Acknowledgements

I sincerely thank Dr. C. Pandu Rangan for having introduced me to the field of algorithms and for having suggested the various problems I have worked on. I am deeply indebted to him for his enthusiastic help, guidance and valuable advice, all of which made him much more than a project guide.

Anand has been a wonderful friend and research collaborator – I don't think I will ever forget the several hours (every day) we spent discussing various things (not just algorithms!), as well as the several days (every month) we spent trying to refine our half-baked solutions to various problems – solutions where counter-examples were more the rule than the exception!

All my classmates and several "Alakites" made my stay here an enjoyable and unforgettable one. Special thanks are due to Bobby, Cheeku, Has, Jyer, Kat and Veena for interesting discussions on numerous varied topics, and the help many of them gave me at various times. Kamakoti, Jayanth, Vamsi, Easwar and Kishore were very nice guys to work with in the TCS lab.

Finally, I thank my parents and my sister Hamsa (who I'm sure will be thrilled to see her name in print!) for all the support and encouragement they've given me.

Abstract

One way of overcoming the intractable nature of problems in graphs is by considering them on special classes of graphs. In this thesis, we study the properties of two important classes of graphs – asteroidal triple-free graphs and distance-hereditary graphs, and use their properties to design efficient polynomial-time algorithms for a variety of problems that are NP-complete on general graphs.

Three independent vertices in a graph are said to constitute an *asteroidal triple (AT)* if there is a path between every two of them that avoids the neighbourhood of the third. Graphs which do not have an AT are said to be AT-free. These graphs strictly contain the well-known class of cocomparability graphs, but are not necessarily perfect. We present $O(n^3)$ algorithms for the problems of finding a minimum cardinality connected dominating set and a Steiner set on these graphs. These are the first known algorithms for these problems on any non-trivial class of non-perfect graphs – in addition, these are the first known algorithms on AT-free graphs for any problem.

Distance-hereditary graphs are an important class of perfect graphs in which all induced paths between any pair of vertices are isometric. We present the following algorithms on these graphs : an $O(m+n)$ algorithm for the minimum weighted connected dominating set problem (improving the existing best result of $O(n^3)$), an $O(m+n)$ algorithm for the minimum weighted clique domination problem, an $O(mn)$ algorithm for the Steiner problem, and an $O(n^3)$ algorithm for the maximum clique problem. A unified approach based on the properties of separators has been used to solve all these problems – to enable this, we have also designed and presented an $O(m+n)$ algorithm to find all the minimal separators of a distance-hereditary graph.

Contents

Chapter 1

Algorithmic and Graph-theoretic Background

1.1 Algorithms and Complexity

An area of Computer Science that has enjoyed a very rapid growth in the past 25 years is the field of *algorithms and complexity theory* – in particular, the design and analysis of computer algorithms. The goals of research in this area are to study the nature of problems that are solvable on a digital computer, to provide solutions to solvable problems, as well as to classify problems into categories depending on their relative degree of difficulty or intractability.

Formally, a *problem* consists of a question to be answered, a requirement to be fulfilled, or a best possible situation or structure to be found, called a *solution*. This is usually in response to several input *parameters* or *variables*. An *instance* of a problem $\mathcal{P}$ is a specification of particular values for its parameters. An *algorithm* for $\mathcal{P}$ is a step-by-step, terminating procedure, which when applied to any instance of $\mathcal{P}$ produces a solution. A *decision problem* is one that requires a simple "yes" or "no" answer – for example, the question, "Is the number of primes infinite?" is a decision problem.

There are mainly two kinds of questions we wish to answer when confronted with a particular problem $\mathcal{P}$.

- *Computability:* Is there an algorithm which solves the problem $\mathcal{P}$? A negative answer to this question would mean that $\mathcal{P}$ is *undecidable*.

- *Complexity:* This deals with the quantitative aspects of algorithm design. It addresses the issue of what can be computed in a practical or reasonable amount of time and space – it is dependent on the problem, the algorithm, and the implementation.

We usually express complexity as a function of the size of the input. We say that an algorithm $\mathcal{A}$ for a problem $\mathcal{P}$ runs in time $O(f(m))$ if for some positive constant c, there exists an implementation of $\mathcal{A}$ which terminates after at most $cf(m)$ computational steps, for *all* instances of size m – this is the notion of *worst-case complexity*.

1.2 Graph-theoretic Preliminaries

Graphs are discrete mathematical structures that have been studied extensively. They have a variety of practical applications, and are used to model and solve several real-world problems arising in diverse areas such as VLSI design, communication networks, and operations research. For mathematical results and properties of graphs, the reader is referred to [7, 20]; classical algorithmic results and applications are given in [7, 13, 19].

The graphs we deal with in this thesis, unless otherwise stated, are simple, finite, undirected, loopless, and without multiple edges.

A graph is a tuple $G = (V, E)$, where V is a finite set and E is a collection of unordered pairs of distinct elements of V. The elements of V are called the *vertices* of G, and the elements of E are called the *edges* of G. We denote $|V|$ by n, and $|E|$ by m.

Two vertices u and v such that $(u, v) \in E$ are said to be *adjacent* to each other, or *neighbours* of each other. Common representations of graphs involve the specification of adjacencies in terms of adjacency lists and matrices [1].

A *path* is a sequence of distinct vertices $v_1, v_2, \ldots, v_k$ such that v_i and v_{i+1} are adjacent for all $1 \leq i < k$. A path from u to v is denoted by $u \cdots v$. A *cycle* is a sequence of vertices $v_1, v_2, \ldots, v_k, v_1$ such that $v_1 \cdots v_k$ is a path and $(v_1, v_k) \in E$.

The *complement* of a graph $G = (V, E)$, denoted G^c, is the graph (V, E'), where $e \in E'$ if and only if $e \notin E$. A graph is said to be *connected* if there exists a path between any two vertices.

Given a subset S of V, the vertex-induced *subgraph* of S, denoted $G[S]$, (or S itself, when no confusion can arise), is the graph (S, E_s), where $E_s = \{(u, v) : u, v \in S \wedge (u, v) \in E\}$. An edge-induced subgraph is defined similarly. In this thesis, unless otherwise mentioned, all subgraphs are vertex-induced.

A graph(subgraph) is a *clique* if there is an edge between any two vertices, and it is *independent* if no two vertices are joined by an edge (i.e, its complement is a clique). The *chromatic number* of a graph is the minimum number of colours required to colour its vertices under the constraint that adjacent vertices get different colours.

The set of vertices adjacent to a vertex u defines the *neighbourhood* of u, denoted $N(u)$. The neighbourhood is said to be *closed* if it includes u, and is denoted $N[u]$.

Given two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, we define the *disjoint union* of G_1 and G_2, denoted $G_1 \cup G_2$ to be the graph $G = (V_1 \cup V_2, E_1 \cup E_2)$. We also define the *join* of the two graphs to be the graph $G = (V_1 \cup V_2, E_1 \cup E_2 \cup \{(x, y) : x \in E_1 \wedge y \in E_2\})$.

A *connected component*, (or simply *component*) of a graph is any subgraph that is maximally connected. A set of vertices S constitutes a *separator* of G if $G[V-S]$ has more components than G. If $|S|=1$, then the single element of S is called a *cut-vertex*. A *biconnected component*, or *block* of G is a maximal subgraph that has no cut-vertices.

In some applications, it may be useful to associate a *weight* with either the vertices or the edges of the graph. Such graphs are called *weighted graphs*; in this thesis, a weighted graph refers to one with weights associated only with the vertices.

1.3 Special Classes of Graphs

Several problems on general graphs are NP-complete, but admit polynomial solutions when restricted to special classes. Since the graphs used to model real-world problems are in general not arbitrary but have some special structure, we are motivated to design algorithms on such special classes.

One very important special class of graphs is the class of *perfect* graphs [5, 6, 19]. The maximum clique size of a perfect graph equals its chromatic number, for every induced subgraph. There are several important subclasses of the class of perfect graphs [19] – we briefly describe some of them below.

- *Chordal graphs:* These have no chordless cycles of length greater than three, because of which they are also referred to as *triangulated* graphs.

- *Comparability graphs:* The edges of a comparability graph are *orientable* in such a way that $(\vec{uv}) \wedge (\vec{vw}) \Rightarrow (\vec{uw})$.

- *Cocomparability graphs:* The complement of a comparability graph is cocomparability.

- *Interval graphs:* These are the graphs obtained by considering the intersection family of a set of intervals on a real line; these are both chordal and cocomparability.

- *Permutation graphs:* Given a permutation Π of the set $\{1, 2, \ldots, n\}$, we may define the graph $G_P = (V_P, E_P)$, where $V_P = \{1, 2, \ldots, n\}$ and $E_P = \{(i, j) : (i - j) \times (\Pi^{-1}(i) - \Pi^{-1}(j) < 0\}$. A graph isomporphic to G_P, for some Π, is said to be a permutation graph. Alternatively, permutation graphs are precisely the class of graphs that are both comparability and cocomparability.

- *Bipartite graphs:* In these graphs, the vertex set can be partitioned into two parts S and T such that the induced subgraphs $G[S]$ and $G[T]$ are both independent, i.e., all edges of G are between S and T. All bipartite graphs are comparability, since the edges can all be oriented in one direction (say, from S to T). All trees are bipartite.

- *Cographs:* These are graphs that have no induced paths of length greater than or equal to 3. All cographs are permutation graphs.

- *Distance-hereditary graphs:* Briefly, these are graphs in which all induced paths between any two vertices have the same length. These are described and studied in detail in Chapter 4.

The rest of this thesis is organized as follows. In Chapter 2, we describe the problems of domination and Steiner set, and mention the current status of these problems on some classes of graphs. Chapter 3 is devoted to solving the problems of connected domination and Steiner set on asteroidal-triple free graphs. In Chapter 4, we look at a few interesting properties of distance-hereditary graphs, and present an efficient algorithm to find all the minimal separators of such a graph. We use this algorithm and these properties to solve the problems of connected and clique domination, Steiner set, and maximum clique on distance-hereditary graphs in Chapter 5. Chapter 6 contains some concluding remarks and open problems.

Chapter 2

Domination and Steiner Problems

In this chapter, we describe the problems of domination and Steiner set, and mention the current status of these problems on some classes of graphs. We also describe an interesting application of the Steiner problem.

2.1 Domination Problems in Graphs

The earliest ideas of dominating sets date back to the origin of the game of chess in India centuries ago, in which one studies sets of chess pieces which cover or dominate opposing pieces or various squares of the chessboard. In recent years, several results have appeared in the literature related to the concept of domination in graphs – these results are theoretical, algorithmic, and applied in nature. Applications of the various kinds of domination include problems in communication networks, operations research (e.g., finding the optimal location of servers to service clients), etc. In this section we describe several kinds of domination problems in graphs, and pay special attention to the connected domination problem. For a comprehensive survey of domination in graphs, the reader is referred to [21].

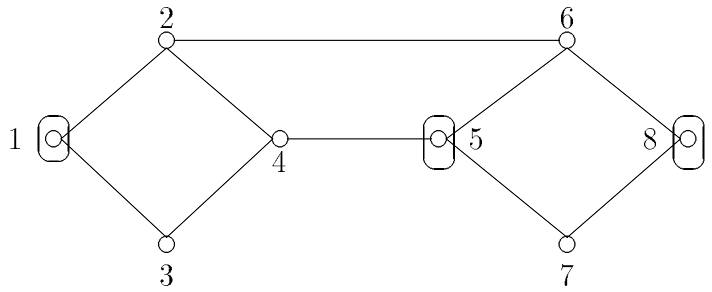

Figure 2.1: A minimum connected dominating set of this graph is $\{2, 4, 5, 6\}$. The vertices enclosed in ovals are the given set of terminal vertices; the set $\{2, 6\}$ is a minimum Steiner set for these vertices.

A *dominating set* of vertices in a graph $G = (V, E)$ is a set S such that any vertex of $V - S$ has a neighbour in S. A dominating set is *independent* if the subgraph induced on S has no edges, *clique* if it is complete, *total* if it has no isolated vertices, and *connected* if it is connected. Finding minimum dominating sets of these various types has attracted much attention. Connected domination involves finding a smallest connected subgraph which dominates the remainder of the vertices (see Figure 2.1). In the graph shown, $\{2, 4, 5, 6\}$ is a minimum connected domination set, and $\{1, 4, 8\}$ is a minimum dominating set. Since the size of any clique dominating set is at least as large as a connected one, and since the graph has no 4-clique, it has no clique that dominates it.

This problem, as well as the other types of domination problems, are NP-complete on the class of comparability graphs; some of them are NP-complete even on bipartite graphs [21, 18]. Connected domination is NP-complete for split and chordal graphs too, while polynomial-time algorithms are known for strongly chordal [28], and permutation graphs [2, 8].

In this thesis, we present polynomial-time algorithms for the connected domination problem on two special classes of graphs – AT-free graphs and distance-hereditary graphs. On AT-free graphs, our result generalizes the result of [2], but uses a totally different method based on dominating paths. On distance-hereditary graphs, we use the properties of minimal separators to solve this problem. In [11], d'Atri and Mascarini present an algorithm to solve this problem, which is polynomial, but quite complex. We present an $O(m+n)$ algorithm, thus improving their result. In addition, we also solve the clique domination problem on these graphs.

2.2 The Steiner Problem : Theory and Applications

A very simple but instructive problem was treated by Jacob Steiner, the famous representative of geometry at the University of Berlin in the early nineteenth century. "Three villages A, B, C are to be joined by a system of roads of minimum total length." This remark of Courant and Robbins (1941) led to the name of a problem that originated two hundred years earlier and should have been attributed to Fermat [16]. This geometric problem was already solved around 1640 : the fourth point p has to be chosen in such a way that joining the three given points to p creates three angles of exactly 120 degrees – assuming all angles in the triangle induced by the points are less than 120 degrees. Over the centuries this problem was rediscovered and generalized by many others, among them Jacob Steiner. Nowadays, two versions are especially interesting from an algorithmic point of view.

Problem 2.1 *"Euclidean Steiner Problem" : Given a (finite) set $T \subset R^2$.Find a shortest network that interconnects all points of T.*

Problem 2.2 *"Steiner Problem in Graphs" : Given a graph $G = (V, E)$, and a set of terminals $R \subseteq V$. Find a set $S \subset V$ of vertices such that*

- *$G[R \cup S]$ is connected, and*
- *$|S|$ is as small as possible.*

In this thesis, we deal only with the Steiner problem on graphs. An example of a graph and its Steiner set is shown in Figure 2.1.

2.2.1 Special Cases

Two well-studied problems in graph algorithmics turn out to be special cases of the Steiner problem. If R contains just two terminals, the Steiner problem reduces to the *shortest path* problem. If on the other hand R consists of the entire vertex set V, the Steiner problem coincides with the *minimum spanning tree* problem. Based on the classical algorithm of Dijkstra [14], Fredman and Tarjan [17] describe $O(n \log n + m)$ implementations for both problems using Fibonacci heaps.

In contrary to the polynomial solvability of these special cases, the general problem is NP-complete, as the following theorem shows.

Theorem 2.1 *(Karp, 1972) [23]. The Steiner problem in bipartite graphs is NP-complete.*

2.2.2 Algorithms on Special Classes of Graphs

Because the general Steiner problem is NP-complete, a lot of work has been done to design approximate algorithms, as well as algorithms on special classes of graphs, for this problem. We briefly review the current status of this problem on various classes. Arvind and Pandu Rangan [2] provide an $O(m + n \log n)$ algorithm for this problem on permutation graphs. White, Farber and Pulleyblank [28] consider this problem on chordal graphs and some of its subclasses such as strongly chordal graphs. Various approximate and (exponential) exact algorithms are also known for this problem; see [24] for a survey.

In this thesis, we present polynomial-time algorithms for this problem on AT-free graphs and distance-hereditary graphs. On AT-free graphs, our approach is based on the properties of dominating paths, and we obtain an $O(n^3)$ algorithm. On distance-hereditary graphs, we use the properties of minimal separators to solve this problem. We present an $O(mn)$ algorithm; another $O(mn)$ algorithm is known for this problem [11], but our approach is more general since it can be used to solve some other problems too with some modifications.

2.2.3 An Application of the Steiner Problem

Finding Steiner trees in graphs has proven to be one of the most essential tools in attacking the routing problem in VLSI-layout. Roughly speaking, the routing problem can be described as follows : given a graph G and a collection $N = \{N_0, N_1, \ldots, N_{r-1}\}$ of mutually disjoint subsets of the vertex set of G, find pairwise disjoint trees $S_0, S_1, \ldots, S_{r-1}$ such that S_i spans N_i for every i (but possibly contains more vertices than N_i). The intended interpretation is that the vertices in each of the subsets N_i form a net which has to be connected by wires. Wirings of different nets have to be disjoint to prevent short circuits from occurring.

If each of the sets N_i contains just two elements, the problem reduces to finding a collection of disjoint paths. Unfortunately, in practical applications typically more than half the nets contain more than two elements. Thus, in general, one has to find a collection of disjoint Steiner trees to solve the routing problem – and a minimum number of Steiner vertices must be found to minimise the cycle time of the chip.

Chapter 3

Algorithms on Asteroidal Triple-free Graphs

3.1 AT-free Graphs – Properties and Characterizations

An independent set of vertices x,y,z is an *asteroidal triple* if between every two of them there exists a path that avoids the neighbourhood of the third. An *asteroidal triple-free (AT-free) graph* is a graph that contains no asteroidal triples. AT-free graphs were first considered by Lekkerkerker and Boland [25] in the early 1960s, when they introduced and studied the properties of interval graphs.The following facts about the AT-free graphs are well-known.

Observation 3.1 *AT-free graphs are not necessarily perfect.*

Consider C_5, for example, which is AT-free but not perfect. However, the strong perfect graph conjecture (which states that a graph is perfect if and only if it contains no odd holes or anti-holes) has been shown to be valid on AT-free graphs.

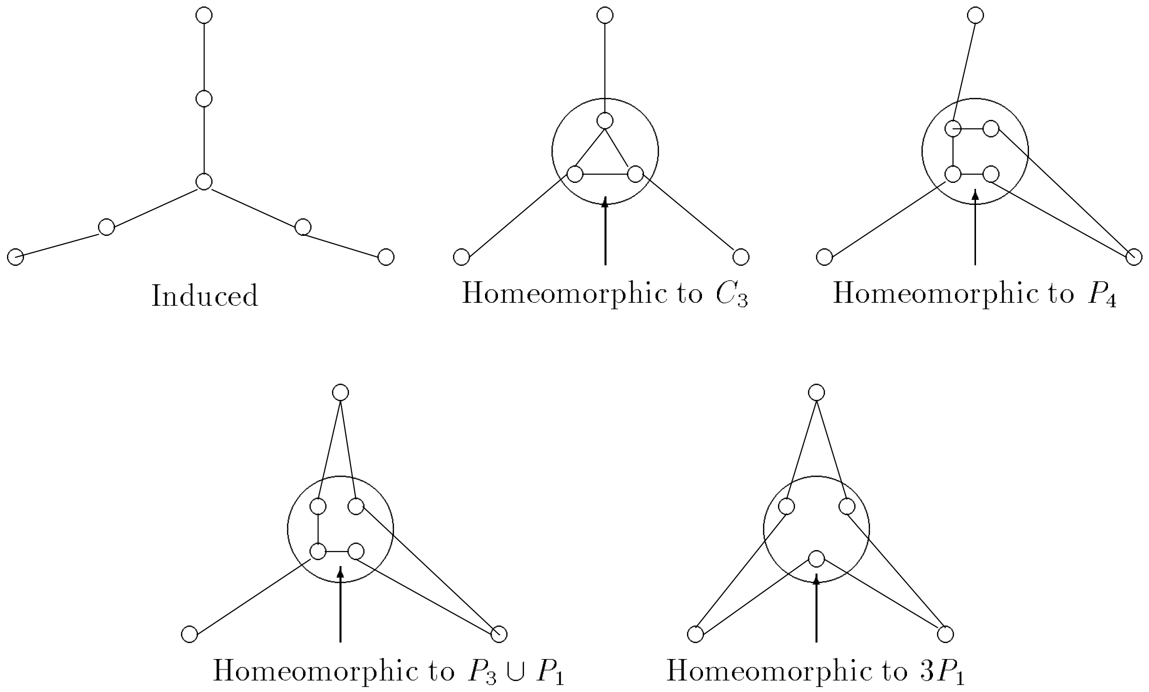

Figure 3.1: Forbidden configurations for AT-free graphs.

Observation 3.2 *G is an interval graph if and only if it is chordal and AT-free [25].*

Actually, an equivalent result states that a graph is interval if and only if it is chordal and cocomparability. Since AT-free graphs strictly contain the cocomparability graphs, this observation is stronger one way, and weaker the other.

Observation 3.3 *[9] Perfect AT-free graphs strictly contain the* cocomparability graphs.

The following theorem provides a characterization of AT-free graphs in terms of forbidden subgraphs.

Theorem 3.1 *[9] The graphs of Figure 3.1 are forbidden configurations for AT-free graphs.*

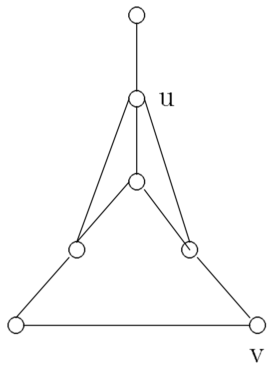

Figure 3.2: A dominating pair (u, v) in an AT-free graph.

AT-free graphs and their subclasses have many areas of applications such as in molecular biology, archaeology, information retrieval, computer memory management, channel routing, and circuit layout [9, 19, 12, 15].

3.2 Dominating Pairs in AT-free Graphs

We first introduce some notation. $dom(S)$ denotes the set of vertices dominated by an arbitrary $S \subseteq V$. Note that a set D is a dominating set of $G = (V, E)$ if and only if $dom(D) = V$. A pair of vertices (u, v) is said to be a *dominating pair* of a graph G if every $u \cdots v$ path in G dominates all vertices of G. Corneil et. al. [9] have stated a property of AT-free graphs in terms of dominating pairs. We state their result below as a theorem :

Theorem 3.2 *([9]) Every connected AT-free graph contains a dominating pair of vertices.*

Note that the presence of a dominating pair in not sufficient to conclude that the graph is AT-free, for instance, C_6 has a domainting pair, but is not AT-free. Figure 3.2 shows an AT-free graph and a dominating pair of vertices.

We now present a simple algorithm to find all the dominating pairs in an arbitrary graph. This is needed as a preprocessing step in our later algorithms. It is clear that a pair of vertices (u, v) is a dominating pair of G if and only if for any vertex $x \in V$, u and v lie in different connected components of $G - N[x]$. (Otherwise, there would exist a $u \cdots v$ path that did not pass through the closed neighbourhood of x). The algorithm proceeds by removing $N[x]$ from G for every vertex x in V, and finding the connectedness status of every other pair of vertices in the resulting reduced graph. We declare all pairs (u, v) which lie in different components in every such reduced graph to be the dominating pairs of G.

Algorithm *Dominating Pairs*

Input : An AT-free graph $G = (V, E)$. $|V| = n$, $|E| = m$.

Output : A list of all the dominating pairs of G.

A, A' : $array[1..n, 1..n]$ of boolean entries.

begin

1. initialize $A[i, j]$ to $true$ for $i = 1, 2, \ldots, n$ and $j = 1, 2, \ldots, n$;
2. for all vertices $v \in V$ do
3. construct $G' = G - N[v]$;
4. do a DFS of G' to find out its connected components;
5. for $i = 1, 2, \ldots, n$ and $j = 1, 2, \ldots, n$ do

. if vertices i and j are in the same connected component of G' then

$A'[i, j] \leftarrow false$

else

$A'[i, j] \leftarrow true$;

6. $A[i, j] \leftarrow A[i, j] \wedge A'[i, j]$;
7. for $i = 1, 2, \ldots, n$ and $j = 1, 2, \ldots, n$

if $A[i, j] = true$ then (i, j) is a dominating pair

end.

Theorem 3.3 *Algorithm* Dominating Pairs *above correctly finds all the dominating pairs of an AT-free graph G in $O(n^3)$ time.*

Proof. Correctness follows from the discussion above. We now prove the complexity bound. Step 1 takes time quadratic in n. Steps 3 and 4 can be executed in linear time for every iteration. Each execution of step 5 takes $O(n^2)$ time. Hence each execution of the loop in step 2 takes $O(n^2)$ time; so step 2 requires $O(n^3)$ time for execution. Step 7 takes $O(n^2)$ time. Hence the overall complexity is dominated by step 2 and is $O(n^3)$. □

3.3 Connected Domination on AT-free Graphs

In this section, we consider the problem of computing a minimum connected dominating set ($MCDS$) of an AT-free graph. This problem has been extensively studied on various classes of graphs [10, 8]. Arvind and Pandu Rangan [2] give an $O(m + n \log n)$ algorithm for this problem on the class of permutation graphs. We present an efficient algorithm on AT-free graphs, which are a large superset of cocomparability (and permutation) graphs. This algorithm can be used on any AT-free graph, regardless of whether it is perfect or not.

Let (u, v) be a dominating pair of G and let $X = N(u)$ and $Y = N(v)$. For each $x \in X$, let $A_x \subseteq V$ be the set of all vertices in $V - \{u\}$ such that if $a \in A_x$, then $\{a, x\}$ dominates all vertices in X. Similarly, for each $y \in Y$ let $B_y \subseteq V$ be the set of all vertices in $V - \{v\}$ such that if $b \in B_y$, then $\{b, y\}$ dominates all vertices in Y. Define Γ as follows :

$$\begin{aligned}\Gamma = \{P \mid\ & P = u \cdots v, && \text{or}\\ & P = u \cdots by, && \text{for } y \in Y, b \in B_y \text{ or}\\ & P = xa \cdots v, && \text{for } x \in X, a \in A_x \text{ or}\\ & P = xa \cdots by, && \text{for } x \in X, y \in Y, a \in A_x, b \in B_y\ \}\end{aligned}$$

Theorem 3.4 *Every path $P \in \Gamma$ such that $|P|$ is minimum in Γ is an MCDS of G.*

Proof. Let $P \in \Gamma$. We show first that P is a connected dominating set. Clearly P is connected. To show that $dom(P) = V$, we have to consider four cases :

Case 1: $P = u \cdots v$. In this case clearly $dom(P) = V$, since (u, v) is a dominating pair.

Case 2: $P = u \cdots by$, for $y \in Y$, $b \in B_y$. Note that $Q = Pv$ is a dominating path since (u, v) is a dominating pair. But $\{b, y\}$ dominates every vertex dominated by v, and so P is also a dominating path.

Case 3: $P = xa \cdots v$, for $x \in X$, $a \in A_x$. This case is symmetric to Case 2.

Case 4: $P = xa \cdots by$. In this case, note that $Q = uPy$ is a dominating set; but $\{a, x, b, y\}$ dominates every vertex dominated by $\{u, v\}$ and so P is a dominating path.

Next, we show that given any MCDS S, we can convert it into an MCDS S' such that $S' \in \Gamma$. Once again, we consider four cases:

Case 1: $u, v \in S$. In this case clearly $S' \in \Gamma$.

Case 2: $u \notin S, v \in S$. In this case since S dominates u, some vertex $x \in X$ is in S. Since S is connected, it has a $x \cdots v$ path P.
Subcase 2(a): If $S \neq P$, then consider $S' = P \cup \{u\}$. Clearly, S' contains a (u, v)

path and is therefore a dominating set of G with size smaller than or equal to that of S, satisfying $S' \in \Gamma$.

Subcase 2(b): In this case, $S = P$. If the vertex a adjacent to x in P belongs to A_x, clearly $S \in \Gamma$. If not, then there is some $x' \in X$ such that $\{x, a\}$ does not dominate x'. So x' must be dominated by some other vertex a' on the $x \cdots v$ path P. Replace $\{x, a\}$ in S ($= P$) by $\{u, x'\}$ to obtain S' from S. Clearly, S is connected because u is adjacent to x', x' is adjacent to a' ($\in S$), and S itself was originally connected. Now, S' is a $u \cdots v$ path, and so $S' \in \Gamma$. Also note that $|S'| = |S|$; this shows that the required transformation is possible in this case as well.

Case 3: $u \in S, v \notin S$. This case is symmetric to Case 2.

Case 4: $u \notin S, v \notin S$. In this case consider x, x', a, a' as in case 2(b) and replace $\{x, a\}$ by $\{u, x'\}$ to get S'' which satisfies the conditions of Case 3. Using the same technique as in Case 3, this can be transformed into an $MCDS$ $S' \in \Gamma$.

So in all cases we can transform S to $S' \in \Gamma$ as desired. This proves the theorem. □

3.3.1 The Algorithm

We present below an algorithm to find an MCDS of an AT-free graph G which relies on the above theorem.

Algorithm *Connected Domination*

Input: An AT-free graph $G = (V, E)$. $|V| = n, |E| = m$.

Output: An MCDS for G.

begin

1. Find a dominating pair (u, v) of G.
2. $X \leftarrow N(u), Y \leftarrow N(v)$.
3. for all $x \in X$ do

    construct A_x.

   Let $A \leftarrow \bigcup_{x \in X} A_x$.
4. for all $y \in Y$ do

    construct B_y.

   Let $B \leftarrow \bigcup_{y \in Y} B_y$.
5. do an All Pairs Shortest Paths on G and store the results in array D.

   i.e. $D[i, j]$ = length of a shortest (i, j) path.
6. Let

    $l_1 \leftarrow D[u, v]$

    $l_2 \leftarrow \min_{a \in A} D[a, v] + 1$

    $l_3 \leftarrow \min_{b \in B} D[u, b] + 1$

    $l_4 \leftarrow \min_{a \in A, b \in B} D[a, b] + 2$
7. $l \leftarrow min(l_1, l_2, l_3, l_4)$.
8. l gives the size of the MCDS. We can also easily construct the MCDS with the above information.

end.

3.3.2 Proof of Correctness

Theorem 3.5 *Algorithm* Connected domination *correctly computes the MCDS of an AT-free graph G in $O(n^3)$ time using $O(n^2)$ space.*

Proof. The correctness of the algorithm follows from the previous theorem. The only thing to note is that we add 1 or 2 to l_2, l_3 and l_4 in step 6 to take care of the fact that the actual dominating path must contain a vertex from X or Y or both, as stated in the previous theorem.

We now prove the complexity bound. Step 1 can be done in $O(n^3)$ time using Algorithm *Dominating Pairs*. Step 2 takes linear time. Each iteration of steps 3 and 4 takes $O(n^2)$ time, and so steps 3 and 4 require $O(n^3)$ time. Step 5 takes $O(n^3)$ time using the standard All Pairs Shortest Paths algorithm [1, 26]. Step 6 takes $O(n^2)$ time because there are totally $O(n^2)$ pairs of distances to be compared. Step 7 takes constant time. Step 8 can be executed in $O(m+n)$ time using a BFS to find the actual shortest path given the endpoints of the path. Hence the overall time complexity of the algorithm is $O(n^3)$.

Computing the dominating pair requires $O(n^2)$ space (Theorem 2.2). X and Y take $O(n)$ space as do each A_x and B_y. Observe that we can avoid storing each A_x (B_y) explicitly by constructing A (B) directly, thus requiring only $O(n)$ space. The All Pairs algorithm requires $O(n^2)$ space, and the subsequent BFS requires $O(m+n)$ space. Hence the overall space requirement is $O(n^2)$. □

The above algorithm computes the minimum cardinality connected dominating set of an AT-free graph in $O(n^3)$ time. We have exploited the property that there exists an $MCDS$ that induces a path (or a cycle) in the algorithm. On permutation graphs, *every* minimum *weighted* connected dominating set induces a path or a cycle. This property has been exploited in [2] to solve the weighted version of the problem on permutation graphs in $O(m+n\log n)$ time. However on AT-free graphs it is not necessary that there be a minimum *weighted* connected dominating set that induces a path or a cycle. (See Figure 3.3.) Hence the same approach may not be extendable to the weighted case on AT-free graphs.

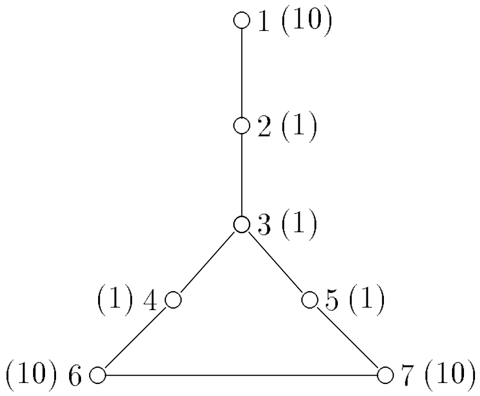

Figure 3.3: The weighted $MCDS = \{2, 3, 4, 5\}$ is not a path or a cycle (numbers in brackets are the weights of the vertices).

3.4 Steiner Set on AT-free Graphs

In this section we consider the problem of computing the Steiner set of an AT-free graph G, with respect to a set $R \subseteq V$ of vertices. The method used to solve this problem is similar to that used to solve the $MCDS$ problem in the previous section. It has already been observed that these two problems are algorithmically closely related [28] in the sense that they are either both polynomial or both NP-complete on all classes of graphs investigated so far. However, their precise relationship with each other is unknown. Arvind and Pandu Rangan [2] provide an $O(m + n \log n)$ algorithm for this problem on the class of permutation graphs. We present an $O(n^3)$ algorithm for the Steiner set problem on AT-free graphs.

We denote by R the set of vertices with respect to which the Steiner set is to be computed. Denote by $R_1, R_2, \ldots, R_k$ the k connected components of R. Let P be a dominating path in G between any dominating pair of vertices. Number the vertices of this path 1,2, $\ldots$, p, where $|P| = p$. Let the number assigned to any vertex v on the path be $number(v)$. Define

$min_R(P) = \{v | v \in P,\ v \in R$ or $(v, r) \in E$ for some $r \in R$, and $number(v)$ is minimum}, and

$max_R(P) = \{v | v \in P,\ v \in R$ or $(v, r) \in E$ for some $r \in R$, and $number(v)$ is maximum$\}$.

We define an R-dominating pair of an AT-free graph with respect to any $R \subseteq V$ to be a pair of vertices (u, v) such that every path from u to v dominates all vertices of R. This is a generalization of the concept of a dominating pair defined previously, which is just a V-dominating pair of G.

Construct an auxiliary graph G_R from G in the following manner :

1. Replace all the vertices of each connected component R_i by a single vertex r_i, with the adjacency of r_i equal to the union of the adjacencies of all the vertices in R_i.

2. Choose any dominating path P corresponding to some dominating pair. Compute $u = min_R(P)$ and $v = max_R(P)$.

3. Delete all vertices of P that are not numbered in the range $[number(u),\ number(v)]$. Let P_R be the corresponding path obtained.

4. Delete all vertices of $G - R$ that are not adjacent to any vertex of P_R.

5. Delete all vertices of $G - R$ that are adjacent to either u or v.

The graph thus obtained is $G_R = (V_R, E_R)$, which has certain interesting properties.

Lemma 3.1 *G_R is AT-free.*

Proof. Straightforward. For if not, there exists an asteroidal triple (x, y, z) in G_R. Since G_R is constructed from G by only deleting certain vertices and edges, and adding no new edges, (x, y, z) is an asteroidal triple in G as well – a contradiction. $\square$

Lemma 3.2 *$u = min_R(P)$ and $v = max_R(P)$ form an R-dominating pair in G_R.*

Proof. It is clear that the subpath P_R of $P = x \cdots y$ between u and v is a connected dominating set of G_R (this follows directly from the construction of G_R), and therefore dominates all vertices of R. If (u, v) is not an R-dominating pair of G_R, then there is some path P' between u and v, and some vertex $r \in R$ such that r is adjacent to no vertex on P'. Now consider the path $Q = (x \cdots u) \cup P' \cup (v \cdots y)$. Since (x, y) is a dominating pair of G, r must be adjacent to Q somewhere – moreover, this adjacency *must* be in the P'-segment of Q, since u and v are the smallest and largest numbered vertices on P to which any vertex of R is adjacent. This completes the proof. □

3.4.1 The Algorithm

We now present the actual algorithm for the Steiner set.

Algorithm *Steiner-Set*

Input: An AT-free graph $G = (V, E)$ and a set $R \subseteq V$ of vertices.

Output: A minimum cardinality Steiner set of G with respect to R.

begin

1. Construct the graph G_R as explained previously.
2. Let $u = min_R(P)$ and $v = max_R(P)$.
3. Let $X = N(u) \cap R$ and $Y = N(v) \cap R$.
4. As in the $MCDS$ algorithm, construct sets A_x and B_y.
   Use these to construct the sets A and B.
5. Construct a weighted graph $G' = (V', E')$ from G_R in the following manner:
   $wt(w) = 0$ if $w \in R$, and 1 otherwise.
6. As in the $MCDS$ algorithm, compute the shortest path among the four different kinds of paths using the All Pairs Shortest Paths algorithm.
7. Let P_m be the minimum weighted path.
8. Output $S = P_m - R$

end.

3.4.2 Proof of Correctness

Theorem 3.6 *Algorithm Steiner-Set presented above correctly computes the minimum cardinality Steiner set of an AT-free graph G with respect to a set $R \subseteq V$ of vertices.*

Proof. Let S be a Steiner set. If S is not already in a form that can be the output of the above algorithm, we show how it can be transformed to such a form, S'. Note that from the algorithm, (u, v) is an R-dominating pair of G_R, and that no two vertices of R are adjacent (since we have replaced each connected component of R by a single vertex). It is evident that this replacement does not affect the Steiner set – the same set is the output here as well. Since the weight of only those vertices in R is 0, and the others 1, it follows that a shortest weighted path will be one that minimises the number of vertices not in R, as we require. From the method of construction of the output set, S' in the above algorithm, it is apparent that $R \cup S'$ is *connected*. Define $T' = R \cup S'$. To show minimality, we distinguish the following cases :

Case 1. u and v are both in S.
In this case, the minimum weight path between u and v in $R \cup S$ sets a lower limit on the size of the Steiner set. The above algorithm attains this limit, and so is correct.

Case 2. $u \notin S$, $v \in S$.
We show that S can be transformed to a set S' of same cardinality that the above algorithm can produce. We distinguish two sub-cases:
Sub-case 2(a). Consider $T = R \cup S$. If in T, $r \in X$ is adjacent to some $a \in A_r$, we are done, because the above algorithm definitely considers a minimum weight A_r to v path.

Sub-case 2(b). $r \in X$ is adjacent to no $a \in A_r$.

Since T is connected, r is adjacent to some $a' \notin A$. In addition, $a' \notin R$, since each r is maximally connected, and so $wt(a') = 1$. The set X consists only of vertices of R (by construction); if r is the only vertex in X, we are done since this sub-case doesn't apply at all. Therefore, there is a vertex $r' \in X$, not adjacent to r, which is connected to S in some manner, say, to vertex $s \in S$. Replace $a' \in S$ by u to obtain S' from S. Now, $T' = R \cup S'$ is connected and also dominates all the vertices of R, since (u, v) is an R-dominating pair of G_R. In addition, we have replaced one vertex of unit weight (a') by another (u); therefore $|S'| = |S|$. In our algorithm we find a minimum weight $u \cdots v$ path, which sets a lower limit on the size of the Steiner set – a limit which we actually attain. Hence the required transformation is possible in this case too.

Case 3. $u \in S$, $v \notin S$.

This is symmetric to *Case 2* considered above, with u and v interchanged.

Case 4. Both u and $v \notin T'$.

In this case, as in cases 2 and 3, replace a by u and b by v to get a Steiner set of the same cardinality as S'. Formally, we can replace one of them and argue as in Case 2 (or 3).

Thus, in all cases, we can transform any Steiner set into one capable of being output by the above algorithm. This completes the proof. □

Theorem 3.7 *Algorithm Steiner-Set presented above terminates in $O(n^3)$ time.*

Proof. Step 1 takes $O(n^3)$ time since we require to determine a dominating pair. Steps 2 and 3 can be done in linear time. As in the *MCDS* algorithm, Step 4 requires $O(n^3)$ time. Step 5 is trivially linear, while Step 6 can be done in $O(n^3)$

time using the standard algorithm. Once again, the vertices on the path can be obtained in $O(m+n)$ time using a BFS, as we did in the $MCDS$. Thus, we see that the overall time complexity of the algorithm is $O(n^3)$. The algorithm can be implemented using $O(n^2)$ space, since we only require to store all graphs in the adjacency matrix form. The precise details are similar to those explained in the $MCDS$ algorithm. □

Theorem 3.8 *The minimum cardinality Steiner set problem can be solved on AT-free graphs in $O(n^3)$ time, using $O(n^2)$ space.*

Proof. Follows from the discussion above. □

3.5 Remarks

In this chapter, we have presented $O(n^3)$ algorithms on AT-free graphs for the problems of connected domination and Steiner set. To the best of our knowledge, no algorithms have so far appeared in the literature for solving these problems on any non-trivial class of non-perfect graphs. As already noted, AT-free graphs are not necessarily perfect. These algorithms also generalize the algorithms presented in [2] for connected domination and Steiner sets on permutation graphs. In addition, this is the largest known class of graphs that strictly contains both interval and permutation graphs on which these problems have been solved. However, we have not been able to generalize the algorithms for the weighted case, which remain open. We also note that the algorithms presented here are actually applicable to a class of graphs *larger* than the class of AT-free graphs – more specifically, to the class of graphs that contain a dominating pair of vertices. This class is a proper superset of the class of AT-free graphs (e.g., C_6 belongs to this class, but is not AT-free). Our results also strengthen the conjecture in [28] that the connected domination and Steiner set problems are algorithmically closely related. We also feel that AT-free graphs have nice structural properties which can be exploited to design algorithms for several problems which are NP-complete on general graphs.

Chapter 4

Properties of Distance-hereditary Graphs

In this chapter, we study a few properties of distance-hereditary graphs and the relationship of these graphs with some other important classes of graphs. We also investigate in detail the properties of separators in these graphs. Our algorithms for a variety of problems presented in Chapter 5 are based on these properties. We also present an $O(m+n)$ algorithm to find all the minimal separators of a distance-hereditary graph. This is used in all the algorithms of the next chapter.

4.1 Properties and Characterizations

A graph G is distance-hereditary if and only if for every pair $u, v \in V$, and for every connected induced subgraph H of G containing both u and v, $d_H(u,v) = d_G(u,v)$. In other words, a graph is distance-hereditary if and only if every two vertices have the same distance in every connected subgraph containing both of them; alternatively, all induced paths in a distance-hereditary graph are isometric. Distance-hereditary graphs were first introduced by Howorka [22]. Further characterizations in terms of metric properties, forbidden subgraphs, separators and relationships

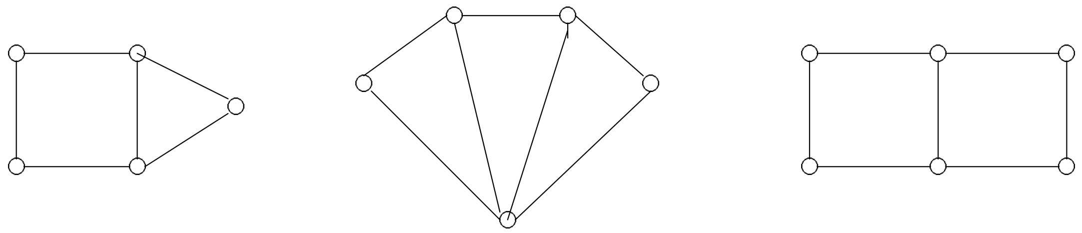

Figure 4.1: Some forbidden configurations for distance-hereditary graphs

with other graph classes have appeared in [4] and [11]. It is well-known that distance-hereditary graphs are perfect[4], and there exists an $O(mn)$ recognition algorithm for these graphs [11].

We state below some important known facts about distance-hereditary graphs as lemmas. We don't formally prove all of them here, but sometimes refer the reader to the appropriate papers where these are proved.

Lemma 4.1 *[22] Distance-hereditary graphs are precisely the class of graphs in which every cycle on five or more vertices has at least two crossing chords.*

Here, two chords (u, v) and (w, z) *cross* if the vertices u, w, v, z are distinct and in this order on the cycle.

Lemma 4.2 *[4] A graph G is distance hereditary if and only if none of the configurations shown in Figure 4.1, and $C_n, n > 4$, is a subgraph of G.*

It is easy to see that in each of these configurations there is a five cycle without two crossing chords.

Before we look at distance-hereditary graphs from the point of view of their separators, we first explore their relationship with some other important classes of graphs. We first consider an important subclass of distance-hereditary graphs, called *cographs*. Cographs are the class of graphs which contain no induced path

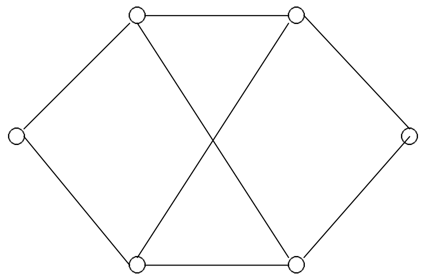

Figure 4.2: A distance-hereditary graph that is not a cograph

on four vertices (denoted by P_4). They are exactly the class of distance-hereditary graphs with diameter less than or equal to two [4], and may be defined recursively starting from K_1 as follows: if G_1 and G_2 are cographs, so are their disjoint union and join (Chapter 1 defines these terms).

Lemma 4.3 *All cographs are distance-hereditary, and there are distance-hereditary graphs that are not cographs.*

Figure 4.2 shows a distance-hereditary graph that is not a cograph.

Lemma 4.4 *[4] All distance-hereditary graphs can be obtained starting from K_1 by an application of the following operations:*

1. Operation F – A fraternal vertex split: *Create a new vertex v similar to an existing vertex u such that $N(u) = N(v)$, and $(u, v) \notin E$.*

2. Operation N – A non-fraternal vertex split: *Create a new vertex v similar to an existing vertex u such that $N(u) = N(v)$, and $(u, v) \in E$.*

3. Operation P – Pendant vertex addition: *Add a pendant vertex (a vertex of degree one) to any existing vertex of the graph.*

The above lemma allows us to construct distance-hereditary graphs by a sequence of *local* operations, in contrast to a sequence of *global* operations that a recursive definition such as the one used for cographs would entail. We have already mentioned that all cographs are distance-hereditary, so it is natural to ask what restrictions on the above three operations produce cographs (through a sequence of local operations). We answer this and related questions below. We first make an important observation.

Observation 4.1 *P and N operations do not introduce a new (chordless) C_4 in the graph. In fact, a P operation introduces no new cycle at all.*

Various combinations of F, N, and P operations produce the following subclasses of distance-hereditary graphs.

- If we restrict ourselves to operation P alone, starting from K_1, we obtain all *trees*. This immediately means that all trees are distance-hereditary.
- Operation N alone produces all cliques.
- Operation F alone produces just isolated vertices.
- Operations F and N produce the class of cographs.
- Operations N and P produce all distance-hereditary graphs that are chordal – this is the class of *Ptolemaic* graphs.
- Operations F and P alone produce the class of distance-hereditary graphs that have no odd cycles, i.e., the class of bipartite distance-hereditary graphs, also called *6-2 chordal bipartite graphs*.

We also mention here that distance-hereditary graphs are a subset of the class of *parity graphs*, which are the graphs in which all induced paths between a given pair of vertices have the same parity.

Now that we have stated some important characterizations of distance-hereditary graphs, and have described their relationship to other important graph classes, we are in a position to investigate the properties of separators in these graphs.

4.2 Separators in Distance-hereditary Graphs

We denote the subgraph $G[V-S]$ by $G-S$, and $G[V-\{v\}]$ by $G-v$. S is a *separator* of G, if the number of connected components of $G-S$ is greater than that of G. S is said to be an (x,y)-separator of a connected graph G is if x and y lie in different components of $G-S$. A separator S is said to be a *minimal separator* if no proper subset of S is a separator. A separator of cardinality one is called a *cut-vertex*. We denote by $d_G(u,v)$, the distance between vertices u and v in the graph G; the subscript may be dropped if the graph is clear from the context.

The following theorem, due to d'Atri and Mascarini [11], provides an important characterization of distance-hereditary graphs in terms of their minimal separators.

Theorem 4.1 *[11] G is a distance-hereditary graph if and only if for every $T \subseteq V$ and for every vertex v in a minimal separator S of $G[T]$, we have $N_{T-S}(v) = N_{T-S}(S)$.*

Corollary 4.1.1 *Let X be a minimal separator of a distance-hereditary graph G and let $Y = N(X)$. Let $C = \{(x,y) : x \in X, y \in Y\}$. Then, the edges of C induce a complete bipartite subgraph of G with X and Y as its bipartitions.*

This means that the edges between every minimal separator S and its neighbourhood $N(S)$ form an edge-induced complete bipartite subgraph of G. It also implies that, as far as the rest of the graph is concerned, all the vertices of a minimal separator are identical. These facts are the central ideas on which our algorithms on distance-hereditary graphs are based. We state below another important and interesting lemma.

Lemma 4.5 *The induced subgraph on any minimal separator S of a distance-hereditary graph is a cograph.*

Proof. If not, then S has a P_4, say P, as a subgraph. Let v be any vertex in $G-S$ that is adjacent to some vertex of S (the existence of such a vertex is guaranteed). By Theorem 4.1, v is adjacent to all vertices of S. The sugraph $G[P \cup v]$ is a C_5 without two crossing chords; hence the result. $\Box$

We define the *hanging* (also known as a *level diagram*, or a *neighbourhood diagram*) of G with respect to a vertex u to be a collection of sets $N_1(u), N_2(u), \ldots, N_d(u)$, such that if $v \in N_i(u)$, then $d(u,v) = i$. In other words, a hanging is simply what is obtained upon doing a *breadth first search (BFS)* [1] of G, starting from vertex u. Define the *interval*, $I(u,v)$, of two vertices u and v of a distance-hereditary graph G as follows.

$$I(u,v) = \{x : x \text{ lies on some shortest } u \cdots v \text{ path}\}.$$

Denote $N_k(u) \cap I(u,v)$ by $N_k(u,v)$ for all $1 \leq k < d(u,v)$.

Lemma 4.6 *Let G be a distance-hereditary graph and let u and v be some two vertices of G. Consider the hanging of G from u, and let $d(u,v) = k$. Then, $N_{k-1}(u,v) = N_{k-1}(u) \cap N(v)$.*

Proof. It is clear that all the vertices x that lie on a shortest $u \cdots v$ path such that $d(u,v) = k-1$, i.e, the set $N_{k-1}(u,v)$, are obtained by considering the adjacencies of v in the previous level; hence the result. $\Box$

Lemma 4.7 *Let x and y be two vertices in the same level k of the hanging of a distance-hereditary graph from vertex u, i.e, let $d(u,x) = d(u,y) = k$. Then, if x and y are in the same connected component of $G[N_k(u)]$, then $N(x) \cap N_{k-1}(u) =$*

$N(y) \cap N_{k-1}(u)$. If not, then either the two intersections are disjoint, or one is a subset of the other.

Proof. Suppose x and y are in the same connected component, but have different adjacencies in the previous level. Then, induced paths of different lengths between u and x (one directly, and the other through y) are present in G, a contradiction. If, on the other hand, x and y are in different components of $G[N_k(u)]$, and the two intersections are neither disjoint nor such that one contains the other, then once again there are non-isometric induced paths between u and x in G. This completes the proof. $\square$.

Lemma 4.8 *[4] For any vertex u, $N_k(u)$ is a cograph.*

Proof. If $N_k(u)$ had a P_4, P, then by the previous lemma there is a vertex v in $N_{k-1}(u)$ adjacent to all the vertices of P. But then $G[P \cup v]$ induces a 5-cycle without two crossing chords. $\square$

Lemma 4.9 *If a vertex v of $N_k(u)$ is adjacent to two non-adjacent vertices x and y in the same connected component of $N_{k-1}(u)$, then the vertices of this component which are not adjacent to v are adjacent to both x and y, or none.*

Proof. Let, if possible, $t \in N_{k-1}(u)$ be a vertex in the same connected component as x and y, such that $(t,x) \in E$ and $(t,y) \notin E$. Since $(x,y) \notin E$, there is a P_4 (x,t,w,y) in $N_{k-1}(u)$. This is not possible since, by Lemma 4.5, the induced subgraph on each level is a cograph. $\square$

The above lemma (due to Bandelt and Mulder) originally appeared in [4], but the requirement that x and y be non-adjacent was not mentioned, which was an error. If they were adjacent, the statement wouldn't necessarily be true – see Figure 4.3, for instance.

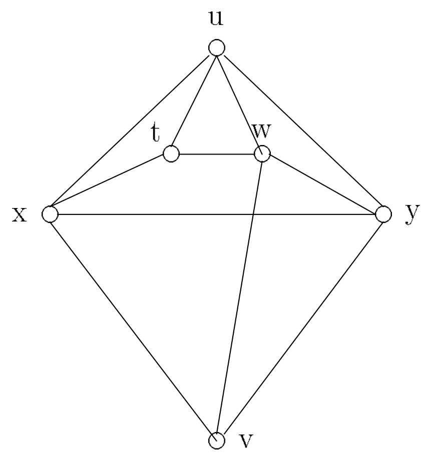

Figure 4.3: Figure showing a counter-example to the claim of Bandelt and Mulder

Lemma 4.10 *Same conditions as in the previous lemma. Then, $N_{k-1}(u,x)$ is a minimal (u,x) separator in G.*

Proof. Let $X = N_{k-1}(u,x)$. We first show that X is a (u,x) separator. If not, then there is a path P from x to u passing through a level numbered *larger* than $d(u,x)$ as well as *another* vertex y in the same level as x. If x and y are in the same connected component in $G[N_k(u)]$, then by Lemma 4.7 such a path P is not possible since x and y have the same adjacencies in $N_{k-1}(u)$. If they are in different components, then there are induced paths of lengths k and greater than k between u and x in G. Hence X is a (u,x) separator. If X were not minimal, a proper subset S of X is – and there will surely be a path from u to x passing through any $t \in X - S$. $\square$

The above lemma suggests a (somewhat inefficient) method by which we can obtain all the minimal separators of a distance-hereditary graph G. For each pair of vertices (u,v), find $N_i(u,v)$, which is clearly a separator of G (since it is a (u,v) separator). In addition, *all* minimal separators are clearly included in this set, so it is an easy matter to sequentially maintain and update a list (set) of such separators. This crude method takes roughly $O(mn)$ time.

We now present an efficient algorithm to find all the minimal separators in $O(m+n)$ time.

Algorithm *Candidate-Separators*

Input: A distance-hereditary graph G.

Output: A list L, which is a list of candidate separators of G.

{Note that some elements of L need not be minimal separators, though}

begin

Initialize L to an empty list;

Use a BFS to construct the hanging of G with respect to u;

Let d be the depth of the hanging;

Treat u as part of both levels 1 and 2;

for $i \leftarrow d$ downto 2 do begin

for each $v \in N_i(u)$ do

$S \leftarrow N_{i-1}(u) \cap N(v)$;

append(L, S);

endfor;

end.

We call each element of the list L at the end of the algorithm a *candidate separator.*

Theorem 4.2 *Algorithm Candidate-Separators finds all the minimal separators of a distance- hereditary graph G in $O(m+n)$ time.*

Proof. Each set obtained in the list L is clearly a separator of G, since it is a (u,x) separator for some x. Also, by Theorem 4.1, a minimal separator cannot include vertices from more than one level (the only exception to this is vertex u itself, which is treated as being part of level 1 as well as level 2, since it can share adjacencies fully with some vertices in both these levels). Note that all minimal separators are (x, y)-separators for some x and y. To show that all these separators are obtained as candidate separators, we distinguish the following cases.

Case 1: $x = u$ or $y = u$. This case is trivial, since by construction all candidate separators are (u, v)-separators.

Case 2: x and y are in the same level, k. This implies that $d(u, x) = d(u, y) = k$.
Sub-case 2(a): x and y are in the same connected component of $G[N_k(u)]$. In this case, this (x, y)-separator *cannot* be minimal, since there is a path from x to y through previous levels, as well as through the current level, and a minimal separator will *have* to include vertices from both. This is forbidden by Theorem 4.1.
Sub-case 2(b): x and y are in different components. In this case, there clearly exists a $x \cdots y$ path passing through earlier levels. But, at each vertex, we keep track of the set of vertices it is adjacent to in the earlier level, and if two vertices are in different components, these sets have an empty intersection.

Case 3: x and y are in different levels. Without loss of generality, let x be in a level earlier than y. Since $(x, y) \notin E$, any $x \cdots y$ path must pass through $N_{k-1}(u) \cap N(y)$.
The time complexity follows from the fact that the BFS takes $O(m+n)$ time, and after that each edge is considered at most once. append(L, S) takes $O(n)$ time to perform, so the overall time complexity is $O(m + n)$. $\square$

Corollary 4.2.1 *Minimal separators in a distance-hereditary graph are vertex-disjoint.*

Corollary 4.2.2 *There are $O(n)$ minimal separators in a distance-hereditary*

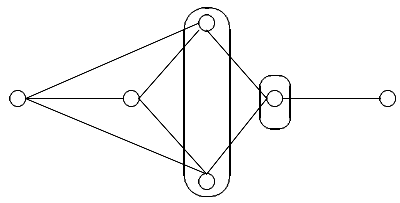

Figure 4.4: A distance-hereditary graph and its vertex-disjoint minimal separators

graph.

From the above corollaries, it follows that the procedure append(L, S) needs to maintain only vertex disjoint subsets of V, and so its size never exceeds n. Which is why append(L, S) takes $O(n)$ time to do.

Corollary 4.2.3 *A minimum cardinality separator of a distance-hereditary graph can be found in $O(m + n)$ time.*

A distance-hereditary graph and its vertex-disjoint minimal vertex separators are shown in Figure 4.4.

Lemma 4.11 *Let X be a minimal separator of a distance-hereditary graph G. Let H be obtained from G by identifying all $v \in X$. Then H is distance-hereditary as well.*

Proof. Straightforward. For otherwise, there would be non-isometric paths between the same two vertices in G, as in H. $\square$

Chapter 5

Algorithms on Distance-hereditary Graphs

In this chapter, we solve the problems of connected and clique domination, Steiner set, and maximum clique using a unified approach based on the properties of minimal separators described in the previous chapter. The algorithm presented in Chapter 4 to find all the minimal separators of a distance-hereditary graph is used in these algorithms.

5.1 Domination Problems

In this section, we solve the problem of computing a minimum weighted connected dominating set and a minimum weighted clique dominating set of a distance-hereditary graph. For the first problem, we are given a distance-hereditary graph G and are required to find a connected set of minimum possible weight that dominates G. For the second problem, such a set must be a clique, i.e, the subgraph of G induced on it must be complete. An algorithm due to d'Atri and Mascarini [11] is already known for the first problem, but this algorithm, though polynomial, is quite complex. We present efficient linear algorithms for both problems using the properties of separators; more precisely, by exploiting the manner in which minimal separators are connected with one another. Both these problems are NP-complete on general graphs; their status on some other special classes of graphs has been mentioned in Chapter 2.

We first introduce some notation. Let G be a distance-hereditary graph, let $X_1, X_2, \ldots, X_k$ be its minimal separators, and let $X = \bigcup_{1 \le i \le k} X_i$. By Corollary 4.2.1 to Theorem 4.2, $X_i \cap X_j = \phi$. Let $\mathcal{S}$ be an auxiliary graph obtained from G in the following manner: $\mathcal{S} = (\mathcal{X}, \mathcal{E}')$, where $(X_i, X_j) \in E'$ if and only if there is an edge in G between some $u \in X_i$ and some $v \in X_j$. Note that this also means that $\forall u \in X_i, \forall v \in X_j, (u, v) \in E$.

Lemma 5.1 *$G[X]$ is a connected subgraph of G.*

Proof. Suppose not, then in G there exist minimal separators X_i and X_j such that all paths between them pass through vertices T not part of *any* X_k; however, T would constitute a separator of G, and hence some subset of it would be minimal.
□

In other words, the above lemma implies that the graph $\mathcal{S}$ is connected.

Lemma 5.2 *X is a dominating set of G.*

Proof. Suppose not, then $\exists w \in V : d(w, X) \ge 2$ (since w is not adjacent to any X_i). Consider the first neighbourhood of w in G, $N_G(w)$, which constitutes a separator of G (its removal disconnects w from the rest of the graph) – a subset of this set is minimal, and would therefore be adjacent to w. □

Lemma 5.3 *Any connected dominating set of G must include a vertex from each X_i.*

Proof. Suppose there is some minimal separator X_i such that no vertex from it is included in some connected dominating set, D. Since D is connected, and all other vertices in $V - X_i$ are adjacent to some vertex of D, $G[V - X_i]$ is connected. This contradicts the fact that X_i is a separator of G; hence the result. □

5.1.1 Connected Domination

The above lemmas immediately suggest an algorithm to find the minimum weighted connected dominating set of a distance-hereditary graph.

Algorithm *DH-ConnectedDomination*

Input: A distance-hereditary graph G.

Output: A minimum weighted connected dominating set C of G.

begin

  $C \leftarrow \phi$;

  $weight \leftarrow 0$;

  Use Algorithm *Candidate-Separators* to find the

    set of minimal separators $X = \{X_1, X_2, \ldots X_k\}$;

  for $i \leftarrow 1$ to k do begin

    let s be a vertex of minimum weight in X_i;

    $C \leftarrow C \cup \{s\}$;

    $weight = weight + weight(s)$;

  end;

end.

Theorem 5.1 *Algorithm DH-ConnectedDomination finds a minimum weighted connected dominating set of a distance-hereditary graph in $O(m + n)$ time.*

Proof. Correctness follows from the lemmas and discussions above, since in the algorithm we clearly include a vertex of smallest weight from every minimal separator in the set. Finding the set of all minimal separators takes $O(m+n)$ time. Once this is done, the algorithm considers each vertex at most once, performing a constant amount of work each time. Hence the time complexity of the algorithm is $O(m+n)$. $\square$

5.1.2 Clique Domination

We wish to determine a dominating set S of a distance-hereditary graph G such that $G[S]$ is a clique, with as small a total weight as possible. Our algorithm for clique domination is an extension of the one used for connected domination. The following lemma is the basis for the algorithm.

Lemma 5.4 *A d-h graph G has a clique dominating set if and only if the algorithm for MCDS described in Section 5.1 produces a clique as output.*

Proof. Every clique dominating set is obviously connected, and any connected dominating set of G must include at least one vertex from each cutmodule. Now, given any minimal separator T, there cannot be edges between vertices in different components of $G[V-T]$; hence, such vertices cannot be part of the same clique. This means that a clique dominating set, if any, can include only vertices of that belong to some minimal separator. $\square$

5.2 Steiner Set

A problem that is closely related to the connected domination problem is the Steiner set problem. Here, we are given a graph $G=(V,E)$ and a set $R \subseteq V$ of vertices, and are required to compute a set $S \subseteq V$ such that $G[R \cup S]$ is connected, and $|S|$ is as small as possible.

We present below an algorithm to compute the Steiner set of a distance-hereditary graph G with respect to a set R of vertices.

Our algorithm works as follows. If R is already connected, then there's nothing to compute. If all the vertices of R are part of the same minimal separator (but

R is not connected), then any vertex v adjacent to some vertex of this separator (but not part of the separator itself) constitutes the required Steiner set. This is because v is adjacent to all the vertices of the separator (and hence to all the vertices of R), and so $G[R \cup \{v\}]$ is connected. If, on the other hand, R has no vertex in any minimal separator, but all the vertices of R have the same adjacencies in the set X, then the same argument applies here too. This situation is detected by the function $find\_if\_same\_adj()$ in the algorithm. Finally, if none of the above situations hold, we check (one by one) if the removal of minimal separator X_i disconnects G, such that the vertices of R lie in different components. If so, we include (since we must) some vertex from X_i into our set S. It is not difficult to see that the above cases exhaust all possibilities and indeed lead to a correct Steiner set being computed.

Algorithm *DH-SteinerSet*

Input: A distance-hereditary graph G and a set R of vertices.

Output: A Steiner set S for G.

begin

Use Algorithm *Candidate-Separators* to find the

set of minimal separators $X = \{X_1, X_2, \ldots X_k\}$;

if R is connected then

$S \leftarrow \phi$;

else if $R \subseteq X_i$ for some X_i then begin

let $s \notin X_i$ be a vertex adjacent to some $r \in R$;

$S \leftarrow \{s\}$;

endif;

else if $R \subseteq V - X$ then begin

$flag \leftarrow find\_if\_same\_adj(R)$;

if $flag = TRUE$ then begin

```
            let s ∈ X be a vertex adjacent to some r ∈ R;
            S ← {s};
        endif;
    endif;
    else
        for i ← 1 to k do begin
            if R ∩ X_i = φ then begin
                let H_1, H_2, ..., H_l be the connected components of
                G − X_i that contain some vertex of R;
                if k > 1 then begin
                let s be some vertex in X_i;
                S ← S ∪ {s};
                endif;
            endif;
        endfor;
end.
```

When the above algorithm terminates, S is the required set of Steiner vertices. The function *find_if_same_adj()*, takes as input a set R of vertices, and for all those vertices in this set that don't belong to X (the set of vertices in some minimal separator), determines if they all have the same adjacencies outside R. It is formally stated below. In this function, $adj(v)$ is the set of all vertices belonging to X that $v \in R$ is adjacent to.

Function *find_if_same_adj*

Input: A set R of vertices of a distance-hereditary graph G
that has no vertex in any minimal separator.

Output: TRUE if all vertices of R have the same adjacencies
in X, FALSE otherwise.

begin

$adjset \leftarrow adj(v)$ for some $v \in R$;

for each $w \in R - \{v\}$ do

if $adj(w) \neq adjset$ then

return FALSE;

return TRUE;

end.

Theorem 5.2 *Algorithm DH-SteinerSet finds the Steiner set of a distance-hereditary graph G in $O(mn)$ time.*

Proof. Correctness follows from the discussion above. The function $find\_if\_same\_adj()$ takes $O(n^2)$ time, since it does $O(n)$ iterations, each iteration requiring $O(n)$ work. The last case when the vertices of R are "spread out" in G requires $O(n)$ iterations to process, each doing $O(m+n)$ work; hence the overall time complexity of the algorithm is $O(mn)$. □

5.3 Maximum Clique

We now consider the problem of determining a clique of largest size in a distance-hereditary graph. This problem is NP-complete on general graphs, but has been shown to be polynomial on perfect graphs. However, the complexity of the algorithm on perfect graphs is very high and has large embedded constants, because of which it is interesting and useful to study this problem on subclasses of perfect graphs.

Algorithm *DHMaxClique*

Input: A distance-hereditary graph $G = (V, E)$.

Output: A maximum clique of G.

begin

if G is disconnected then begin

let $G_1, G_2, \ldots, G_l$ be its components;

$maxcliq \leftarrow \max_i(DHMaxClique(G_i))$;

endif;

else if G is a join of two graphs G_1 and G_2 then

$maxcliq \leftarrow \max(DHMaxClique(G_1) + |V_2|, |V_1| + DHMaxClique(G_2))$;

else begin { neither G nor G^c is connected }

Use Algorithm *Candidate-Separators* to find

a minimal separator S;

Let $H_1, H_2, \ldots, H_t$ be the components of $G - S$;

Let $G_i \leftarrow G[V_i \cup S]$;

$maxcliq \leftarrow \max_i(DHMaxClique(G_i))$;

endelse;

end.

Theorem 5.3 *Algorithm DHMaxClique computes the maximum clique of a distance-hereditary graph G in $O(n^3)$ time.*

Proof. Correctness follows from the fact that the maximum clique cannot include two vertices from different components of $G - X$ for any separator X. To show the time bound of $O(n^3)$, we show that in $O(n)$ iterations, we have only cographs to handle (disjoint union or join cases). In each iteration, we do $O(n^2)$ work, which is the cost of determining if either G or G^c is disconnected (G here refers to the "current" graph being processed). Consider the function $\phi = n - k + c$, where n is the number of vertices, k is the number of components, and c is the number of minimal separators, all in the current graph. This is a monotonically decreasing function, and is bounded above and below by $O(n)$. Hence the number of iterations before a cograph is obtained is $O(n)$, leading to an overall time complexity of $O(n^3)$.

5.4 Remarks

In this chapter, we have presented algorithms for the problems of connected and clique domination, Steiner set, and maximum clique of distance-hereditary graphs. The algorithm presented for the maximum clique problem can easily be extended to the weighted case as well. We have solved these problems based on a unified approach motivated by the properties of minimal separators. It would be interesting to investigate if the same or similar techniques can be extended to obtain efficient algorithms for some other problems too. We mention here that we have solved the problem of computing the treewidth of distance-hereditary graphs [27] using an extension of the techniques presented in this Chapter.

Chapter 6

Concluding Remarks and Open Problems

6.1 Conclusions

An important method of overcoming the intractable nature of problems in graphs is to focus attention on special classes of graphs and try to obtain solutions for them. If a problem remains intractable even on some special class, it is worth studying what structural properties of this class make the problem inherently difficult on it. This is followed by investigating the same problem on related classes and sub-classes of this class, and to provide as "narrow" a margin as possible between tractability and intractability for the problem – colloquially, this may be referred to as "narrowing down the gap between P and NP", on some hierarchy of graphs. The whole exercise leads to a thorough understanding of the problem at hand, and of the properties of the graphs under consideration.

The results presented in this thesis are as follows :

- An $O(n^3)$ algorithm for the connected domination problem on AT-free graphs.

- An $O(n^3)$ algorithm for the Steiner set problem on AT-free graphs.
- An $O(m+n)$ algorithm for the weighted connected domination problem on distance-hereditary graphs.
- An $O(m+n)$ algorithm for the weighted clique domination problem on distance-heredtary graphs.
- An $O(mn)$ algorithm for the Steiner set problem on distance-hereditary graphs.
- An $O(n^3)$ algorithm for the weighted maximum clique problem on distance-hereditary graphs.

In this thesis, we have presented $O(n^3)$ algorithms on AT-free graphs for the problems of connected domination and Steiner set [3]. To the best of our knowledge, no algorithms have so far appeared in the literature for solving these problems on any non-trivial class of non-perfect graphs. As already noted, AT-free graphs are not necessarily perfect. These algorithms also generalize the results presented in [2] for connected domination and Steiner sets on permutation graphs, but use an entirely different approach based on dominating pairs. In addition, this is the largest known class of graphs that strictly contains both interval and permutation graphs on which these problems have been solved. We also note that the algorithms presented here are actually applicable to a class of graphs *larger* than the class of AT-free graphs – more specifically, to the class of graphs that contain a dominating pair of vertices. This class is a proper superset of the class of AT-free graphs (e.g., C_6 belongs to this class, but is not AT-free). We also feel that AT-free graphs have nice structural properties (especially in the context of dominating pairs) which can be exploited to design algorithms for several problems which are NP-complete on general graphs.

On distance-hereditary graphs, we have presented algorithms for many problems that are NP-complete on general graphs. We have used the properties of separators in these graphs as a paradigm to design a unified approach to tackle these problems. The underlying theme that makes such a unified approach possible is the fact that vertices of a minimal separator are indistinguishable outside the separator, i.e, they all have the same adjacencies outside. This approach has been used to solve the problem of computing the treewidth of a distance-hereditary graph as well. The previous best known algorithm for the connected domination problem was $O(n^3)$ [11]; we have improved the complexity to $O(m + n)$. We feel that this paradigm can be used to solve some other problems on distance-hereditary graphs, that are NP-complete on general graphs.

Our algorithms for connected domination and Steiner set on AT-free and distance-hereditary graphs strengthen the conjecture in [28] that these two problems are algorithmically closely related, in the sense that they have the same complexity status on any special class of graphs.

6.2 Open Problems

We conclude this chapter by suggesting some related open problems and directions for future work.

Problem 6.1 *The weighted version of the connected domination problem on AT-free graphs.*

We showed in Section 3.3.1 why we were unable to extend our algorithm for connected domination to the weighted case – perhaps a different approach (or at any rate, a modified one) is required to solve this problem.

Problem 6.2 *Recognition of AT-free graphs.*

An $O(n^3)$ recognition algorithm is known for the class of AT-free graphs. A more efficient algorithm is worth trying for.

Problem 6.3 *Domination problems (such as minimum domination), maximum clique, independent set, and related problems on AT-free graphs.*

These are unsolved problems; we strongly believe that the properties of dominating paths can be exploited to solve some of these problems. We mention here that we have an inefficient (but polynomial) algorithm for the maximum clique problem.

Problem 6.4 *The linear structure of AT-free graphs, and a numbering scheme.*

Subsets of AT-free graphs such as cocomparability graphs have a linear numbering scheme on the vertices – in [9], the authors conjecture that the reason for such a numbering on these graphs is a consequence of their AT-free nature. However, a linear numbering for AT-free graphs is still unknown.

Problem 6.5 *A general relationship between the connected domination and Steiner set problems.*

On all classes of graphs studied so far, these two problems have the same complexity status. However, a polynomial transformation from one to the other is not known in general. Our algorithms for these problems on AT-free and distance-hereditary graphs further strengthen the conjecture that these two problems are algorithmically closely related to each other. The status of these problems on some classes of graphs is shown in Figure 6.1. Polynomial transformations from one problem to the other are known on some special classes of graphs such as bipartite graphs; the discovery of such a transformation in general (i.e, for any class of graphs – the tough part lies in ensuring that the transformation produces a graph belonging to the same class) would settle long-standing open problem in algorithmic graph theory.

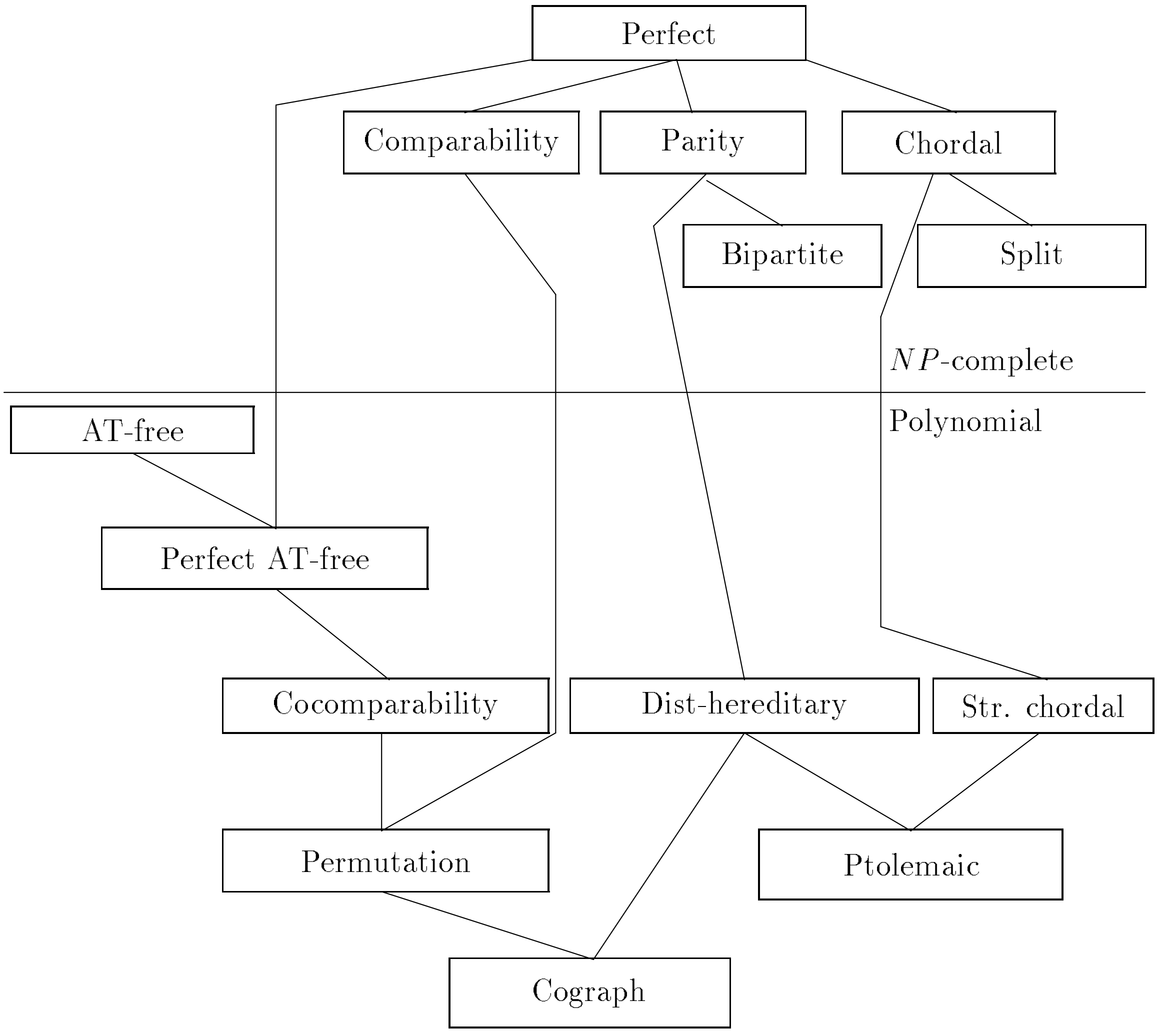

Figure 6.1: The status of the connected domination and Steiner set problems on some classes of graphs.

Bibliography

[1] A.V. Aho, J.E. Hopcroft, and J.D. Ullman. *The Design and Analysis of Computer Algorithms.* Addison Wesley, Reading, Mass., 1976.

[2] K. Arvind and C. Pandu Rangan. Connected domination and Steiner set on weighted permutation graphs. *Info. Proc. Letts.*, 41:215–220, 1992.

[3] Hari Balakrishnan, Anand Rajaraman, and C. Pandu Rangan. Connected domination and Steiner set on asteroidal triple-free graphs. Technical Report TR-TCS-93-01, Dept. of Computer Science and Engineering, Indian Institute of Technology, Madras, India, 1993. To appear in the Proceedings, WADS '93.

[4] H.J. Bandelt and H.M. Mulder. Distance-hereditary graphs. *J. Comb. Theory. Series B*, pages 182–208, 1986.

[5] C. Berge. *Graph Theory and its Applications.* John Wiley, New York, 1962.

[6] C. Berge. *Graphs and Hypergraphs.* North-Holland, Amsterdam, 1973.

[7] J.A. Bondy and U.S.R. Murty. *Graph Theory with Applications.* Elsevier, 1976.

[8] C.J. Colburn and L.K. Stewart. Permutation graphs: Connected domination and Steiner trees. *Discrete Math.*, 86:145–164, 1990.

[9] D.G. Corneil, S. Olariu, and L. Stewart. Asteroidal triple-free graphs. Technical Report 262/92, Univ. of Toronto, June 1992.

[10] D.G. Corneil and L.K. Stewart. Dominating sets in perfect graphs. *Discrete Math. (Special issue on Topics in Domination)*, 86(1–3):179–189, 1990.

[11] A. D'Atri and M. Mascarini. Distance hereditary graphs, Steiner trees and connected domination. *SIAM J. Computing*, 17:521–538, 1988.

[12] I. Degan, M.C. Golumbic, and R.Y. Pinter. Trapezoid graphs and their coloring. *Disc. Appl. Math.*, 21:35–46, 1988.

[13] N. Deo. *Graph Theory with Applications to Engineering and Computer Science*. Prentice-Hall, Englewood Cliffs, 1974.

[14] E.W. Dijkstra. A note on two problems in connexion with graphs. *Numer. Math.*, 1:269–271, 1959.

[15] S. Even, A. Pnueli, and A. Lempel. Permutation graphs and transitive graphs. *Journal of the ACM*, 19:400–410, 1972.

[16] P. Fermat. Maxima et minima IV : Methode du maximum et minimum. In P. Tannery and C. Henry, editors, *Oeuvres de Fermat*, pages 131–136. Gauthier-Villars et Fils, Paris, 1896.

[17] M.L Fredman and R.E. Tarjan. Fibonacci heaps and their uses in improved network optimization algorithms. *Journal of the ACM*, 34:596–615, 1987.

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

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

[20] F. Harary. *Graph Theory.* Addison-Wesley, Reading, Mass., 1969.

[21] S.T. Hedetniemi and R. Laskar. Special issue on topics in domination. *Discrete Math.*, 86(1–3), 1990.

[22] E. Howorka. A characterization of distance-hereditary graphs. *Quart. J. Math. Oxford Series 2*, pages 417–420, 1977.

[23] R.M. Karp. Reducibility among combinatorial problems. In R.E. Miller and J.W. Thatcher, editors, *Complexity of computer computations*, pages 85–103. Plenum Press, New York, 1972. IBM Res. Symp. Ser., Vol. 4.

[24] B. Korte, H.J. Promel, and A. Steger. Steiner trees in VLSI-layout. In et al B. Korte, editor, *Paths, flows, and VLSI-layout*, pages 195–214. Springer-Verlag, Berlin, 1990.

[25] C.G. Lekkerkerker and J.C. Boland. Representation of a finite graph by a set of intervals on a real line. *Fundamenta Mathematicae*, 51:45–64, 1962.

[26] U. Manber. *Introduction to Algorithms – A Creative Approach.* Addison-Wesley, Reading, Mass., 1989.

[27] Anand Rajaraman, Hari Balakrishnan, and C. Pandu Rangan. Modular decomposition techniques for distance-hereditary graphs. Technical Report TR-TCS-93-03, Dept. of Computer Science and Engineering, Indian Institute of Technology, Madras, India, 1993.

[28] K. White, M. Farber, and W.R. Pulleybank. Steiner trees, connected domination and strongly chordal graphs. *Networks*, 15:109–124, 1985.