex (see [129, 143]) It seems that minimum dominating sets of this type cannot be found by a shortest path approach. It is an open question whether Step 1 of Algorithm ConDomCC can be implemented in a more efficient way.

Notice that the $O(n)$-time algorithms computing a minimum connected dominating set for permutation graphs [129] and trapezoid graphs [143] both rely on Theorem 8.3.

Corneil, Olariu and Stewart have done a lot of research on asteroidal triple-free graphs, usually called AT-free graphs [74, 77]. They are defined as those graphs not containing an asteroidal triple, i.e., a set of three vertices such that between any two of the vertices there is a path avoiding the neighborhood of the third.

AT-Free graphs form a superclass of the cocomparability graphs. They are a "large class of graphs" with nice structural properties and some of them are related to domination. One of the major theorems on the structure of AT-free graphs states that every connected AT-free graph has a dominating pair, i.e., a pair of vertices u, v such that the vertex set of each path between u and v is a dominating set.

An $O(n + m)$ algorithm computing a dominating pair for a given connected AT-free graph has been presented in [77]. This can be used to obtain an $O(n+m)$ algorithm computing a dominating path for connected AT-free graphs (see also [75]). An $O(n^3)$ algorithm computing a minimum connected dominating set for

connected AT-free graphs is given in [11]. An $O(n+m)$ algorithm computing a minimum connected dominating set in connected AT-free graphs with diameter greater than three is given in [77].

9 Distance-hereditary graphs

9.1 Hangings of distance-hereditary graphs

A graph is *distance-hereditary* if every two vertices have the same distance in every connected induced subgraph. Distance-hereditary graphs were introduced by Howorka [123]. The characterization and recognition of distance-hereditary graphs have been studied in [12, 83, 84, 107, 123]. Distance-hereditary graphs are parity graphs [37] and include all cographs [70, 79].

The *hanging* h_u of a connected graph $G = (V, E)$ at a vertex $u \in V$ is the collection of sets $L_0(u)$, $L_1(u)$, ..., $L_t(u)$ (or L_0, L_1, ..., L_t if there is no ambiguity), where $t = \max_{v \in V} d_G(u, v)$ and

$$L_i(u) = \{v \in V : d_G(u, v) = i\}$$

for $0 \leq i \leq t$. For any $1 \leq i \leq t$ and any vertex $v \in L_i$, let $N'(v) = N(v) \cap L_{i-1}$. A vertex $v \in L_i$ with $1 \leq i \leq t$ is said to have a *minimal neighborhood* in L_{i-1} if $N'(x)$ is not a proper subset of $N'(v)$ for any $x \in L_i$. Such a vertex v certainly exists.

Theorem 9.1 [12, 83, 84] *A connected graph $G = (V, E)$ is distance-hereditary if and only if for every hanging $h_u = (L_0, L_1, \ldots, L_t)$ of G and every pair of vertices x, $y \in L_i$ $(1 \leq i \leq t)$ that are in the same component of $G - L_{i-1}$, we have $N'(x) = N'(y)$.*

Theorem 9.2 [12] *Suppose $h_u = (L_0, L_1, \ldots, L_t)$ is a hanging of a connected distance-hereditary graph at u. For any two vertices $x, y \in L_i$ with $i \geq 1$, $N'(x) \cap N'(y) = \emptyset$ or $N'(x) \subseteq N'(y)$ or $N'(x) \supseteq N'(y)$.*

Theorem 9.3 (Fact 3.4 in [107]) *Suppose $h_u = (L_0, L_1, \ldots, L_t)$ is a hanging of a connected distance-hereditary graph at u. If vertex $v \in L_i$ with $1 \leq i \leq t$ has a minimal neighborhood in L_{i-1}, then $N_{V \setminus N'(v)}(x) = N_{V \setminus N'(v)}(y)$ for every pair of vertices x and y in $N'(v)$.*

9.2 Weighted connected domination

D'Atri and Moscarini [83] gave $O(|V||E|)$ algorithms for connected domination and Steiner tree problems in distance-hereditary graphs. Brandstädt and Dragan [30] presented a linear-time algorithm for the connected r-domination and Steiner tree problems in distance-hereditary graphs.

This section presents a linear-time algorithm given by Yeh and Chang [202] for finding a minimum weighted connected dominating set of a connected distance-hereditary graph $G = (V, E)$ in which each vertex v has a weight $w(v)$ that is a real number. According to Lemma 2.1, assume that the vertex weights are nonnegative.

Lemma 9.4 *Suppose $h_u = \{L_0, L_1, \ldots, L_t\}$ is a hanging of a connected distance-hereditary graph at u. For any connected dominating set D and $v \in L_i$ with $2 \leq i \leq t$, $D \cap N'(v) \neq \emptyset$.*

Proof. Choose a vertex y in D that dominates v. Then $y \in L_{i-1} \cup L_i \cup L_{i+1}$. If $y \in L_{i-1}$, then $y \in D \cap N'(v)$. So, assume that $y \in L_i \cup L_{i+1}$. Choose a vertex $x \in D \cap (L_0 \cup L_1)$ and an x-y path

$$P : x = v_1, v_2, \ldots, v_m = y$$

using vertices only in D. Let j be the smallest index such that $\{v_j, v_{j+1}, \ldots, v_m\} \subseteq L_i \cup L_{i+1} \cup \ldots \cup L_t$. Then $v_j \in L_i$, $v_{j-1} \in N'(v_j)$, and v and v_j are in the same component of $G - L_{i-1}$. By Theorem 9.1, $N'(v) = N'(v_j)$ and so $v_{j-1} \in D \cap N'(v)$. In any case, $D \cap N'(v) \neq \emptyset$. ∎

Theorem 9.5 *Suppose $G = (V, E)$ is a connected distance-hereditary graph with a non-negative weight function w on its vertices. Let $h_u = \{L_0, L_1, \ldots, L_t\}$ be a hanging at a vertex u of minimum weight. Consider the set $\mathcal{A} = \{N'(v)$: $v \in L_i$ with $2 \leq i \leq t$ and v has a minimal neighborhood in $L_{i-1}\}$. For each $N'(v)$ in $\mathcal{A}$, choose one vertex v^* in $N'(v)$ of minimum weight, and let D be the set of all such v^*. Then D or $D \cup \{u\}$ or some $\{v\}$ with $v \in V$ is a minimum weighted connected dominating set of G.*

Proof. For any $x \in L_i$ with $2 \leq i \leq t$, by Theorem 9.2, $N'(x)$ includes some $N'(v)$ in $\mathcal{A}$. This gives Claim 1.

Claim 1. For any $x \in L_i$ with $2 \leq i \leq t$, $x \in N[L_{i-1} \cap D]$.

Claim 2. $D \cup \{u\}$ is a connected dominating set of G.
Proof of Claim 2. By Claim 1 and $N[u] = L_1 \cup \{u\}$, $D \cup \{u\}$ is a dominating set of G. Also, by Claim 1, for any vertex x in $D \cup \{u\}$ there exists an x-u path using vertices only in $D \cup \{u\}$, i.e., $G[D \cup \{u\}]$ is connected. □

Suppose M is a minimum weighted connected dominating set of G. According to Lemma 9.4, $M \cap N'(v) \neq \emptyset$ for each $N'(v) \in \mathcal{A}$, say $v^{**} \in M \cap N'(v)$. Since any two sets in $\mathcal{A}$ are disjoint, $|M| \geq |\mathcal{A}| = |D|$.

Case 1. $|M| = 1$. The theorem is obvious in this case.

Case 2. $|M| > |D|$. In this case, there is at least one vertex x in M that is not a v^{**}. Then

$$w(M) \geq \sum\nolimits_{v^{**}} w(v^{**}) + w(x) \geq \sum\nolimits_{v^*} w(v^*) + w(u) = w(D \cup \{u\}).$$

This together with Claim 2 gives that $D \cup \{u\}$ is a minimum weighted connected dominating set of G.

Case 3. $|M| = |D| \geq 2$. Since $\mathcal{A}$ contains pairwise disjoint sets, $M = \{v^{**}: N'(v) \in \mathcal{A}\}$. Then $w(M) = \sum_{v^{**}} w(v^{**}) \geq \sum_{v^*} w(v^*) = w(D)$.

For any two vertices x^* and y^* in D, x^{**} and y^{**} are in M. Since $G[M]$ is connected, there is an x^{**}-y^{**} path in $G[M]$:

$$x^{**} = v_0^{**}, v_1^{**}, \ldots, v_n^{**} = y^{**}.$$

For any $1 \leq i \leq n$, since v_i^* and v_i^{**} are both in $N'(v_i) \in \mathcal{A}$, by Theorem 9.3, $N_{V \setminus N'(v_i)}(v_i^*) = N_{V \setminus N'(v_i)}(v_i^{**})$. But $v_{i-1}^{**} \in N_{V \setminus N'(v_i)}(v_i^{**})$. Therefore $v_{i-1}^{**} \in N_{V \setminus N'(v_i)}(v_i^*)$ and $v_i^* \in N_{V \setminus N'(v_{i-1})}(v_{i-1}^{**})$. Also, that v_{i-1}^* and v_{i-1}^{**} are both in $N'(v_{i-1}) \in \mathcal{A}$ implies that $N_{V \setminus N'(v_{i-1})}(v_{i-1}^*) = N_{V \setminus N'(v_{i-1})}(v_{i-1}^{**})$. Then $v_i^* \in N_{V \setminus N'(v_{i-1})}(v_{i-1}^*)$. This proves that v_{i-1}^* is adjacent to v_i^* for $1 \leq i \leq n$ and then

$$x^* = v_0^*, v_1^*, \ldots, v_n^* = y^*$$

is an x^*-y^* path in $G[D]$, i.e., $G[D]$ is connected.

For any x in V, since M is a dominating set, $x \in N[v^{**}]$ for some $N'(v) \in \mathcal{A}$. Note that v^{**} and v^* are both in $N'(v)$. According to Theorem 9.3, $N_{V \setminus N'(v)}(v^{**}) = N_{V \setminus N'(v)}(v^*)$. In the case of $x \notin N'(v)$, $x \in N[v^{**}]$ implies $x \in N[v^*]$, i.e., D dominates x. In the case of $x \in N'(v)$, $N_{V \setminus N'(v)}(v^*) = N_{V \setminus N'(v)}(x)$. Since $G[D]$ is connected and $|D| \geq 2$, v^* is adjacent to some $y^* \in D \setminus N'(v)$. Then x is also adjacent to y^*, i.e., D dominates x. In any case, D is a dominating set. Therefore D is a minimum weighted connected dominating set of G. ∎

Lemma 2.1 and Theorem 9.5 together give an efficient algorithm for the weighted connected domination problem in distance-hereditary graphs as follows. To implement the algorithm efficiently, the set $\mathcal{A}$ is not actually found. Instead, the following step is performed for each $2 \leq i \leq t$. Sort the vertices in L_i such that

$$|N'(x_1)| \leq |N'(x_2)| \leq \ldots \leq |N'(x_j)|.$$

Then process $N'(x_k)$ for k from 1 to j. At iteration k, if $N'(x_k) \cap D = \emptyset$, then $N'(x_k)$ is in $\mathcal{A}$ and choose a vertex of minimum weight to put it into D; otherwise $N'(x_k) \notin \mathcal{A}$ and do nothing.

Algorithm WConDomDH. Find a minimum weighted connected dominating set of a connected distance-hereditary graph.
Input: A connected distance-hereditary graph $G = (V, E)$ and a weight $w(v)$ of real number for each $v \in V$.
Output: A minimum weighted connected dominating set D of graph G.
Method.

$D \leftarrow \emptyset$;
let $V' = \{v \in V : w(v) < 0\}$;

$w(v) \leftarrow 0$ for each $v \in V'$;
let u be a vertex of minimum weight in V;
determine the hanging $h_u = (L_0, L_1, \ldots, L_t)$ of G at u;
for $i = 2$ **to** t **do**
    let $L_i = \{x_1, x_2, \ldots, x_j\}$;
    sort L_i such that $|N'(x_{i_1})| \leq |N'(x_{i_2})| \leq \ldots \leq |N'(x_{i_j})|$;
    for $k = 1$ **to** j **do**
        if $N'(x_{i_k}) \cap D = \emptyset$ **then** $D \leftarrow D \cup \{y\}$ where y is a vertex of minimum weight in $N'(x_{i_k})$;
    end do;
if not ($L_1 \subseteq N[D]$ and $G[D]$ is connected) **then** $D \leftarrow D \cup \{u\}$;
for $v \in V$ that dominates V **do**
    if $w(v) < w(D)$ **then** $D \leftarrow \{v\}$;
$D \leftarrow D \cup V'$.

Theorem 9.6 *Algorithm WConDomDH gives a minimum weighted connected dominating set of a connected distance-hereditary graph in linear time.*

Proof. The correctness of the algorithm follows from Lemma 2.1 and Theorem 9.5. For each i, sort L_i by using a bucket sort. Then the algorithm runs in $O(|V| + |E|)$ time. ∎

References

[1] S. Arnborg. Efficient algorithms for combinatorial problems on graphs with bounded decomposability–a survey. *Bit*, 25:2–23, 1985.

[2] S. Arnborg, S. T. Hedetniemi and A. Proskurowski, editors. *Efficient Algorithms and Partial k-trees, Special Issue Discrete Math.*, volume 54. 1994.

[3] S. Arnborg and A. Proskurowski. Linear time algorithms for NP-hard problems restricted to partial k-trees. *Discrete Appl. Math.*, 23:11–24, 1989. (3.6)

[4] K. Arvind, H. Breu, M. S. Chang, D. G. Kirkpatrick, F. Y. Lee, Y. D. Liang, K. Madhukar, C. Pandu Rangan and A. Srinivasan. Efficient algorithms in cocomparability and trapezoid graphs. Manuscript, 1996. (8)

[5] K. Arvind and C. Pandu Rangan. Efficient algorithms for domination problems on cocomparability graphs. Technical Report TR-TCS-909-18, Indian Institute of Technology, 1990.

[6] K. Arvind and C. Pandu Rangan. Transitive reduction and efficient polylog algorithms on permutation graphs. Submitted, 1996.

[7] T. Asano. Dynamic programming on intervals. *Internat. J. Comput. Geom. Appl.*, 3:323–330, 1993.

[8] M. J. Atallah and S. R. Kosaraju. An efficient algorithm for maxdominance, with applications. *Algorithmica*, 4:221–236, 1989.

[9] M. J. Atallah, G. K. Manacher and J. Urrutia. Finding a minimum independent dominating set in a permutation graph. *Discrete Appl. Math.*, 21:177–183, 1988. (7)

[10] B. S. Baker. Approximation algorithms for NP-complete problems on planar graphs. In *24th Annual Symp. on the Foundations of Computer Science*, pages 265–273, 1983.

[11] H. Balakrishnan, A. Rajaraman and C. Pandu Rangan. Connected domination and Steiner set on asteroidal triple-free graphs. In F. Dehne, J. R. Sack, N. Santoro and S. Whitesides, editors, *Proc. Workshop on Algorithms and Data Structures (WADS'93)*, volume 709, pages 131–141, Montreal, Canada, 1993. Springler-Verlag, Berlin. (8)

[12] H. J. Bandelt and H. M. Mulder. Distance-hereditary graphs. *J. Comb. Theory, Series B*, 41:182–208, 1986. (9.1)

[13] D. W. Bange, A. E. Barkauskas and P. J. Slater. Efficient dominating sets in graphs. In R. D. Ringeisen and F. S. Roberts, editors, *Applications of Discrete Mathematics*, pages 189–199. SIAM, Philadelphia, PA, 1988. (3.3, 5.2)

[14] R. Bar-Yehuda and U. Vishkin. Complexity of finding k-path-free dominating sets in graphs. *Inform. Process. Lett.*, 14:228–232, 1982.

[15] R. E. Bellman and S. E. Dreyfus. *Applied Dynamic Programming*. Princeton University Press, 1962. (3.3)

[16] A. A. Bertossi. Dominating sets for split and bipartite graphs. *Inform. Process. Lett.*, 19:37–40, 1984. (5.2)

[17] A. A. Bertossi. Total domination in interval graphs. *Inform. Process. Lett.*, 23:131–134, 1986. (4.3)

[18] A. A. Bertossi. On the domatic number of interval graphs. *Inform. Process. Lett.*, 28:275–280, 1988. (4.2)

[19] A. A. Bertossi and M. A. Bonuccelli. Some parallel algorithms on interval graphs. *Discrete Appl. Math.*, 16:101–111, 1987.

[20] A. A. Bertossi and A. Gori. Total domination and irredundance in weighted interval graphs. *SIAM J. Discrete Math.*, 1:317–327, 1988. (4.3)

[21] A. A. Bertossi and S. Moretti. Parallel algorithms on circular-arc graphs. *Inform. Process. Lett.*, 33:275–281, 1990.

[22] T. A. Beyer, A. Proskurowski, S. T. Hedetniemi and S. Mitchell. Independent domination in trees. *Congr. Numer.*, 19:321–328, 1977. (3.3)

[23] N. L. Biggs, E. K. Lloyd and R. J. Wilson. *Graph Theory* 1736–1936. Clarendon Press, Oxford, 1986. (1)

[24] J. R. S. Blair, W. Goddard, S. T. Hedetniemi, S. Horton, P. Jones and G. Kubicki. On domination and reinforcement numbers in trees. *Discrete Math.*, 308(7):1165–1175, 2008.

[25] H. L. Bodlaender. Dynamic programming on graphs with bounded treewidth. In T. Lepisto and A. Salomaa, editors, *Lecture Notes in Comput. Sci., Proc. 15th Internat. Colloq. on Automata, Languages and Programming*, volume 317, pages 105–118, Heidelberg, 1988. Springer Verlag.

[26] K. S. Booth and J. H. Johnson. Dominating sets in chordal graphs. *SIAM J. Comput.*, 11:191–199, 1982. (5.2)

[27] S. Booth and S. Lueker. Testing for the consecutive ones property, interval graphs, and graph planarity using PQ-tree algorithms. *J. Comput. Sys. Sci.*, 13:335–379, 1976. (4.1)

[28] A. Brandstädt. The computational complexity of feedback vertex set, Hamiltonian circuit, dominating set, Steiner tree and bandwidth on special perfect graphs. *J. Inform. Process. Cybernet.*, 23:471–477, 1987.

[29] A. Brandstädt, V. D. Chepoi and F. F. Dragan. The algorithmic use of hypertree structure and maximum neighbourhood orderings. In E. W. Mayr, G. Schmidt and G. Tinhofer, editors, *Lecture Notes in Comput. Sci., 20th Internat. Workshop Graph-Theoretic Concepts in Computer Science (WG'94)*, volume 903, pages 65–80, Berlin, 1995. Springer-Verlag.

[30] A. Brandstädt and F. F. Dragan. A linear-time algorithm for connected r-domination and Steiner tree on distance-hereditary graphs. Technical Report SM-DU-261, Univ. Duisburg, 1994. (9.2)

[31] A. Brandstädt, F. F. Dragan, V. D. Chepoi and V. I. Voloshin. Dually chordal graphs. In *Lecture Notes in Comput. Sci., 19th Internat. Workshop Graph-Theoretic Concepts in Computer Science (WG'93)*, volume 790, pages 237–251, Berlin, 1993. Springer-Verlag.

[32] A. Brandstädt and D. Kratsch. On the restriction of some NP-complete graph problems to permutation graphs. In L. Budach, editor, *Lecture Notes in Comput. Sci., Proc. FCT'85*, volume 199, pages 53–62, Berlin, 1985. Springer-Verlag. (7)

[33] A. Brandstädt and D. Kratsch. On domination problems on permutation and other graphs. *Theoret. Comput. Sci.*, 54:181–198, 1987. (7)

[34] B. Bresar, M. A. Henning and D. F. Rall. Rainbow domination in graphs. *Taiwanese J. Math.*, 12(1):213–225, 2008.

[35] H. Breu and D. G. Kirkpatrick. Algorithms for dominating and Steiner set problems in cocomparability. Manuscript, 1996. (8)

[36] M. W. Broin and T. J. Lowe. A dynamic programming algorithm for covering problems with greedy totally balanced contraint matrices. *SIAM J. Algeb. Discrete Methods*, 7:348-357, 1986. (6.2)

[37] M. Burlet and J. P. Uhry. Parity graphs. *Annals of Discrete Math.*, 21:253–277, 1984. (9.1)

[38] D. I. Carson. *Computational Aspects of Some Generalized Domination Parameters*. PhD thesis, Univ. Natal, 1995.

[39] D. I. Carson and O. R. Oellermann. Linear time algorithms for capacity-domination in trees and series parallel graphs. Submitted, 1997.

[40] G. J. Chang. Labeling algorithms for domination problems in sun-free chordal graphs. *Discrete Appl. Math.*, 22:21–34, 1988/89. (6.2)

[41] G. J. Chang. Total domination in block graphs. *Oper. Res. Lett.*, 8:53–57, 1989. (3.6)

[42] G. J. Chang. The domatic number problem. *Discrete Math.*, 125:115–122, 1994.

[43] G. J. Chang. The weighted independent domination problem is NP-complete for chordal graphs. *Discrete Appl. Math.*, 143:351–352, 2004. (5.3)

[44] G. J. Chang and G. L. Nemhauser. The k-domination and k-stability problems on graphs. Technical Report TR-540, School Oper. Res. Industrial Eng., Cornell Univ., 1982.

[45] G. J. Chang and G. L. Nemhauser. R-domination of block graphs. *Oper. Res. Lett.*, 1:214–218, 1982. (3.6)

[46] G. J. Chang and G. L. Nemhauser. The k-domination and k-stability on sun-free chordal graphs. *SIAM J. Algebraic Discrete Methods*, 5:332–345, 1984. (6.2)

[47] G. J. Chang and G. L. Nemhauser. Covering, packing and generalized perfection. *SIAM J. Algeb. Discrete Methods*, 6:109–132, 1985. (6.2)

[48] G. J. Chang, C. Pandu Rangan and S. R. Coorg. Weighted independent perfect domination on cocomparability graphs. *Discrete Appl. Math.*, 63:215–222, 1995.

[49] G. J. Chang, J. Wu and X. Zhu. Rainbow domination on trees. *Discrete Appl. Math.*, 158(1):8–12, 2010. (3.2)

[50] M.-S. Chang. Efficient algorithms for the domination problems on interval graphs and circular-arc graphs. In *IFIP Transactions A-12, Proc. IFIP 12th World Congress*, volume 1, pages 402–408, 1992. (4.3)

[51] M.-S. Chang. Weighted domination on cocomparability graphs. In *Lecture Notes in Comput. Sci., Proc. ISAAC'95*, volume 1004, pages 121–131, Berlin, 1995. Springer-Verlag. (8)

[52] M.-S. Chang, F.-H. Hsing and S.-L. Peng. Irredundance in weighted interval graphs. In *Proc. National Computer Symp.*, pages 128–137, Taipei, Taiwan, 1993. (4.3)

[53] M.-S. Chang and C.-C. Hsu. On minimum intersection of two minimum dominating sets of interval graphs. *Discrete Appl. Math.*, 78(1-3):41–50, 1997.

[54] M.-S. Chang and Y.-C. Liu. Polynomial algorithms for the weighted perfect domination problems on chordal and split graphs. *Inform. Process. Lett.*, 48:205–210, 1993.

[55] M.-S. Chang and Y.-C. Liu. Polynomial algorithms for weighted perfect domination problems on interval and circular-arc graphs. *J. Inform. Sci. Engineering*, 10:549–568, 1994. (4.3)

[56] M.-S. Chang, S. Wu, G. J. Chang and H.-G. Yeh. Domination in distance-hereditary graphs. *Discrete Appl. Math.*, 116(1-2):103–113, 2002.

[57] G. A. Cheston, G. H. Fricke, S. T. Hedetniemi and D. P. Jacobs. On the computational complexity of upper fractional domination. *Discrete Appl. Math.*, 27:195–207, 1990. (3.4)

[58] E. Cockayne, S. Goodman and S. Hedetniemi. A linear algorithm for the domination number of a tree. *Inform. Process. Lett.*, 4(2):41–44, 1975. (1, 3.2)

[59] E. J. Cockayne and S. T. Hedetniemi. A linear algorithm for the maximum weight of an independent set in a tree. In *Proc. Seventh Southeastern Conf. on Combinatorics, Graph Theory and Computing*, pages 217–228, Winnipeg, 1976. Utilitas Math.

[60] E. J. Cockayne, G. MacGillivray and C. M. Mynhardt. A linear algorithm for 0-1 universal minimal dominating functions of trees. *J. Combin. Math. Combin. Comput.*, 10:23–31, 1991.

[61] E. J. Cockayne and F. D. K. Roberts. Computation of minimum independent dominating sets in graphs. Technical Report DM-76-IR, Dept. Math., Univ. Victoria, 1974.

[62] E. J. Cockayne and F. D. K. Roberts. Computation of dominating partitions. *Infor.*, 15:94–106, 1977.

[63] C. J. Colbourn, J. M. Keil and L. K. Stewart. Finding minimum dominating cycles in permutation graphs. *Oper. Res. Lett.*, 4:13–17, 1985.

[64] C. J. Colbourn and L. K. Stewart. Permutation graphs: connected domination and Steiner trees. *Discrete Math.*, 86:179–189, 1990. (7)

[65] C. Cooper and M. Zito. An analysis of the size of the minimum dominating sets in random recursive trees, using the Cockayne-Goodman-Hedetniemi algorithm *Disctere Appl. Math.*, 157(9):2010–2014, 2009.

[66] D. G. Corneil. Lexicographic breadth first search–a survey. *Lecture Notes Comput. Sci.*, 3353:1–19, 2004. (*4.1)

[67] D. G. Corneil. A simple 3-sweep LBFS algorithm for the recognition of unit interval graphs. *Discrete Appl. Math.*, 138:371–379, 2004. (*4.1)

[68] D. G. Corneil and J. M. Keil. A dynamic programming approach to the dominating set problem on k-trees. *SIAM J. Algeb. Discrete Methods*, 8:535–543, 1987. (3.6, 5.2)

[69] D. G. Corneil, E. Kohler and J-M. Lanlignel. On end-vertices of Lexicographic Breadth First Search. *Discrete Appl. Math.*, 158:434–443, 2010. (*4.1)

[70] D. G. Corneil, H. Lerchs and L. Stewart. Complement reducible graphs. *Discrete Appl. Math.*, 3:163–174, 1981. (9.1)

[71] D. G. Corneil, S. Olariu and L. Stewart. The ultimate interval graph recognition algorithm? (extended abstract). in: *Proc. Ninth Annual ACM-SIAM Symp. Discrete Alg.*, ACM, New York, SIAM, Philadelphia 1998, pp. 175–180. (*4.1)

[72] D. G. Corneil, S. Olariu and L. Stewart. The LBFS structure and recognition of interval graphs. *SIAM J. Discrete Math.*, 23:1905–1953, 2009. (4.1)

[73] D. G. Corneil, S. Olariu and L. Stewart. A linear time algorithm to compute a dominating path in an AT-free graph. *Inform. Process. Lett.*, 54:253–258, 1995. (8)

[74] D. G. Corneil, S. Olariu and L. Stewart. Asteroidal triple-free graphs. *SIAM J. Discrete Math.*, 790:211–224, 1994. (8)

[75] D. G. Corneil, S. Olariu and L. Stewart. Computing a dominating pair in an asteroidal triple-free graph in linear time. In *Proc. 4th Algorithms and Data Structures Workshop, LNCS 955*, volume 955, pages 358–368. Springer, 1995.

[76] D. G. Corneil, S. Olariu and L. Stewart. A linear time algorithm to compute dominating pairs in asteroidal triple-free graphs. In *Lecture Notes in Comput. Sci., Proc. 22nd Internat. Colloq. on Automata, Languages and Programming (ICALP'95)*, volume 994, pages 292–302, Berlin, 1995. Springer-Verlag.

[77] D. G. Corneil, S. Olariu and L. Stewart. Linear time algorithms for dominating pairs in asteroidal triple-free graphs. *SIAM J. Comput.*, 944:292–302, 1995. (8)

[78] D. G. Corneil and Y. Perl. Clustering and domination in perfect graphs. *Discrete Appl. Math.*, 9:27–39, 1984. (5.2)

[79] D. G. Corneil, Y. Perl and L. Stewart Burlingham. A linear recognition algorithm for cographs. *SIAM J. Comput.*, 14:926–934, 1985. (9.1)

[80] D. G. Corneil and L. K. Stewart. Dominating sets in perfect graphs. *Discrete Math.*, 86:145–164, 1990. (7)

[81] J. Dabney, B. C. Dean and S. T. Hedetniemi. A Linear-Time algorithm for broadcast domination in a tree *Networks*, 53(2):160–169, 2009.

[82] P. Damaschke, H. Müller and D. Kratsch. Domination in convex and chordal bipartite graphs. *Inform. Process. Lett.*, 36:231–236, 1990. (5.2)

[83] A. D'Atri and M. Moscarini. Distance-hereditary graphs, Steiner trees, and connected domination. *SIAM J. Comput.*, 17:521–538, 1988. (9.1, 9.2)

[84] D. P. Day, O. R. Oellermann and H. C. Swart. Steiner distance-hereditary graphs. *SIAM J. Discrete Math.* 7:437–442, 1994. (9.1)

[85] C. F. De Jaenisch. *Applications de l'Analyse mathematique au Jeu des Echecs.* Petrograd, 1862. (1)

[86] G. S. Domke, J. H. Hattingh, S. T. Hedetniemi, R. C. Laskar and L. R. Markus. Restrained domination in graphs. *Discrete Math.*, 203(1-3):61–69, 1999.

[87] S. E. Dreyfus and A. M. Law. *The Art and Theory of Dynamic Programming.* Academic Press, New York, 1977. (3.3)

[88] J. E. Dunbar, W. Goddard, S. Hedetniemi, A. A. McRae. and M. A. Henning. The algorithmic complexity of minus domination in graphs. *Discrete Appl. Math.*, 68(1-2):73–84, 1996.

[89] S. Even, A. Pnueli and A. Lempel. Permutation graphs and transitive graphs. *J. Assoc. Comput. Mach.*, 19(3):400–410, 1972. (7)

[90] L. Euler. Solutio problematis ad geometriam situs pertinentis. *Acad. Sci. Imp. Petropol.*, 8:128–140, 1736. (1)

[91] M. Farber. *Applications of Linear Programming Duality to Problems Involving Independence and Domination.* PhD thesis, Simon Fraser Univ., 1981.

[92] M. Farber. Domination and duality in weighted trees. *Congr. Numer.*, 33:3–13, 1981. (3.4)

[93] M. Farber. Independent domination in chordal graphs. *Oper. Res. Lett.*, 1:134–138, 1982. (5.3)

[94] M. Farber. Characterizations of strongly chordal graphs. *Discrete Math.*, 43:173–189, 1983.

[95] M. Farber. Domination, independent domination and duality in strongly chordal graphs. *Discrete Appl. Math.*, 7:115–130, 1984. (6.1, 6.2)

[96] M. Farber and J. M. Keil. Domination in permutation graphs. *J. Algorithms*, 6:309–321, 1985. (7)

[97] A. M. Farley, S. T. Hedetniemi and A. Proskurowski. Partitioning trees: matching, domination and maximum diameter. *Internat. J. Comput. Inform. Sci.*, 10:55–61, 1981. (3.5)

[98] M. R. Fellows and M. N. Hoover. Perfect domination. *Australas. J. Combin.*, 3:141–150, 1991. (3.3, 5.2)

[99] C. M. H. de Figueiredo, J. Meidanis and C. P. de Mello. A linear-time algorithm for proper interval graph recognition. *Inform. Process. Lett.*, 56:179–184, 1995. (*4.1)

[100] G. H. Fricke, M. A. Henning, O. R. Oellermann and H. C. Swart. An efficient algorithm to compute the sum of two distance domination parameters. *Discrete Appl. Math.*, 68:85–91, 1996.

[101] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman, New York, 1979. (1, 5.1, 5.2)

[102] F. Gavril. Algorithms for minimum colorings, maximum clique, minimum coverings by cliques and maximum independent set of a chordal graph. *SIAM J. Comput.*, 1:180–187, 1972.

[103] M. C. Golumbic. *Algorithmic Graph Theory and Perfect Graphs*. Academic Press, New York, 1980. (2.3, 5.1)

[104] M. C. Golumbic. Algorithmic aspect of perfect graphs. *Annals of Discrete Math.*, 21:301–323, 1984. (5.1)

[105] J. Guo, R. Niedermeier and D. Raible. Improved algorithms and complexity results for power domination in graphs. *Algorithmica*, 52(2):177–202, 2008. (3.5)

[106] M. Habib, R. McConnell, C. Paul and L. Viennot. Lex-BFS and partition refinement, with applications to transitive orientation, interval graph recognition and consecutive ones testing. *Theor. Comput. Sci.*, 234:, 59–84, 2000. (4.1)

[107] P. H. Hammer and F. Maffray. Completely separable graphs. *Discrete Appl. Math.*, 27:85–99, 1990. (9.1)

[108] E. O. Hare and S. T. Hedetniemi. A linear algorithm for computing the knight's domination number of a $K \times N$ chessboard. *Congr. Numer.*, 59:115–130, 1987.

[109] J. H. Hattingh and M. A. Henning. The complexity of upper distance irredundance. *Congr. Numer.*, 91:107–115, 1992.

[110] J. H. Hattingh, M. A. Henning and P. J. Slater. On the algorithmic complexity of signed domination in graphs. *Australas. J. Combin.*, 12, 1995. 101–112.

[111] J. H. Hattingh, M. A. Henning and J. L. Walters. On the computational complexity of upper distance fractional domination. *Australas. J. Combin.*, 7:133–144, 1993.

[112] J. H. Hattingh and R. C. Laskar. On weak domination in graphs. *Ars Combin.* 49:205–216, 1998.

[113] T. W. Haynes, S. M. Hedetniemi, S. T. Hedetniemi and M. A. Henning. Domination in graphs applied to electric power networks. *SIAM J. Disc. Math.*, 15(4):519–529, 2002. (3.5)

[114] T. W. Haynes, S. T. Hedetniemi and P. J. Slater. *Fundamentals of Domination in Graphs*. Marcel Dekker, Inc., New York, 1997. (1)

[115] T. W. Haynes, S. T. Hedetniemi and P. J. Slater, editors. *Domination in Graphs: Advanced Topics*. Marcel Dekker, Inc., New York, 1997.

[116] S. M. Hedetniemi, S. T. Hedetniemi and D. P. Jacobs. Private domination: theory and algorithms. *Congr. Numer.*, 79:147–157, 1990. (3.3, 5.2)

[117] S. M. Hedetniemi, S. T. Hedetniemi and R. C. Laskar. Domination in trees: models and algorithms. In Y. Alavi, G. Chartrand, L. Lesniak, D. R. Lick and C. E. Wall, editors, *Graph Theory with Applications to Algorithms and Computer Science*, pages 423–442. Wiley, New York, 1985.

[118] S. M. Hedetniemi, A. A. McRae and D. A. Parks. Complexity results. In T. W. Haynes, S. T. Hedetniemi and P. J. Slater, editors, *Domination in Graphs: Advanced Topics*, Chapter 9. Marcel Dekker, Inc., 1997.

[119] S. T. Hedetniemi and R. C. Laskar. Bibliography on domination in graphs and some basic definitions of domination parameters. *Discrete Math.*, 86:257–277, 1990.

[120] S. T. Hedetniemi, R. C. Laskar and J. Pfaff. A linear algorithm for finding a minimum dominating set in a cactus. *Discrete Appl. Math.*, 13:287–292, 1986. (3.6)

[121] P. Hell and J. Huang. Certifying LexBFS recognition algorithms for proper interval graphs and proper interval bigraphs. *SIAM J. Discrete Math.* 18:55–570, 2005. (*4.1)

[122] A. J. Hoffman, A. W. J. Kolen and M. Sakarovitch. Totally-balanced and greedy matrices. *SIAM J. Algebraic and Discrete Methods*, 6:721–730, 1985. (6.1, 6.2)

[123] E. Howorka. A characterization of distance-hereditary graphs. *Quart. J. Math. Oxford Ser.* 2, 28:417–420, 1977. (9.1)

[124] W. Hsu. The distance-domination numbers of trees. *Oper. Res. Lett.*, 1:96–100, 1982. (3.3)

[125] W.-L. Hsu. A simple test for interval graphs. *Lecture Notes Comput. Sci.*, 657:11–16, 1993. (4.1)

[126] W.-L. Hsu and T.-H. Ma. Fast and simple algorithms for recognizing chordal comparability graphs and interval graphs. *SIAM J. Comput.*, 28:1004–1020, 1999. (4.1)

[127] W.-L. Hsu and R. M. McConnell. PC trees and circular-ones arrangements. *Theor. Comput. Sci.*, 296:59–74, 2003. (4.1)

[128] S. F. Hwang and G. J. Chang. The k-neighbor domination problem in block graphs. *European J. Oper. Res.*, 52:373–377, 1991. (3.6, 5.2)

[129] O. H. Ibarra and Q. Zheng. Some efficient algorithms for permutation graphs. *J. Algorithms*, 16:453–469, 1994. (8)

[130] R. W. Irving. On approximating the minimum independent dominating set. *Inform. Process. Lett.*, 37(4):197–200, 1991.

[131] M. S. Jacobson and K. Peters. Complexity questions for n-domination and related parameters. *Congr. Numer.*, 68:7–22, 1989.

[132] T. S. Jayaram, G. Sri Karishna and C. Pandu Rangan. A unified approach to solving domination problems on block graphs. Report TR-TCS-90-09, Dept. of Computer Science and Eng., Indian Inst. of Technology, 1990. (3.6)

[133] D. S. Johnson. The NP-completeness column: an ongoing guide. *J. Algorithms*, 5:147–160, 1984. (5.2)

[134] D. S. Johnson. The NP-completeness column: an ongoing guide. *J. Algorithms*, 6:291–305,434–451, 1985. (5.2)

[135] L. Kang, M. Y. Sohn and T. C. E. Cheng. Paired-domination in inflated graphs. *Theorect. Comput. Sci.*, 320(2-3):485–494, 2004.

[136] O. Kariv and S. L. Hakimi. An algorithmic approach to network location problems I: the p-centers. *SIAM J. Appl. Math.*, 37:513–538, 1979.

[137] O. Kariv and S. L. Hakimi. An algorithmic approach to network location problems II: the p-medians. *SIAM J. Appl. Math.*, 37:539–560, 1979.

[138] J. M. Keil. Total domination in interval graphs. *Inform. Process. Lett.*, 22:171–174, 1986.

[139] J. M. Keil. The complexity of domination problems in circle graphs. *Discrete Appl. Math.*, 42:51–63, 1993.

[140] J. M. Keil and D. Schaefer. An optimal algorithm for finding dominating cycles in circular-arc graphs. *Discrete Appl. Math.*, 36:25–34, 1992.

[141] T. Kikuno, N. Yoshida and Y. Kakuda. The NP-completeness of the dominating set problem in cubic planar graphs. *Trans. IEEE*, pages 443–444, 1980.

[142] T. Kikuno, N. Yoshida and Y. Kakuda. A linear algorithm for the domination number of a series-parallel graph. *Discrete Appl. Math.*, 5:299–311, 1983.

[143] E. Köhler. Connected domination on trapezoid graphs in $O(n)$ time. Manuscript, 1996. (8)

[144] A. Kolen. Solving covering problems and the uncapacitated plant location problem on trees. *Eropean J. Oper. Res.*, 12:266-278, 1983 (3.4)

[145] N. Korte and R. H. Mohring. An incremental linear-time algorithm for recognizing interval graphs. *SIAM J. Comput.*, 18:68–81, 1989. (4.1)

[146] D. Kratsch. Finding dominating cliques efficiently, in strongly chordal graphs and undirected path graphs. *Discrete Math.*, 86:225–238, 1990. (6.2)

[147] D. Kratsch. Algorithms. In T. W. Haynes, S. T. Hedetniemi and P. J. Slater, editors, *Domination in Graphs: Advanced Topics*, chapter 8. Marcel Dekker, Inc., 1997.

[148] D. Kratsch, R. M. McConnel, K. Mehlhorn and J. P. Spinrad. Certifying algorithms for recognizing interval graphs and permutation graphs. *SIAM J. Discrete Math.*, 36:326–353, 2006. (*4.1)

[149] D. Kratsch and L. Stewart. Domination on cocomparability graphs. *SIAM J. Discrete Math.*, 6(3):400–417, 1993. (8)

[150] R. Laskar, J. Pfaff, S. M. Hedetniemi and S. T. Hedetniemi. On the algorithmic complexity of total domination. *SIAM J. Algebraic Discrete Methods*, 5(3):420–425, 1984. (3.2)

[151] E. L. Lawler and P. J. Slater. A linear time algorithm for finding an optimal dominating subforest of a tree. In *Graph Theory with Applications to Algorithms and Computer Science*, pages 501–506. Wiley, New York, 1985.

[152] Y. D. Liang, C. Rhee, S. K. Dall and S. Lakshmivarahan. A new approach for the domination problem on permutation graphs. *Inform. Process. Lett.*, 37:219–224, 1991.

[153] C.-S. Liao and G. J. Chang. Algorithmic aspect of k-tuple domination in graphs. *Taiwanese J. Math.*, 6(3):415-420, 2002.

[154] M. Livingston and Q. F. Stout. Constant time computation of minimum dominating sets. *Congr. Numer.*, 105:116–128, 1994.

[155] P. J. Looges and S. Olariu. Optimal greedy algorithms for indifference graphs. *Comput. Math. Applica.*, 25:15–25, 1993. (*4.1)

[156] E. Loukakis. Two algorithms for determining a minimum independent dominating set. *Internat. J. Comput. Math.*, 15:213–229, 1984.

[157] T. L. Lu, P. H. Ho and G. J. Chang. The domatic number problem in interval graphs. *SIAM J. Discrete Math.*, 3:531–536, 1990. (4.2)

[158] K. L. Ma and C. W. H. Lam. Partition algorithm for the dominating set problem. *Congr. Numer.*, 81:69–80, 1991.

[159] G. K. Manacher and T. A. Mankus. Finding a domatic partition of an interval graph in time $O(n)$. *SIAM J. Discrete Math.*, 9:167–172, 1996. (4.2)

[160] M. V. Marathe, H. B. Hunt III and S. S. Ravi. Efficient approximation algorithms for domatic partition and on-line coloring of circular arc graphs. *Discrete Appl. Math.*, 64:135–149, 1996.

[161] R. M. McConnell and J. P. Spinrad. Modular decomposition and transitive orientation. Manuscript, 1995. (8)

[162] A. A. McRae and S. T. Hedetniemi. Finding n-independent dominating sets. *Congr. Numer.*, 85:235–244, 1991.

[163] S. L. Mitchell and S. T. Hedetniemi. Edge domination in trees. *Congr. Numer.*, 19:489–509, 1977. (3.2)

[164] S. L. Mitchell and S. T. Hedetniemi. Independent domination in trees. *Congr. Numer.*, 29:639–656, 1979.

[165] S. L. Mitchell, S. T. Hedetniemi and S. Goodman. Some linear algorithms on trees. *Congr. Numer.*, 14:467–483, 1975.

[166] H. Müller and A. Brandstädt. The NP-completeness of STEINER TREE and DOMINATING SET for chordal bipartite graphs. *Theoret. Comput. Sci.*, 53:257–265, 1987. (5.2)

[167] K. S. Natarajan and L. J. White. Optimum domination in weighted trees. *Inform. Process. Lett.*, 7:261–265, 1978. (3.3)

[168] G. L. Nemhauser. *Introduction to Dynamic Programming.* John Wiley & Sons, 1966. (3.3)

[169] S. L. Peng and M. S. Chang. A new approach for domatic number problem on interval graphs. *Proc. National Comp. Symp. R. O. C.*, pages 236–241, 1991. (4.2)

[170] S. L. Peng and M. S. Chang. A simple linear time algorithm for the domatic partition problem on strongly chordal graphs. *Inform. Process. Lett.*, 43:297–300, 1992. (4.2, 6.3)

[171] J. Pfaff, R. Laskar and S. T. Hedetniemi. Linear algorithms for independent domination and total domination in series-parallel graphs. *Congr. Numer.*, 45:71–82, 1984.

[172] A. Pnueli, A. Lempel and S. Even. Transitive orientation of graphs and identification of permutation graphs. *Canad. J. Math.*, 23:160–175, 1971. (7)

[173] A. Proskurowski. Minimum dominating cycles in 2-trees. *Internat. J. Comput. Inform. Sci.*, 8:405–417, 1979. (3.6)

[174] S. Rajbaum. Improved tree decomposition based algorithms for domination-like problems. *Lecture Notes Compy. Sci.*, 2286:613-627, 2002.

[175] G. Ramalingam and C. Pandu Rangan. Total domination in interval graphs revisited. *Inform. Process. Lett.*, 27:17–21, 1988. (4.3)

[176] G. Ramalingam and C. Pandu Rangan. A unified approach to domination problems in interval graphs. *Inform. Process. Lett.*, 27:271–274, 1988. (4.1, 4.3–4.6)

[177] A. Raychaudhuri. On powers of interval and unit interval graphs. *Cong. Numer.*, 59:235–242, 1987. (*4.1)

[178] F. S. Roberts. On the compatibility between a graph and a simple order. *J. Combin. Theory B*, 11:28–38, 1971. (*4.1)

[179] J. S. Rohl. A faster lexicographic N queens algorithm. *Inform. Process. Lett.*, 17:231–233, 1983.

[180] D. J. Rose, R. E. Tarjan and G. S. Lueker. Algorithmic aspects of vertex elimination on graphs. *SIAM J. Discrete Math.*, 5:266–283, 1976. (*4.1)

[181] P. Scheffler. Linear-time algorithms for NP-complete problems restricted to partial k-trees. Technical Report 03/87, IMATH, Berlin, 1987. (3.6)

[182] W.-K. Shih and W.-L. Hsu. A new planarity test. *Theor. Comput. Sci.*, 223:179–191, 1999. (4.1)

[183] K. Simon. A new simple linear algorithm to recognize interval graphs. *Lecture Notes Comput. Sci.*, 553:289–308, 1991. (*4.1)

[184] P. J. Slater. R-domination in graphs. *J. Assoc. Comput. Mach.*, 23:446–450, 1976. (3.2)

[185] P. J. Slater. Domination and location in acyclic graphs. *Networks*, 17:55–64, 1987. (3.3)

[186] R. Sosic and J. Gu. A polynomial time algorithm for the N-queens problem. *SIGART Bull.*, 2(2):7–11, 1990.

[187] J. Spinrad. On comparability and permutation graphs. *SIAM J. Comput.*, 14:658–670, 1985. (7, 8)

[188] A. Srinivasa Rao and C. Pandu Rangan. Linear algorithm for domatic number problem on interval graphs. *Inform. Process. Lett.*, 33:29–33, 1989/90. (4.2)

[189] A. Srinivasan Rao and C. Pandu Rangan. Efficient algorithms for the minimum weighted dominating clique problem on permutation graphs. *Theoret. Comput. Sci.*, 91:1–21, 1991.

[190] J. A. Telle and A. Proskurowski. Efficient sets in partial k-trees. *Discrete Appl. Math.*, 44:109–117, 1993. (3.6)

[191] J. A. Telle and A. Proskurowski. Practical algorithms on partial k-trees with an application to domination-type problems. In F. Dehne, J. R. Sack, N. Santoro and S. Whitesides, editors, *Lecture Notes in Comput. Sci., Proc. Third Workshop on Algorithms and Data Structures (WADS'93)*, volume 703, pages 610–621, Montréal, 1993. Springer-Verlag. (3.6)

[192] K. H. Tsai and W. L. Hsu. Fast algorithms for the dominating set problem on permutation graphs. *Algorithmica*, 9:109–117, 1993. (7)

[193] C. Tsouros and M. Satratzemi. Tree search algorithms for the dominating vertex set problem. *Internat. J. Computer Math.*, 47:127–133, 1993.

[194] D. B. West. *Introduction to Graph Theory, 2nd Edition*. Prentice Hall, 2001. (3.1)

[195] K. White, M. Farber and W. Pulleyblank. Steiner trees, connected domination and strongly chordal graphs. *Networks*, 15:109–124, 1985. (6.2)

[196] T. V. Wimer. Linear algorithms for the dominating cycle problems in series-parallel graphs, partial k-trees and Halin graphs. *Congr. Numer.*, 56:289–298, 1986.

[197] T. V. Wimer. An $O(n)$ algorithm for domination in k-chordal graphs. Manuscript, 1986.

[198] M. Yannakakis and F. Gavril. Edge dominating sets in graphs. *SIAM J. Appl. Math.*, 38(3):364–372, 1980. (3.2)

[199] H. G. Yeh. *Distance-Hereditary Graphs: Combinatorial Structures and Algorithms*. PhD thesis, Univ. Taiwan, 1997.

[200] H. G. Yeh and G. J. Chang. Algorithmic aspect of majority domination. *Taiwanese J. Math.*, 1:343–350, 1997.

[201] H. G. Yeh and G. J. Chang. Linear-time algorithms for bipartite distance-hereditary graphs. Submitted.

[202] H. G. Yeh and G. J. Chang. Weighted connected domination and Steiner trees in distance-hereditary graphs. *Discrete Appl. Math.*, 87:245–253, 1998. (9.2)

[203] C. Yen and R. C. T. Lee. The weighted perfect domination problem. *Inform. Process. Lett.*, 35(6):295–299, 1990. (3.3, 5.2)

[204] C. Yen and R. C. T. Lee. The weighted perfect domination problem and its variants. *Discrete Appl. Math.*, 66:147–160, 1996. (3.6, 5.2)

[205] W. C.-K. Yen. The bottleneck independent domination on the classes of bipartite graphs and block graphs. *Inform. Sci.*, 157:199–215, 2003.

[206] W. C.-K. Yen. Bottleneck domination and bottleneck independent domination on graphs. *J. Inform. Sci. Engin.*, 18(2):311–331, 2002.