UNIVERSIDAD DE BUENOS AIRES

Facultad de Ciencias Exactas y Naturales

Departamento de Matemática

Tesis de Licenciatura

Structural and Algorithmic Results on Neighborhood-Perfect Graphs and Neighborhood Numbers

Xavier S. Warnes

Director: Martín D. Safe
Co-director: Guillermo Durán

Noviembre 2014

Resumen

Un grafo es *vecindad-perfecto* si en cada subgrafo inducido, el mínimo número de vecindades cerradas necesarias para cubrir los vértices y aristas es igual al máximo cardinal de un conjunto de vértices y aristas sin dos elementos pertenecientes a la misma vecidad cerrada. A diferencia de los grafos perfectos, los grafos vecindad-perfectos no han sido aún caracterizados por subgrafos inducidos prohibidos, ni tampoco se conoce la complejidad algorítmica del problema de reconocimiento de la clase. En esta tesis probamos caracterizaciones de los grafos vecindad-perfectos por subgrafos inducidos prohibidos minimales restringidas a ciertas clases de grafos, incluyendo la clase de los grafos P_4-tidy, la de los tree-cographs y ciertas clases relacionadas a los grafos clique-Helly hereditarios. Por otro lado consideramos el problema de reconocimiento de los grafos vecindad-perfectos y encontramos algoritmos polinomiales para resolverlo restringido a distintas clases de grafos. Finalmente mostramos dos algoritmos lineales para encontrar los parámetros involucrados en la definición de vecindad-perfección (y los conjuntos óptimos que realizan los parámetros) restringiendo la entrada a grafos pertenececients a la clases de los grafos P_4-tidy y los tree-cographs, y probamos que el problema de determinar estos mismos parámetros es $\mathcal{NP}$-completo aún para grafos complemento de bipartito.

Abstract

A graph is *neighborhood-perfect* if for every induced subgraph, the minimum number of closed neighborhoods needed to cover all the vertices and edges equals the maximum size of a set of vertices and edges, no two of which belong to the same closed neighborhood. Unlike perfect graphs, neither a forbidden induced subgraph characterization, nor the computational complexity of the recognition problem are known for the whole class of neighborhood-perfect graphs. In this thesis we give characterizations of neighborhood-perfect graphs by minimal forbidden induced subgraphs restricted to several graph classes, including the classes of the P_4-tidy graphs, tree-cographs and several graph classes related to the class of hereditary clique-Helly graphs. Moreover we consider the problem of recognizing neighborhood-perfect graphs and propose polynomial-time algorithms to solve it restricted to different graph classes. Finally we present two linear-time algorithms to find the parameters involved in the definition of neighborhood-perfectness for P_4-tidy graphs and tree-cographs and prove that the problems of these same parameters are $\mathcal{NP}$-complete when restricted to complement of bipartite graphs.

Agradecimientos

A Martín y Willy, por el trabajo y dedicación que le pusieron a este trabajo. A Willy por darme un lugar para trabajar con tan buen grupo de investigadores y siempre darme el apoyo necesario para poder aprender y crecer académicamente. A Martín, porque sin él esta tesis no hubiese sido posible, por sus correcciones precisas y puntillosas que tanto me enseñaron y por dedicar el tiempo que no tenía a ayudarme.

A Flavia, por mostrarme el mundo de la teoría de grafos y siempre estar dispuesta a darme una mano. A Oscar, porque sin sus clases de algoritmos en grafos nunca hubiese podido escribir el cuarto capítulo de la tesis. A Marina y a Diego, por todos sus consejos a lo largo de estos años.

A mis compañeros, a Eugenio, Agustín, Pedro, Fede, Pablo, Santi V, Santi D, por hacer que muchas materias fuesen mucho más placenteras de cursar. A Flor y Vero, por la buena onda que le ponen a toda la oficina.

A mis padres, si no fuese por su apoyo constante e inagotable no hubiese podido terminar esta carrera. A mi padre, por enseñarme matemática desde siempre. A mi hermano Pablo, por parecerse a mí.

A Sabri, por aguantarme durante toda la carrera, pese a todos esos fines de semanas en que todo lo que hacíamos era estudiar.

“Wahrlich es ist nicht das Wissen, sondern das Lernen, nicht das Besitzen sondern das Erwerben, nicht das “Da-sein”, sondern das Hinkommen, was den grössten Genuss gewährt.”

Carl Friedrich Gauß

Contents

Nomenclature

$\alpha_c(G)$	clique-independence number, 13
$\alpha(G)$	independence number, 11
$\alpha_2(G)$	2-independence number, 14
$\alpha_k(G)$	k-independence number, 14
$\alpha_n(G,k)$	k-neighborhood-independence number, 16
$\alpha_n(G)$	neighborhood-independence number, 12
$\beta(G)$	vertex-cover number, 15
$\chi(G)$	chromatic number, 11
$\Delta(G)$	maximum degree of vertices of G, 8
$\delta(G)$	minimum degree of vertices of G, 8
$\gamma(G)$	domination-number, 14
$\mathcal{P}$	class of polynomial-time solvable problems, 28
$\mathcal{NP}$	non-deterministic polynomial problems, 28
$\omega(G)$	maximum clique, 11
$\pi(G)$	set of all $\pi(h)$, for every $h \in T(G)$, 24

$\pi(h)$	graph representing the adjacencies among the children of $h \in T(G)$, *23*
$\rho_n(G)$	neighborhood-covering number, *12*
$\rho_n(G, k)$	k-neighborhood-covering number, *16*
$\tau(G)$	size of a minimum vertex cover of G, *10*
$\tau_c(G)$	clique-transversal number, *13*
C_n	chordless n-cycle, *9*
$d_G(v)$	degree of v in G, *8*
$G[h]$	$G[M(h)]$ for $h \in T(G)$, *23*
$G[W]$	subgraph of G induced by W, *8*
h_R	child representing the head of a starfish or urchin in N-node, *26*
K_n	complete graph of n nodes, *8*
m	number of edges, *8*
$M(h)$	module corresponding to $h \in T(G)$, *23*
n	number of vertices, *8*
$N_G(v)$	open neighborhood of v in G, *8*
$N_G[v]$	closed neighborhood of v in G, *8*
P_n	chordless n-path, *9*
$T(G)$	modular decomposition tree, *22*

Chapter 1

Introduction

One of the most celebrated results in the last fifteen years in Graph Theory is without a doubt the characterization by forbidden induced subgraphs of the class of perfect graphs. A graph G is said to be *perfect* if $\chi(G') = \omega(G')$, for all induced subgraph G' of G[1], where $\chi(G)$ is the minimum number of colors needed to color the vertices of graph G and $\omega(G)$ the size of the largest clique of G. Perfect graphs were defined by Berge in 1961 [4]. Not only are perfect graphs of great theoretical importance, but they have as well significant importance in real-world applications. They arise naturally in many of these, for example optimization of computer storage, analysis of genetic structure and scheduling problems [54, 98]. In his initial work, Berge proposed two conjectures that were later known as the *Perfect Graph Theorem* and the *Strong Perfect Graph Theorem*. The first of these stated that a graph is perfect if and only if its complement is perfect[2], and was proved to be true by Lovász in 1972 [74, 75]. Although Lovasz admited that much credit should be given to Fulkerson, who had previously established the relation between perfect graphs

[1]We shall assume, in this chapter, basic knowledge of Graph Theory. In case of doubt we refer the reader to Chapter 2.

[2]The original conjecture used the terms χ-perfect and α-perfect, where G is α-perfect if $\alpha(G') = \theta(G')$ for all induced subgraphs G' of G, $\alpha(H)$ being the independence number and $\theta(H)$ being the minimum clique cover of the vertices of a graph H.

and polyhedral combinatorics, which led him to an independent proof of the theorem [49]. The second conjecture was a characterization of perfect graphs as those graphs such that no induced subgraph is an odd chordless cycle of length at least 5 or a complement of such an odd cycle. Graphs containing no such induced cycles or their complements were later on called *Berge graphs*. An interesting report of the history and motivations of these conjectures can be found in [6]. The Strong Perfect Graph Theorem was proved by Chudnovsky, Robertson, Seymour and Thomas in 2002 [32], using a structural theorem, in which they proved that a Berge graph belongs to one of five basic classes or admits one of a few kinds of decompositions. An account of how this final proof was found was written in 2006 by Seymour [92].

The class of *neighborhood-perfect graphs* was defined in 1986 by Lehel and Tuza [70] based on a min-max equality similar to that of perfect graphs. Given a graph G, a set $C \subseteq V(G)$ is a *neighborhood-covering set* if each edge and vertex of G belongs to $G[v]$ for some $v \in C$, where $G[v]$ is the subgraph of G induced by the closed neighborhood of the vertex v [3]. This concept of neighborhood-covering was first introduced by Sampathkumar and Neeralagi [89]. We shall say that two elements of $E(G) \cup V(G)$ are *neighborhood-independent* if there is no vertex $v \in V(G)$ such that both elements belong to $G[v]$. A set $S \subseteq V(G) \cup E(G)$ is said to be a *neighborhood-independent* set if every pair of elements of S is neighborhood-independent[4]. Let $\rho_n(G)$ be the size of the minimum neighborhood-covering set and $\alpha_n(G)$ of the maximum neighborhood-independent set. Clearly, the inequality

$$\rho_n(G) \geqslant \alpha_n(G)$$

holds for every graph G. Thus, Lehel and Tuza define neighborhood-perfect graphs as those graphs G where $\rho_n(G') = \alpha_n(G')$ holds for every induced subgraph G' of G. They observe as well that as a consequence of the Strong

[3] Note that this definition is slightly different from the one given in [70], where isolated vertices are not required to be covered.

[4] As for neighborhood-covering sets, neighborhood-independent sets are defined in [70] as sets of edges, without taking into account isolated vertices.

Perfect Graph Theorem, all neighborhood-perfect graphs must be perfect. In Figure 1.1 we can see two graphs that will appear throughout the thesis, one is neighborhood-perfect and the other is not.

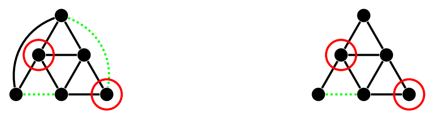

Figure 1.1: Example of a neighborhood-perfect graph (left) and a non neighborhood-perfect graph (right). For each graph, a maximum neighborhood-independent set is marked by dotted (green) edges and a minimum neighborhood-covering set by circling the vertices (in red).

No forbidden induced subgraph characterization of the whole class of neighborhood-perfect graphs is known, but several partial characterizations have been proven. In the paper where Lehel and Tuza defined the class, they proved that, when restricted to chordal graphs, neighborhood-perfect graphs are exactly odd-sun-free graphs. Lehel and Tuza showed as well that triangle-free graphs are neighborhood-perfect if and only if they are bipartite. In [57] minimally non-neighborhood-perfect graphs were defined and all those minimally non-neighborhood-perfect graphs G satisfying $\alpha_n(G) = 1$ were found. Moreover, in this same work, they proved that all line graphs of bipartite graphs are neighborhood-perfect and found a forbidden induced subgraph characterization for neighborhood-perfect graphs in the class of chordal graphs. As a generalization of the result on line graphs of bipartite graphs, Lehel proved a characterization of those line graphs that are neighborhood-perfect [69].

Several results have been published on the algorithmic problems of finding $\alpha_n(G)$ and $\rho_n(G)$ for different graph classes. If G is a cograph, interval graph or a strongly chordal graph, then this problems can be solved in linear time [57, 23, 70]. If G is a neighborhood-perfect chordal graph, then finding the parameters can be done in polynomial-time [70]. On the other hand it has been seen that the same problems are $\mathcal{NP}$-complete for line graphs, planar graphs and split graphs [56, 27].

Although by many previously published results it is clear that the problem of recognizing neighborhood-perfect graphs is polynomial in several graph classes, the computational complexity of the recognition problem of the whole class is still unknown. Gyárfás et al. [57] described an algorithm to find $\alpha_n(G)$ and $\rho_n(G)$ of any neighborhood-perfect cograph G, which can easily be modified to a linear-time recognition algorithm of neighborhood-perfect cographs. Moreover by results of [70], a polynomial-time recognition algorithm for chordal neighborhood-perfect graphs follows. Finally the fact, implicitly proven in [69], that neighborhood-perfectness coincides with clique-perfectness when restricted to the class of hereditary clique-Helly graphs, implies that polynomial-time recognition algorithms for neighborhood-perfectness within hereditary clique-Helly graph classes can be derived from analogous existing results for clique-perfectness. These results are stated explicitly in Section 4.2.

This thesis is divided into three main chapters. In Chapter 2 we give the basic definitions and known results used in the rest of the thesis. In Chapters 3 and 4 we present the main results of the thesis; each of these chapters begins with a summary of previously proven results on the subject.

In Chapter 3 we first prove some structural results. For instance, we prove that if a graph G is the join of two neighborhood-perfect graphs F and H, then G is neighborhood-perfect if and only if it contains no induced $C_4 \vee 2K_1$, $C_6 \vee 3K_1$ or $P_6 \vee 3K_1$, where $\vee$ denotes the join operation. To reach this conclusion, we begin by showing how to obtain $\alpha_n(G)$ and $\rho_n(G)$ from the dominating number, the 2-independence number and the neighborhood-covering number of F and H. Using the structural results proved in Section 3.1, in the next two sections of Chapter 3 we prove forbidden induced subgraph characterizations of neighborhood-perfect graphs restricted to P_4-tidy graphs and tree-cographs, respectively. In the last section of this chapter we prove forbidden induced subgraph characterizations of neighborhood-perfect graphs restricted to several other graph classes, namely Helly circular-arc graphs, gem-free circular-arc graphs, diamond-free graphs, hereditary clique-Helly claw-free graphs and paw-free graphs. Most of these results are direct corollaries of the fact implicit in [69] that, if a graph is hereditary clique-Helly, then it is clique-perfect if and only if it is neighborhood-perfect.

In Chapter 4, we give linear-time algorithms that recognize neighborhood-perfect graphs within the classes of P_4-tidy graphs and tree-cographs. We present as well two linear-time algorithms that find a minimum neighborhood-covering set and a maximum neighborhood-independent set for any P_4-tidy graph or tree-cograph. Furthermore, based on the characterizations that we prove in the last section of Chapter 3, we prove that the recognition problem of neighborhood-perfect graphs can be solved in polynomial time when the input graph is restricted to belong to certain graph classes, namely paw-free graphs, diamond-free graphs, claw-free hereditary clique-Helly graphs and Helly circular-arc graphs. Finally we finish the chapter by proving that the problems of determining $\alpha_n(G)$ and $\rho_n(G)$ are $\mathcal{NP}$-complete even if G is restricted to be the complement of a bipartite graph.

Chapter 5 contains a summary of the main results of the thesis and some further remarks.

Chapter 2

Preliminaries

In this chapter we shall define the basic concepts and give the basic results that shall be used throughout the thesis. References are given in each section to works where these concepts are explained in full detail.

2.1 Basic Definitions and Notations

For the sake of clarity we shall follow when possible West's [104] notations. In his book he defines a *graph* G as a triple, consisting of a vertex set $V(G)$, an edge set $E(G)$ and a relation that assigns two vertices, called the *endpoints*, to each edge. We will only deal with *simple* graphs, that is, graphs that are finite, undirected and have no loops nor multiple edges. Thus we can view an edge as an unordered pair of vertices, ignoring the formality of the relation associating vertices and edges. Hence, if v and w are vertices of G, we shall denote the edge joining them as vw. The *complement* of a graph G is a graph $\overline{G}$ such that $V(G) = V(\overline{G})$ and for every two different $v, w \in V(\overline{G})$, $vw \in E(\overline{G})$ if and only if $vw \notin E(G)$. An *edge-vertex incidence* matrix of G is a $\{0, 1\}$-matrix having one column for each vertex and one row for each edge, such that for each row, the only two $1'$s that appear correspond to the endpoints of the associated edge . We shall say that a graph H is a *subgraph* of G if $V(H) \subseteq V(G)$ and $E(H) \subseteq E(G)$.

Given a set $W \subseteq V(G)$, we shall say that $G[W]$ is the subgraph of G *induced* by W when $V(G[W]) = W$ and $E(G[W]) = \{vw \in E(G) \colon v, w \in W\}$. The graph H shall be called an *induced subgraph* of G if there is a set W such that $H = G[W]$; moreover, if $W \neq V(G)$, then H shall be called a *proper induced subgraph*. We shall denote the subgraph of G induced by $V(G) \setminus W$ by $G - W$; if $W = \{v\}$, simply by $G - v$. For any set S, $|S|$ denotes the cardinality of S. For simplicity, we shall note the number of vertices of a G as $n_G = |V(G)|$, and the number of edges as $m_G = |E(G)|$. When clear by context to which graph we are referring, we shall simply denote its number of vertices by n and its number of edges by m.

The *neighborhood* of a vertex v in G is the set of vertices of G adjacent to v, and is denoted by $N_G(v)$. The *closed neighborhood* of v in G is $N_G(v) \cup \{v\}$ and is noted $N_G[v]$. When the graph G is clear by context, the neighborhood and the closed neighborhood shall be denoted by $N(v)$ and $N[v]$, respectively. A vertex v is called *universal* if $N[v] = V(G)$ (that is, v is adjacent to all other vertices of G), *pendant* if it is adjacent to exactly one vertex of G, *isolated* if it is adjacent to none, and *simplicial* if its neighborhood is a clique. An edge is *pendant* if it has at least one pendant endpoint. The *common neighborhood* of an edge $e = vw$ is $N(e) = N(v) \cap N(w)$; in general the *common neighborhood* of a nonempty set of vertices S is $N(S) = \bigcap_{s \in S} N(s)$. The *degree* of a vertex v is $d_G(v) = |N_G(v)|$. The *maximum degree* of the vertices of G is noted by $\Delta(G)$ and the *minimum degree* by $\delta(G)$.

A graph is *complete* if its vertices are pairwise adjacent, we shall denote the complete graph on n vertices by K_n. A complete graph of 3 vertices is called a *triangle*. A *clique* of a graph is a set of pairwise adjacent vertices and a *maximal clique* of a graph is an inclusion-wise maximal clique. A *clique-matrix* of a graph is a $\{0, 1\}$-matrix, with one row for each maximal clique and one column for each vertex and such that there is a 1 in a given entry if and only if the vertex corresponding to the column belongs to the clique corresponding to the row. A *stable set* of a graph is a set of pairwise nonadjacent vertices. Clearly a stable set of G is a clique of $\overline{G}$ and vice versa.

A *walk* in G is a sequence of vertices $v_1, v_2 \ldots, v_k$, such that for each $i \in \{1, \ldots, k-1\}$, $v_i v_{i+1} \in E(G)$. If $v_1 = v_k$, the walk is called *closed*. A walk where no vertex is repeated is called a *path*. A closed walk where no vertex except for the last and first ones is repeated is called a *cycle*. We shall consider the smallest possible cycle as a cycle of 3 vertices, which we shall call as well a *triangle*. The first and last vertex of a path shall be called its *endpoints*. An *n-path* (resp. *n-cycle*) is a path (resp. cycle) with n vertices. Let Z be a path or cycle of G. We shall denote the set of vertices of Z by $V(Z)$, and the set of edges of Z by $E(Z)$. The *length* of Z will be $|E(Z)|$. The *distance* in G between two vertices is the minimum length of a path having them as endpoints. A *chord* in Z is an edge joining two nonconsecutive vertices of Z and Z is *chordless* if it has no chords. The chordless n-path and n-cycle are denoted by P_n and C_n, respectively. For each $n \geqslant 4$, W_n denotes the *wheel* graph on n vertices, which arises form C_n by adding a universal vertex. A cycle is *odd* if it has a odd number of vertices, and *even* otherwise. A *hole* is a chordless cycle of length at least 4 and an *antihole* is the complement of a hole of length at least 5. A *k-sun* (or *trampoline* of order k) is a graph having $2k$ vertices $v_1, \ldots, v_k, w_1, \ldots, w_k$ such that $v_1 \ldots v_k$ is a cycle and every w_i ($1 \leqslant i \leqslant k$) has exactly two neighbors: v_i and v_{i+1} ($v_{k+1} = v_1$). A k-sun is said to be *odd* if k is odd.

A graph G is *connected* if for every two vertices in it, there is a path that joins them. A *component* of a graph is a containment-wise maximal connected subgraph. An *anticomponent* of G is the subgraph in G induced by the vertices of a component of $\overline{G}$. A component is *trivial* if it has only one vertex. A connected graph without cycles is called a *tree* . The complement of a tree is called a *co-tree* . A graph is called a *forest* if all its components are trees. A *cutpoint* is a vertex v such that $G - v$ has more components than G. An edge e is a *bridge* if $G - e$ has more components that G.

A *dominating* set of a graph G is a set $A \subseteq V(G)$ such that each $v \in V(G)$ either is in A or is adjacent to some vertex of A. A *total dominating* set is a set $A \subseteq V(G)$ such that every vertex in $V(G)$ is adjacent to at least one vertex of A. A k-*independent* set of G is a set of vertices such that for every pair of vertices in it there is no path of length k or less that connects them in G. A set $A \subseteq V(G)$ is called a *vertex cover* of G if every edge $e \in E(G)$ has at least one endpoint that belongs to A. We shall note by $\tau(G)$ the size of a vertex cover of a graph G with minimum number of nodes.

If G and H are two graphs, then we shall say that G *contains* H if H is isomorphic to a subgraph (not necessarily induced) of G; if H is isomorphic to an induced subgraph of G, we say it is G *contains an induced* H. A class $\mathcal{G}$ of graphs is called *hereditary* if, for every graph $G \in \mathcal{G}$, all induced subgraphs of G belong to $\mathcal{G}$. We say that G is H-*free* if G contains no induced subgraph isomorphic to H. Given a collection of graphs $\mathcal{H}$, we say that G is $\mathcal{H}$-free if G does not contain any induced graph $H \in \mathcal{H}$. A graph H is a *forbidden induced subgraph* for a class $\mathcal{G}$ if every graph of $\mathcal{G}$ is H-free. Moreover if $\mathcal{G}$ is a hereditary class, H is said to be a *minimal forbidden induced subgraph* for $\mathcal{G}$ if H does not belong to $\mathcal{G}$, but all of its proper induced subgraphs do.

Given two graphs G and H, such that $V(G) \cap V(H) = \emptyset$, we shall define the *join* of G and H as the graph $G \vee H$ having set $V(G \vee H) = V(G) \cup V(H)$ and edge set $E(G \vee H) = E(G) \cup E(H) \cup \{vw \colon v \in V(G), w \in V(H)\}$.

A graph is *bipartite* if its vertex set can be partitioned in two stable sets X and Y. If so, $\{X, Y\}$ is a *bipartition* of the graph. Moreover if every vertex of X is adjacent to every vertex of Y, the graph is called *complete bipartite*. The complement of a bipartite graph is called a *co-bipartite* graph.

A *matching* of a graph G is a set of vertex-disjoint edges of G. If M is a matching, M is *maximal* if it is inclusion-wise maximal and *maximum* if it is of maximum size. An *induced matching* is a matching M where no two edges are joined by a common edge. Equivalently a matching M is induced if the subgraph induced by its vertices has exactly M as its edge set. Induced matchings were first defined by Cameron in [26], where she proves as well that the problem of finding maximum induced matchings in bipartite graphs is $\mathcal{NP}$-complete.

Some of the graphs that we will refer to in the thesis are depicted in Figure 2.1.

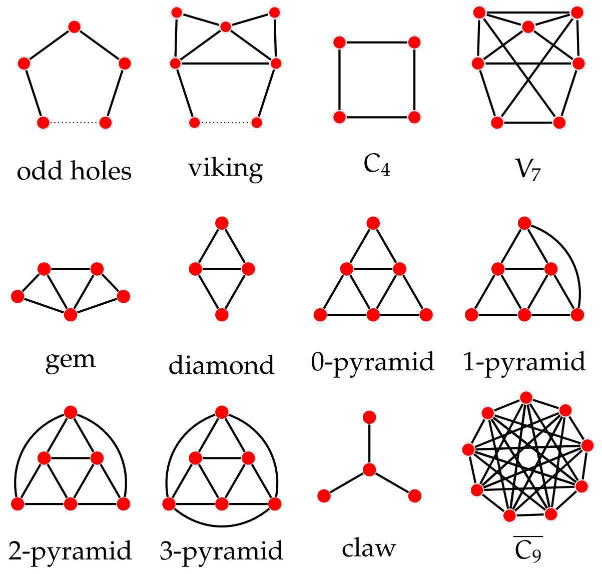

Figure 2.1: Some special graphs.

For any definition not presented here, we refer the reader to the excellent book by West [104].

2.2 Important Parameters

In this section we shall present the most important graph parameters that we shall use in this work. They shall all take the form of an operator that applied to a simple graph gives a positive integer.

Let us first consider four parameters that were already defined in Chapter 1 and play a crucial role in the definition of perfect graphs. Given a simple graph G, we shall define the *independence number* as the maximum size of a stable set in G, and we shall denote it by $\alpha(G)$. Analogously we shall denote the maximum number of vertices of a clique of G as $\omega(G)$. The *chromatic number*, $\chi(G)$, is the minimum number of colors needed to color the vertices of G such that no two adjacent vertices share the same color. Note that this is equivalent to the minimum partition of $V(G)$ into stable sets. Following the same relationship between stable sets and cliques used in the previous two parameters, we shall define $\theta(G)$ as the minimum number of cliques needed to cover all of the vertices of G. Note that $\theta(G) = \chi(\overline{G})$ and $\omega(G) = \alpha(\overline{G})$. Moreover, in every valid coloring of G, a clique must have all of its vertices colored with different colors, and clearly in a every cover of $V(G)$ by cliques, every element of a stable set must be covered by a different clique. These remarks imply the following two inequalities:

$$\omega(G) \leqslant \chi(G), \tag{2.1}$$

$$\alpha(G) \leqslant \theta(G). \tag{2.2}$$

The problem of computing any of these four parameters is $\mathcal{NP}$-complete for general graphs [67].

The two most important parameters that appear in this thesis are the *neighborhood-independence number* and the *neighborhood-covering number* . The idea of neighborhood-covering was first introduced by Sampathkumar and Neeralagi [89] and was studied in [27, 65, 70]. A *neighborhood-covering set* or simply *neighborhood set* of a graph G is a subset of $V(G)$ such that every edge and vertex of G is covered by some vertex of the set. We say that a vertex v *covers* a vertex or edge if it belongs to $G[N[v]]$. A *neighborhood-independent set* is a set of vertices and edges such that no two of its elements are covered by a common vertex in G. The *neighborhood-covering number* is defined as the minimum size of a neighborhood-covering set, and is denoted by $\rho_n(G)$. The *neighborhood-independence number* is defined as the maximum size of a neighborhood-independent set, and is denoted by $\alpha_n(G)$. It is worth noting that this definitions are not exactly the same as those used in [70, 27], where isolated vertices are not required to be covered and neighborhood-independent sets contain only edges. However, it is clear that the only difference between the two pairs of definitions is that our definitions take into account the isolated vertices. Namely, each of the parameters $\alpha_n(G)$ and $\rho_n(G)$ as defined here arises from the homonymous operator defined in [70] by adding up the number of isolated vertices of G. Thus, all equalities and inequalities between $\alpha_n(G)$ and $\rho_n(G)$ proved for the variants not taking into account the isolated vertices still hold. As different elements of a neighborhood-independent set must be covered by different vertices of any neighborhodd-covering set, the following inequality must hold for every graph G:

$$\alpha_n(G) \leqslant \rho_n(G). \tag{2.3}$$

It is $\mathcal{NP}$-complete to determine $\alpha_n(G)$ and $\rho_n(G)$ even for split graphs G with certain degree constraints [27], but there are linear-time algorithms for finding them when restricted to interval graphs [70] and, more generally, strongly chordal graphs [23]. These problems have been generalized to a k-distance version [65]
A *clique-transversal* of a graph G is a set T of vertices such that for every maximal clique C of G, $C \cap T \neq \emptyset$; that is, every maximal clique of G has at least one vertex in T. Two maximal cliques are said to be *independent* if they share no common vertex. The minimum size of a clique-transversal of G is called the *clique-transversal number*, and is noted $\tau_c(G)$. The maximum number of pairwise independent maximal cliques on G is called the *clique-independence number* and is denoted by $\alpha_c(G)$. Once again, it is clear by the definition of these parameters that for every every two independent cliques we shall need two different vertices in the clique-transversal. Thus the following inequality must hold for every graph G:

$$\alpha_c(G) \leqslant \tau_c(G). \tag{2.4}$$

These two parameters have been studied in [102, 1, 44, 27, 56, 43]. It was proven that determining $\alpha_c(G)$ and $\tau_c(G)$ is $\mathcal{NP}$-complete even for split graphs [27], as well as for planar graphs, line graphs, cocomparability graphs and total graphs [56]. There are linear-time algorithms for finding these parameters in strongly chordal graphs (given a strong elimination order) [27], and polynomial-time algorithms in the class of Helly circular-arc graphs [56]. These parameters have been generalized in [28] as follows. Suppose that each maximal clique C_i is associated with an integer r_i, where $0 \leqslant r_i \leqslant |C_i|$. A *generalized clique-transversal set* is a set of vertices D such that for every C_i, $|C_i \cap D| \geqslant r_i$. The *generalized* *clique-transversal number* is the minimum size of generalized clique-transversal set, whereas the *generalized clique-independence number* is the maximum of $\sum_{C_i \in \mathcal{P}} r_i$, among the collections $\mathcal{P}$ of pairwise disjoint maximal cliques of G.

The last pair of parameters we shall present is the *domination number* and the k-*independence* number. The concept of domination arises naturally in many location problems in operation research and has been extensively studied. We have already defined dominating sets, as well as k-independent sets in the previous section. The *domination number* $\gamma(G)$ is the minimum size of a dominating set of G, and the k-*independence number* $\alpha_k(G)$ is the maximum size of a k-independent set of G. It is easy to see that the following inequality holds for every graph G:

$$\alpha_2(G) \leqslant \gamma(G). \tag{2.5}$$

There have been many results regarding domination and independence problems (see [62, 29]). Determining $\gamma(G)$ is $\mathcal{NP}$-complete in general graphs [51] and determining $\alpha_2(G)$ is $\mathcal{NP}$-complete even for split graphs [27].

Besides the inequalities (2.3), (2.4) and (2.5), the six previously defined parameters are related by the following two inequalities that hold for every graph G:

$$\gamma(G) \leqslant \rho_n(G) \leqslant \tau_c(G) \quad \text{and} \quad \alpha_2(G) \leqslant \alpha_n(G) \leqslant \alpha_c(G). \tag{2.6}$$

The first inequality follows form the fact that every neighborhood-covering set is a dominating set. The second inequality is a consequence of the fact that every clique-transversal set is a neighborhood-covering set, because every edge and vertex belongs to at least one maximal clique and thus is covered by at least one vertex of any clique-transversal. The third inequality follows from the fact that every 2-independent set is a neighborhood-independent set. The last inequality follows from the fact that replacing each edge or vertex of a neighborhood-independent set by a maximal clique containing it must result in a clique-independent set.

Nevertheless it is worth noting that all the preceding inequalities can be all strict.

An example of a graph where all inequalities are strict can be seen in Figure 2.2.

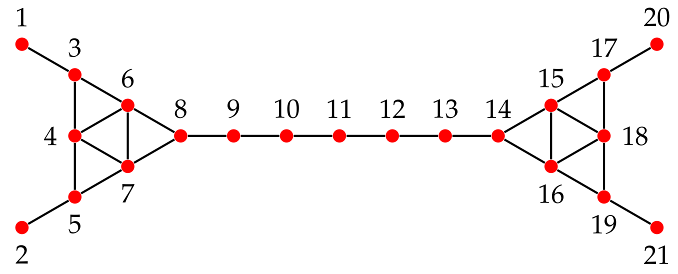

Figure 2.2: A chordal graph where the inequalities of (2.6) *are strict. This figure appears in* [28].

For G the graph in Figure 2.2, the following are the values of the aforementioned parameters, with D^* and S^* the corresponding optimum sets, this example is taken from [28].

$$\begin{aligned}
\gamma(G) &= 7, \quad & D^* &= \{3,5,8,11,14,17,19\} \\
\rho_n(G) &= 8, \quad & D^* &= \{3,5,8,10,12,14,17,19\} \\
\tau_c(G) &= 9, \quad & D^* &= \{3,5,7,9,11,13,16,17,19\} \\
\alpha_2(G) &= 7, \quad & S^* &= \{1,2,8,11,14,20,21\} \\
\alpha_n(G) &= 8, \quad & S^* &= \{(1,3),(2,5),(6,7),(9,10),(12,13),(15,16),(17,20), \\
& & & \quad (19,21)\} \\
\alpha_c(G) &= 9, \quad & S^* &= \{(1,3),(2,5),(4,6,7),(8,9),(10,11),(13,14),(15,16,18), \\
& & & \quad (17,20),(19,21)\}.
\end{aligned}$$

We will, as well, define the *vertex-cover number* as the size of the minimum vertex cover set of G. We shall denote the vertex cover number of G by $\beta(G)$. The problem of computing this parameter was proven to be $\mathcal{NP}$-complete for general graphs [67].

Finally, it is worth mentioning that a generalization of the neighborhood-covering number and the neighborhood-independence number was defined in [65]. A vertex z is said to *k-neighborhood-cover* an edge xy if $d_G(z, x) \leqslant k$ and $d_G(z, y) \leqslant k$, that is, z must k-dominate both points of the edge. A *k-neighborhood-covering set* of a graph G is a set of vertices that k-neighborhood-covers all edges of the graph. The *k-neighborhood-covering number* $\rho_n(G, k)$ is the minimum cardinality of a k-neighborhood-covering set[1]. An edge set $I \subseteq E(G)$ is a *k-neighborhood-independent set* if no pair of distinct edges are k-neighborhood-covered by a same vertex of $V(G)$. The *k-neighborhood-independence number* $\alpha_n(G, k)$ of a graph G is the maximum cardinality of a k-neighborhood-independent set. Clearly for any graph G and any positive integer k, the following inequality must hold:

$$\alpha_n(G, k) \leqslant \rho_n(G, k)$$

.

2.3 Some Special Graph Classes

In this section we shall present some important graph classes that will be used later on. Some of these definitions have already been given in Chapter 1. In Figure 2.3 we can see the relationships between most of the graph classes considered in this section.

2.3.1 Perfect Graphs

In the 1960's, Berge defined the class of perfect graphs by means of a min-max type equality [4]. We have shown in inequality (2.1), that for any graph G, the chromatic number is always at least the size of the maximum complete subgraph. Berge defined *χ-perfect* graphs as those graphs G for which $\chi(G') = \omega(G')$ for all induced subgraphs G' of G. He defined as well *α-perfect graphs* as those where inequality (2.2) holds with equality for all induced subgraphs; that is, $\alpha(G') = \theta(G')$ holds for all induced subgraphs G'. Clearly a graph is χ-perfect if

[1]Note that this definition can be easily extended to include isolated vertices and hence leads to a generalization of the neighborhood-independence and neighborhood-covering parameters used in this thesis.

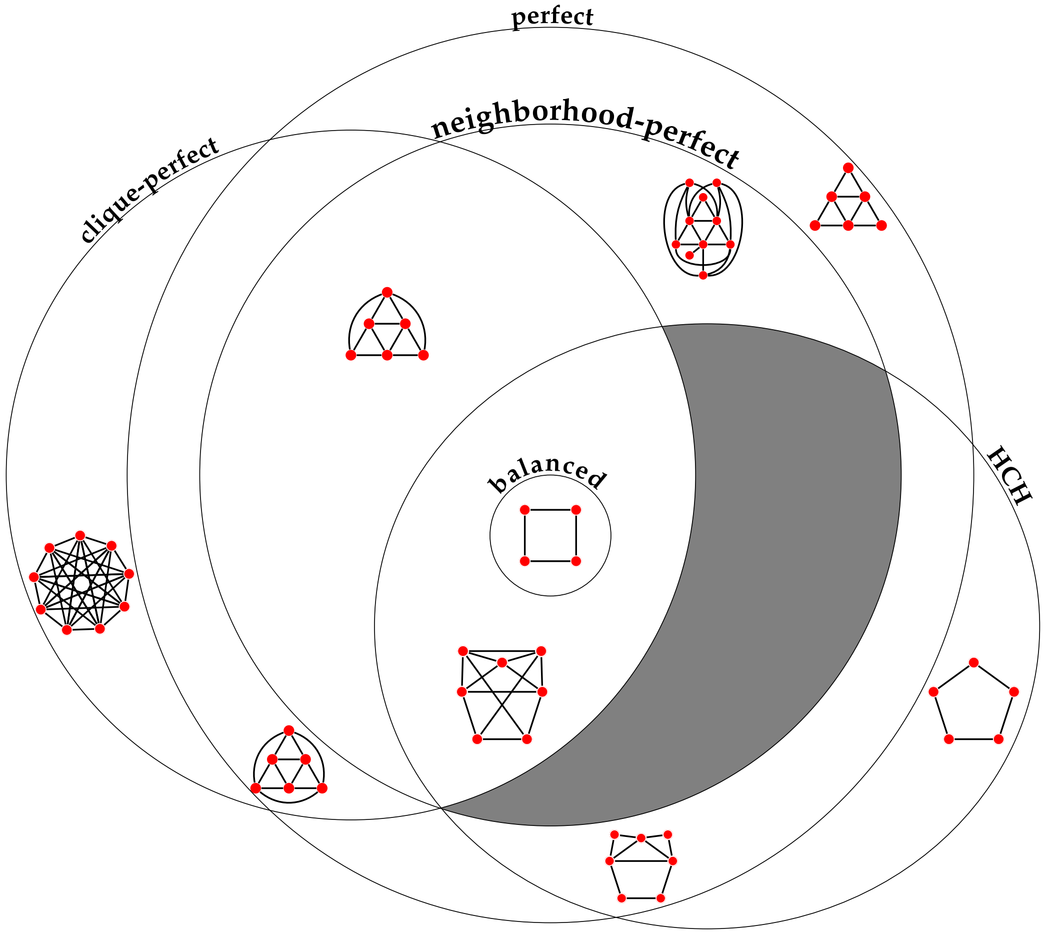

Figure 2.3: Containment relations among neighborhood-perfect, hereditary-clique-helly, clique-perfect, balanced and perfect graphs and their intersections. The shaded region corresponds to an empty set.

and only if its complement is α-perfect. In his work Berge posed two conjectures regarding the structures of these classes and their relationships.

The weaker of these conjectures is known now as the Perfect Graph Theorem, and it states that the class of perfect graphs is closed under complementation (or equivalently that the class of χ-perfect and α-perfect graphs coincide). This conjecture was proven in 1972 by Lovasz [74]. Since then the term *perfect graphs* is used to refer indistinctly to α- and χ-perfection.

The stronger conjecture due to Berge was later known as the Strong Perfect

Graph Theorem, and it states that a graph is perfect if and only if it contains no odd hole and no odd antihole. This fact was proved in 2005 by Chudnovsky et al. [32]. In addition, an $\mathcal{O}(n^9)$-time algorithm was devised in [31] that decides whether or not a given graph G having n vertices has an odd hole or an odd antihole.

Perfect graphs have already been characterized by means of the integrality of their fractional set packing polytopes. That is, Chvatal had proven in 1975 that a graph is perfect if and only if its clique-matrix is perfect [33]. The *fractional set polytope* of a $\{0,1\}$-matrix A is $\mathcal{P}(A) = \{x \in \mathbb{R}^n : Ax \leqslant 1, 0 \leqslant x \leqslant 1\}$, and A is *perfect* when $\mathcal{P}(A)$ is integral (i.e., all its extreme points have integer coordinates).

2.3.2 Neighborhood-Perfect Graphs

The class of *neighborhood-perfect* graphs was defined by Lehel and Tuza en 1986 [70] by means of a min-max type equality very similar to that used in the definition of perfect graphs (hence its name). A graph G is neighborhood-perfect when the inequality (2.3) turns into an equality for all induced subgraphs. That is when $\alpha_n(G') = \rho_n(G')$ for all induced subgraphs G' of G.

It was seen by Lehel and Tuza that odd holes and odd antiholes are not neighborhood-perfect, and hence the Strong Perfect Graph Theorem implies that all neighborhood-perfect graphs are also perfect.

The computational complexity of the problem of recognizing neighborhood-perfect graphs is still unknown in general, but many partial results on this direction have been given. These and other algorithmic properties of this class are studied in Chapter 4. In Chapter 3, we give a summary of previous results on structural characterizations of the class and prove new characterizations of neighborhood-perfect graphs restricted to several other graph classes.

2.3.3 Clique-Perfect Graphs

Graphs for which the inequality (2.4) holds with equality for all induced subgraphs were defined by Guruswami and Pandu Rangan to be *clique-perfect*

[56]. This is, a graph G is clique-perfect if $\alpha_c(G') = \tau_c(G')$, for all induced subgraph G' of G. It is important to mention that clique-perfect graphs are not all perfect (although they were conjectured to be when they were defined) since, for example, the antiholes that have a number of vertices multiple of 3 are all clique-perfect (Reed, 2001, see [43]). There are as well perfect graphs that are not clique-perfect; a simple example of such a graph is 0-pyramid, which is perfect but is not clique-perfect because $\tau_c(P) = 2$ but $\alpha_c(P) = 1$. Although neighborhood-perfect graphs and clique-perfect graphs are very much related [56, 69] there are graphs that are neighborhood-perfect and not clique-perfect and vice versa. For example the 3-pyramid is clearly not neighborhood-perfect [69], but it is clique-perfect. An example of a graph that is neighborhood-perfect but not clique-perfect can be found in Figure 2.3.

Unlike the class of perfect graphs, the complete characterization of clique-perfect graphs by forbidden induced subgraphs is not known. The algorithmic complexity of determining whether or not a graph is clique-perfect, for general graphs, is not known either. Nevertheless partial results in both of these directions have been obtained. In [12, 13, 17, 20, 21], forbidden induced subgraph characterizations of clique-perfectness restricted to two subclasses of claw-free graphs, Helly circular-arc graphs, diamond-free graphs, two superclasses of triangle-free graphs, P_4-tidy graphs and complements of line graphs were proved, leading to polynomial-time (or even linear-time) recognition algorithms for clique-perfectness within the same graph classes.

2.3.4 Helly Property and Hereditary Clique-Helly Graphs

A family $\mathcal{F}$ of sets has the *Helly property* if every nonempty subfamily of $\mathcal{F}$ of pairwise intersecting members has nonempty intersection. A graph is called *clique-Helly* if the family of all its cliques has the Helly property. Clique-Helly graphs have been considered in many papers [59, 45, 87, 9, 73], among others. A *hereditary clique-Helly* graph is a graph such that each of its induced subgraphs is clique-Helly. Prisner characterized hereditary clique-Helly graphs both by forbidden submatrices of their clique-matrices and by minimal forbidden induced subgraphs [86]. He proves that a graph is hereditary clique-Helly if and only if it contains no induced 0-, 1-, 2-, or 3-pyramid (see Figure 2.1). In the same work he gave a $\mathcal{O}(n^2m)$-time recognition algorithm for the class. Moreover he proved that every clique of a hereditary clique-Helly graphs has an edge that does not belong to any other edge; we shall call this edge a *proper* edge. Clearly this implies that if G is hereditary clique-Helly, and m is the number of edges of G, the number of maximal cliques cannot be greater than m plus the number of isolated vertices. Thus, by means of an algorithm devised in [96] that enumerates the cliques of a graph one after another in $\mathcal{O}(nm)$ time per clique, he concluded that in $\mathcal{O}(m^2n)$ time one can decide whether or not a graph is hereditary clique-Helly and if affirmative, output a clique-matrix of it. Prisner proved as well that if a graph G has the property that for every induced subgraph G' of G, all cliques of G' have a proper edge, then G must be hereditary clique-Helly. This together with the observation that all cliques of a hereditary clique-Helly graph have a proper edge, gives a characterization of the class.

It was proved in [69] that if a graph G is such that all its cliques have proper edges, then $\alpha_n(G) = \alpha_c(G)$ and $\rho_n(G) = \tau_c(G)$. This evidently implies that in the class of hereditary clique-Helly graphs, the class of clique-perfect and neighborhood-perfect graphs must coincide. Nevertheless there are graphs that are hereditary clique-Helly and are not clique-perfect or neighborhood-perfect [13] (see Figure 2.3). Moreover, clearly every odd cycle is hereditary clique-Helly, but not perfect.
We shall denote the class of hereditary clique-Helly graphs by *HCH* from now on.

2.3.5 Balanced Graphs

In 1969, Berge defined a $\{0,1\}$-matrix to be *balanced* if it contains no edge-vertex incidence matrix of any cycle of odd length as a submatrix. *Balanced* graphs were defined as those graphs whose clique-matrix is balanced. These graphs were considered by Berge and Las Vergnas in 1970 [8] but the name 'balanced graphs' appears explicitly for the fist time in [7]. It was proven in [8] that balanced graphs are a subclass of perfect graphs. From this same work, it follows that balanced graphs are as well a subclass of clique-perfect graphs. Moreover, from [5] it follows that balanced graphs are hereditary clique-Helly graphs. Thus by the observations made in Section 2.3.4, balanced graphs are also neighborhood-perfect, for they are clique-perfect and hereditary clique-Helly.

Balanced graphs were characterized by a family of forbidden induced subgraphs known as *extended odd suns* [16]. Nevertheless this characterization is not by *minimal* forbidden induced subgraphs, because some extended odd suns contain other extended odd suns as induced subgraphs. In fact the problem of characterizing balanced graphs by minimal forbidden induced subgraphs is still open. Partial results in this direction can be found in [18, 19, 20], where minimal forbidden induced subgraph characterization of balanced graphs restricted to co-bipartite graphs, line graphs and their complements, some classes of circular-arc graphs, paw-free graphs and P_4-tidy graphs are given. The best time complexity of a recognition algorithm for the whole class of balanced graphs is $\mathcal{O}(m^9 + n)$, which is achieved by computing a clique-matrix using the algorithm in [96] and then relying on the algorithm in [105] to decide whether or not such clique-matrix is balanced (details of the derivation can be found in [18]). In fact, it was shown in [88] that there is a strong tie between the time complexities of the problem of recognizing balanced graphs and that of recognizing balanced matrices; namely, any recognition algorithm for balanced graphs having time complexity $O(n^p)$ with $p < 9$ would improve on the time complexity for the recognition of balanced $\{0, 1\}$-matrices given in [105], which is the best currently known. Linear-time recognition algorithms of balancedness were found for graphs that are P_4-tidy or paw-free [20] as well as for co-bipartite graphs and for line graphs and their complements [18].

2.3.6 Circular-arc Graphs and Helly Circular-arc Graphs

Given a finite family $\mathcal{F}$ of sets, the *intersection graph* of $\mathcal{F}$ is a graph having as vertices the sets of $\mathcal{F}$ and where two vertices are adjacent if and only if their corresponding sets intersect. A *circular-arc* graph is a the intersection graph of a set of arcs on a circle. Such a set of arcs is called a *circular-arc model* of the graph. This class of graphs was first studied by Tucker [97, 99, 100, 101]. Graphs of this class can be recognized in linear time [78].

A *Helly circular-arc* graph is a circular-arc graph that admits a circular-arc model having the Helly property [52]. In [72, 66] a linear-time recognition algorithm for Helly circular-arc graphs is given, as well as a characterization by forbidden induced subgraphs (called *obstacles*) of Helly circular-arc graphs within the class of circular-arc graphs. In [13], a characterization by minimal forbidden induced subgraphs and a polynomial-time recognition algorithm are given for clique-perfect graphs within the class of Helly-circular arc graphs.
From now on we shall denote the class of Helly circular-arc graphs as *HCA*.

2.3.7 Modular Decomposition of a Graph and Superclasses of Cographs

Let G be a graph. We shall say that a vertex v of G *distinguishes* between two vertices x and y of G if it is adjacent to one of them and nonadjacent to the other. A set M of vertices shall be called a *module* of G if there is no vertex of $V(G) \setminus M$ that distinguishes any pair of vertices of M, or equivalently every vertex of G not in M is either adjacent to all vertices of M or to none of them. The empty set, the singletons $\{v\}$ for each $v \in V(G)$ and $V(G)$ are the *trivial modules* of G. A graph is said to be *prime* if it has more than two vertices and it has only trivial modules (for example, P_4 is a prime graph.) A nonempty module is *strong* if, for every other module M' of G, either $M' \subseteq M$, $M \subseteq M'$ or $M \cap M' = \emptyset$. The *modular decomposition tree* $T(G)$ of a graph G is a rooted tree having one node for each strong module of G and such that a node h representing a strong module

M has as its children the nodes representing the maximal strong modules of G properly contained in M. Clearly the root of the tree represents the module $V(G)$, and its leaves are the singletons $\{v\}$, for each $v \in V(G)$.
For each node h of $T(G)$, we note the module represented by h as $M(h)$. Note that by construction if we take every leaf of the tree and we associate it with the only vertex of its module, the set of vertices $M(h)$ corresponds to the set of leaves that have h as an ancestor in $T(G)$.
For each node h of $T(G)$, we denote the induced subgraph $G[M(h)]$ by $G[h]$ and call it the *graph represented by h*. Each node of $T(G)$, that is not a leaf, is a *parallel*, *series* or *neighborhood* node, abbreviated *P-node*, *S-node* and *N-node*, respectively. If $G[h]$ is disconnected, then h is a P-node; if $\overline{G[h]}$ is disconnected, then h is a S-node; and, if both $G[h]$ and $\overline{G[h]}$ are connected, then h is a N-node. Thus, if h is an internal node of $T(G)$ and $h_1, \dots, h_k$ are the children of h in $T(G)$, then one of the following conditions holds:

- If $G[h]$ is disconnected, then h is a P-node and $G[h_1], \dots, G[h_k]$ are the components of $G[h]$.
- If $\overline{G[h]}$ is disconnected, then h is an S-node and $G[h_1], \dots, G[h_k]$ are the anticomponents of $G[h]$.
- If $G[h]$ and $\overline{G[h]}$ are both connected, then h is an N-node and $M(h_1, \dots, M(h_k)$ are the maximal strong modules of $G[h]$ properly contained in $M(h)$.

In all of these cases, it holds that $\{M(h_1), \dots, M(h_k)\}$ is a disjoint partition of the vertices in $M(h)$ [50, 24]. An example of a modular decomposition tree together with the graph it represents, can be seen in Figure 2.4.

Let h be a node of $T(G)$ and let $h_1, \dots, h_k$ be its children. We shall denote as $\pi(h)$ the graph having vertex set $\{h_1, \dots, h_k\}$ and such that h_i is adjacent to h_j if and only if there is some edge in G joining a vertex of $M(h_i)$ and a vertex of $M(h_j)$. Since $M(h_i)$ and $M(h_j)$ are both modules of $G[h]$, then clearly there is an edge between them if and only if every vertex of $M(h_i)$ is adjacent to every

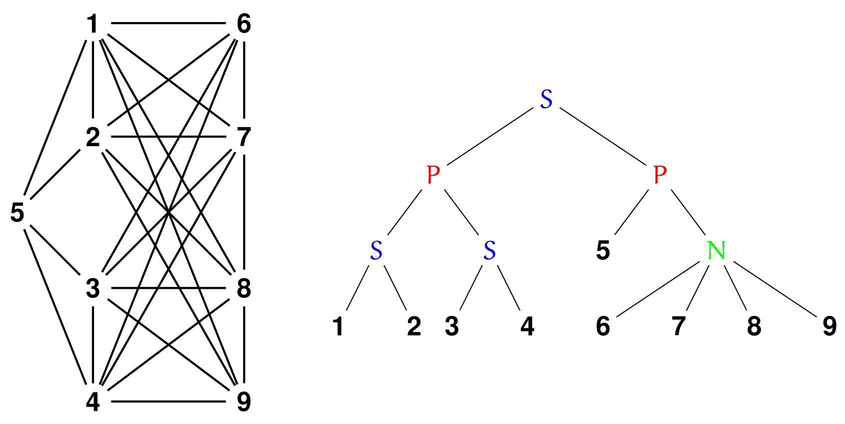

Figure 2.4: A graph with its modular decomposition tree

vertex of $M(h_j)$. Hence $G[h]$ coincides with the graph that arises from $\pi(h)$ by successively substituting h_i by $G[h_i]$, for each h_i. It is easy to see that, if h is an N-node, then $\pi(h)$ is a prime graph. We shall denote by $\pi(G)$ the set $\{\pi(h)\colon h \text{ is an N-node of } T(G)\}$. The following result shows that every induced prime subgraph of a graph G is also an induced subgraph of a graph in $\pi(G)$

Theorem 2.1 ([46]). *Let Z be a prime graph. A graph G is Z-free if and only if each graph of $\pi(G)$ is Z-free.*

In the rest of this work we shall denote $|V(G[h])|$ by $n(h)$, for every $h \in V(T(G))$. If h is an N-node, then we shall note $|V(\pi(h))|$ by $n_\pi(h)$. A fact that will be used in later chapters is that since $T(G)$ has n leaves and each internal node has at least two children, $T(G)$ must have less than $2n$ nodes. An important property that will be used extensively in Chapter 4 is that the sum of $n_\pi(h)$, for all the N-nodes h of $T(G)$, is at most $2n$ [2]

In this thesis we shall assume that each N-node h of the modular decomposition tree $T(G)$ is accompanied by a description of the prime graph $\pi(h)$ by means of an adjacency list. There are linear-time algorithms to compute the rooted tree $T(G)$ [40, 79, 41, 94]; moreover in [2] it is shown that the adjacency lists of each $\pi(h)$, for every N-node h, can be added also in linear time. For a survey on the algorithmic aspects of modular decompositions, see [58].

A *cograph* is a P_4-free graph. In [91], Seinsche proved a property of cographs that imply that they are perfect, namely that for any cograph G either G or $\overline{G}$ is disconnected. This property clearly implies that a modular decomposition tree $T(G)$ of a cograph may only contain S-nodes and P-nodes. If G is a cograph, $T(G)$ is called a *cotree*. Using the fact that cotrees have no N-nodes, linear-time recognition algorithms and simple polynomial and linear-time algorithms to solve classical graph theory problems where given for cographs [35, 36, 37].

The classes of P_4-*tidy* graphs and *tree-cographs* are superclasses of cographs, and have well understood modular decomposition trees.

P_4-tidy

The class of P_4-*tidy* graphs extends many other generalizations of cographs, that arise by limiting the number of induced P_4's that occur in a graph. For example, it contains as a subclass the class of P_4-sparse graphs, which consists of those graphs G such that every set of five vertices induces at most one P_4 in G [64].
A graph G is P_4-*tidy* if for each 4-vertex set A that induces a P_4 in G, there is at most one vertex v such that $G[A \cup \{v\}]$ is a graph with two induced P_4's. There is a structure theorem for P_4-tidy graphs that extends Seinsche's theorem in cographs. To state this theorem, we shall first define two families of graphs called *starfishes* and *urchins*.
A *starfish* is a graph whose vertex set can be partitioned in three disjoint sets, S, C and R, where each of the following conditions holds:

- $S = \{s_1, \ldots, s_t\}$ is a stable set and $C = \{c_1, \ldots, c_t\}$ is a clique, for $t \geqslant 2$.
- R is allowed to be empty. If it is not, then all vertices of R are adjacent to all vertices of C and nonadjacent to all vertices of S.
- s_i is adjacent to c_j if and only if $i = j$.

An *urchin* is a graph whose set can be partitioned intro three sets S, C and R satisfying the first two conditions stated above, but instead of satisfying the third one, it must satisfy that:

- s_i is adjacent to c_j if and only if $i \neq j$.

It is clear that an urchin is the complement of a starfish and vice versa. Given G, a starfish or an urchin, and a partition (S, C, R), we shall call S the *ends* of G, C the *body* of G and R the *head* of G. A *fat urchin* (resp. *fat starfish*) arises from an urchin (resp. starfish), with partition (S, C, R), by substituting exactly one vertex of $S \cup C$ by a K_2 or a $2K_1$. In other words, by adding a true or false twin to a vertex of $S \cup C$.
Now that we have defined starfishes and urchins, we can state the structural theorem of P_4-tidy graph that we shall use in Chapters 3 and 4.

Theorem 2.2 ([53]). *If G is a P_4-tidy graph, then exactly one of the following statements holds:*

1. *G or $\overline{G}$ is disconnected.*
2. *G is isomorphic to C_5, P_5, $\overline{P_5}$, a starfish, a fat starfish, an urchin, or a fat urchin.*

The above theorem implies that if we take a P_4-tidy graph G and an N-node h of $T(G)$, then $\pi(h)$ is isomorphic to C_5, P_5, $\overline{P_5}$, a prime starfish or a prime urchin. And clearly if $\pi(h)$ is isomorphic to a C_5, P_5 or $\overline{P_5}$, then h has five children, each representing a vertex of G. If on the other hand $\pi(h)$ is a prime starfish or urchin, then each of the children of h in $T(G)$ is a leaf, with the possible exception of one child, h_R, that represents the head of the urchin or starfish, and the other possible exception of a child representing a $2K_1$ or K_2 (if $\pi(h)$ is a fat starfish or fat urchin). It was shown in [53] that it can be decided in $\mathcal{O}(n_{\pi}(h))$ time whether or not $\pi(h)$ is a starfish (or an urchin) and if so, find its partition.
The theorem stated above leads to linear-time recognition algorithms for the class of P_4-tidy graphs, as well as polynomial-time algorithms for many well known optimization problems in the same graph class [53].

Tree-cographs

The class of *tree-cographs* is another generalization of the class of cographs. But, instead of limiting the number of P_4's, this class introduces N-nodes to the

modular decomposition tree, such that the corresponding module induces a tree or complement of a tree in the graph.
Tree-cographs were defined in [95] recursively as follows:

- Every tree is a tree-cograph.
- If G is a tree-cograph, then $\overline{G}$ is a tree-cograph.
- The disjoint union of tree-cographs is a tree-cograph.

This definition is completely analogous to the recursive definition of cographs (see [35]), but instead of using single vertices as the building blocks, the building blocks for tree-cographs are all the trees. It is easy to see that if G is a tree-cograph, then if h is an N-node of $T(G)$, $\pi(h)$ must be a tree or the complement of a tree. Suppose we have a tree-cograph G, such that $T(G)$ has an N-node h. As clearly the class of tree-cographs is hereditary, $G[h]$ must be a tree-cograph. But as h is not an S-node or a P-node, $G[h]$ must be a necessarily a co-connected tree or a connected co-tree and hence the prime graph $\pi(h)$ is a tree or a co-tree. Recent research related to tree-cographs includes [14, 15, 22, 77, 82].

2.4 Complexity Theory and Approximation Algorithms

In this subsection we shall give some basic notions of complexity theory and approximation algorithms that we shall need in Chapter 4.
One of the most important factors used to classify different computational problems is the amount of resources required to solve them. These resources are generally measured in the amount of operations needed to compute the answer (time complexity) and the amount of space needed to perform these operations (space complexity). The primary concern of computational complexity theory is to determine these classifications. In particular, the most basic task is to determine which problems can be efficiently solved and which cannot.
We shall say that an algorithm runs in *polynomial time* if there is a polynomial P such that for each input I, the number of operations that the algorithm performs

with this input is less than or equal to $P(|I|)$, where $|I|$ is the size of the input I. A problem can be solved in polynomial time if there is an algorithm that solves it and runs in polynomial time. A problem can be *efficiently* solved if it can be solved in polynomial time. The complexity class $\mathcal{P}$ is the class of all computational problems that can be solved in polynomial time.

While many problems are known to be in $\mathcal{P}$, many others are neither known to be in $\mathcal{P}$, nor proven to be outside $\mathcal{P}$. Many of such problems are known to be in the class $\mathcal{NP}$. The class of $\mathcal{NP}$ problems is the class of all problems whose solutions can be *verified* in polynomial time. Several other equivalent definitions exist, the one which motivates the name of the class states that a problem is $\mathcal{NP}$ if there is a non-deterministic Turing machine that solves it in polynomial time. Hence it is a $\mathcal{N}$on-deterministic $\mathcal{P}$olynomial problem. Clearly all problems in $\mathcal{P}$ are in $\mathcal{NP}$ as well. One of the most fundamental open questions in complexity theory is whether or not $\mathcal{P} = \mathcal{NP}$.

While the $\mathcal{P}$ vs. $\mathcal{NP}$ problem is still open, one may still classify computational problems as those that are in $\mathcal{P}$ and those that are $\mathcal{NP}$-*hard*. $\mathcal{NP}$-hard problems are, in a sense, those problems that are at least as hard to solve as any $\mathcal{NP}$ problem. A problem Π is said to be $\mathcal{NP}$-*hard* if every problem in $\mathcal{NP}$ can be *efficiently reduced* to Π, where an efficient reduction usually means a polynomial-time reduction [67, 34, 71]. Given two decision problem (with "yes"/"no" outputs) Π_1 and Π_2, a *polynomial-time reduction*[2] from Π_1 to Π_2 is a function F that transforms inputs of Π_1 into inputs of Π_2, such that F can be computed in polynomial time and satisfies that $\Pi_1(I_1)$ is affirmative if and only if $\Pi_2(F(I_1))$ is affirmative, where I_1 is any input of Π_1. Hence, it is easy to see that if we can solve Π_2 in polynomial time and there is a polynomial-time reduction from Π_1 to Π_2, that immediately implies that we can also solve Π_1 in polynomial time. Thus, the existence of a polynomial-time algorithm to solve any $\mathcal{NP}$-hard problem would imply that $\mathcal{P} = \mathcal{NP}$. And consequently if it is proved that $\mathcal{P} \neq \mathcal{NP}$, then no efficient algorithm would exist to solve $\mathcal{NP}$-hard problems. This means that unless one intends to prove $\mathcal{NP} = \mathcal{P}$, one should refrain from trying to come up

[2] Also called Karp-reduction. See [34] for the related notion of Cook-reductions.

with an efficient algorithm to solve exactly an $\mathcal{NP}$-hard problem. A subclass of $\mathcal{NP}$-hard problems are the $\mathcal{NP}$-*complete* problems, which are those problems in $\mathcal{NP}$ that are as well $\mathcal{NP}$-hard.

Many important *optimization problems*, in which one looks for an *optimum* among all plausible solutions, are known to be $\mathcal{NP}$-hard (in the sense that their associated decision problem is $\mathcal{NP}$-hard). For example determining most of the parameters presented in Section 2.2 is NP-hard for general graphs. A proof that a optimization problem is $\mathcal{NP}$-hard usually serves as an indicator that one should relax the specifications of the problem if one wishes to find 'good' solutions. This idea leads to consider *approximate* solutions instead of exact ones, that is solutions that are not optimal but that are within a small factor $C \geqslant 1$ from optimal. An approximation algorithm is said to be a C-*approximation* algorithm if the solution that the algorithm finds is at most C times worse than the optimal solution. In this case, C is called the *approximation ratio*. A large amount of research has been dedicated to finding efficient (close to 1) approximation algorithms for a variety of optimization problems.
The *approximation complexity* of a problem is the closest to 1 factor C_{opt} for which there can exist polynomial-time C_{opt}-approximation algorithms. Classifying approximation problems by their approximation complexity has been widely investigated. Some $\mathcal{NP}$-hard problems admit a polynomial-time *approximation scheme* (PTAS), which means that they can be approximated, in polynomial time, to wtihin any constant close to 1 (but not 1). The class of problems that admit a C-approximation algorithm is called $\mathcal{APX}$. This class was identified by Papadimitriou and Yannakakis [85].
The approximation complexity of several approximation problems is still open, namely for these problems the known upper and lower bounds for C_{opt} do not match. Clearly upper bounds for C_{opt} are given by finding polynomial-time approximation algorithms for the problem. On the other hand proving lower bounds is usually a much more difficult task. Numerous combinatorial optimization problems were shown $\mathcal{NP}$-hard to approximate to within a factor even

marginally lower than the best known efficient algorithm [76, 3, 60, 61]. In particular, the problem of Minimum Vertex Cover is perhaps the one that underscores the limitations of known technique for proving hardness of approximation (see [42]). Håstad proved in his celebrated paper [61] an inapproximability result for Minimum Vertex Cover with constant $\frac{7}{6}$, and this result was later improved to an inapproximability result with constant $1.3606\ldots$ by Dinur and Safra [42]. The approximation algorithm having the best approximation complexity for this problem is a 2-approximation algorithm discovered independently by Gavril and Yannakakis [84, 51]. This means that the C_{opt} for this problem is in between 2 and $1.3606\ldots$, although it is conjectured to be 2.
For more information on complexity theory we refer the reader to the classic books of Papadimitriou and Steiglitz [84], and Garey and Johnson [51]. For more information on approximation algorithms, we refer to the book by Vazirani [103], that covers the basic techniques used in modern research on these subjects.

Chapter 3

Partial Characterizations of Neighborhood-Perfect Graphs

In this chapter we shall explore characterizations of neighborhood-perfect graphs by forbidden induced subgraphs, restricted to certain classes of graphs. In Section 3.1 we shall show results on this subject that have already been given. In Section 3.2 we shall characterize minimaly non-neighborhood-perfect graphs with disconnected complement, that is graphs that are not neighborhood-perfect but are the join of two non-null neighborhood-perfect graphs. We shall also state properties of $\alpha_n(G)$ and $\rho_n(G)$ when G is the join of two graphs; these properties shall be used throughout the rest of the thesis. In Sections 3.3 and 3.4 we shall characterize neighborhood-perfect graphs restricted to the classes of P_4-tidy and tree-cographs. Finally in Section 3.5 we prove that the classes of neighborhood-perfect and clique-perfect graphs coincide when restricted to the class of hereditary clique-Helly graphs. Using this fact, we characterize neighborhood-perfect graphs restricted to subclasses of hereditary clique-Helly graphs and related classes.

3.1 Known Characterizations

In this section we shall present known partial characterizations of neighborhood-perfect graphs, so as to give a broader view of what has been done on this topic.

As was already stated in Chapter 1, in 1986 Lehel and Tuza defined neighborhood-perfect graphs and in the same article gave a characterization for neighborhood-perfect graphs restricted to the class of chordal graphs. Moreover they point out a characterization restricted to triangle-free graphs.

Theorem 3.1 ([70]). *Let G be a chordal graph, then it is neighborhood-perfect if and only if it has no induced odd suns.*

Corollary 3.2 ([70]). *All interval graphs are neighborhood-perfect.*

Theorem 3.3 ([70]). *If G is a triangle-free graph, then it is neighborhood-perfect if and only if it is bipartite.*

Lehel and Tuza also define *trivially neighborhood-perfect* graphs as those graphs satisfying that the neighborhood number is exactly 1 for all induced subgraphs.

Theorem 3.4 ([70]). *The class of trivially neighborhood-perfect graphs is equivalent to class of trivially perfect graphs.*

In [57] the authors define the class of minimally non-neighborhood-perfect graphs and characterize those that have $\alpha_n = 1$. They also give a characterization of neighborhood-perfect graphs restricted to the class of cographs and prove that all line graphs of bipartite graphs are neighborhood-perfect.

A *minimally non-neighborhood-perfect* graph is a graph G satisfying that $\alpha_n(G) < \rho_n(G)$ and $\alpha_n(G') = \rho_n(G')$ for all proper induced subgraph G'.

Theorem 3.5 ([57]). *If G is minimally non-neighborhood-perfect and $\alpha_n(G) = 1$, then G is a 0-pyramid or 3-pyramid.*

Theorem 3.6 ([57]). *A cograph is neighborhood-perfect if and only if it does not contain a 3-pyramid as induced subgraph.*

Theorem 3.7 ([57])**.** *The line graph of any bipartite multigraph is neighborhood-perfect.*

An extension of Theorem 3.7 characterizing exactly all graphs with neighborhood-perfect line graph was given in [69].

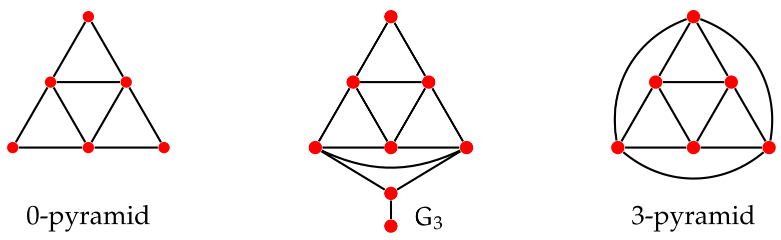

Figure 3.1: Minimal forbidden induced subraphs for neighborhood-perfect line graphs.

Theorem 3.8 ([69])**.** *Let G be the line graph of some graph. Then G is neighborhood-perfect if and only if it contains no odd hole and none of the graphs in Figure 3.1 as an induced subgraph.*

To prove this characterization the author first proves that the line graph of any balanced hypergraph is neighborhood-perfect; it is in the middle of this proof that he implicitly states Theorem 3.27.

3.2 Minimally Non-Neighborhood-Perfect Graphs with Disconnected Complement

In the following sections we shall give new results on characterizations of neighborhood-perfect graphs restricted to graph classes that have been characterized by their modular decomposition tree. In that respect we shall first state some results on the neighborhood number and independence neighborhood number of the join of two non-null graphs. And furthermore, we characterize those graphs that are not neighborhood-perfect but are the join of two neighborhood-perfect graphs.

Theorem 3.9. *If* G *and* H *are graphs, then*

$$\rho_n(G \vee H) = \min\{\gamma(H) + 1, \gamma(G) + 1, \rho_n(H), \rho_n(G)\}. \tag{3.1}$$

Proof. It is immediate to see that

$$\rho_n(G \vee H) \leqslant \min\{\gamma(H) + 1, \gamma(G) + 1, \rho_n(H), \rho_n(G)\},$$

for we can easily find neighborhood sets of $G \vee H$ with all four amounts considered. Simply take a minimum dominating set of either G or H and any vertex in the other graph or, instead, take a minimum neighborhood set in G or H.

Let us then prove that indeed the inequality above cannot hold strictly.

By contradiction let us say that we have a neighborhood set of S of $G \vee H$, with size strictly less than $\min\{\gamma(H)+1, \gamma(G)+1, \rho_n(H), \rho_n(G)\}$. Hence, as S has fewer vertices than $\rho_n(G)$ and $\rho_n(H)$, it must have at least one vertex in each G and H. For if not, there would be uncovered edges in the subgraphs corresponding to G or H in the join. Thus if we take $S_G = S \cap V(G)$ and $S_H = S \cap V(H)$, then $|S_H| \leqslant |S| - 1$ and $|S_G| \leqslant |S| - 1$. But as we are assuming that $|S| - 1 < \gamma(G)$ and $|S| - 1 < \gamma(H)$, we have that neither S_G nor S_H can be dominating sets of G and H respectively. This means that there must be at least some $v \in V(G)$ and some $w \in V(H)$ such that $v \notin N_G[S_G]$ and $w \notin N_H[S_H]$. And then if we take the edge (v, w) in $G \vee H$, it cannot be covered by S, for there is no vertex in S_H or S_G adjacent to both vertices and $S = S_G \cup S_H$. Thus, S is not a neighborhood set of the join, reaching the contradiction that proves the theorem. □

The following fact is easy to prove but still useful.

Lemma 3.10. *If* $G_1, \ldots, G_k$ *are graphs and* $k \geqslant 2$, *then*

$$\gamma(G_1 \vee \cdots G_k) = \min\{2, \gamma(G_1), \ldots, \gamma(G_k)\}.$$

Proof. Let $G = G_1 \vee \cdots\cdots G_k$. Clearly, $\gamma_(G) \leqslant \min\{2, \gamma(G_1), \ldots, \gamma(G_k)\}$ since any dominating set of any of the graphs $G_1, \ldots, G_k$ as well as any set $\{v_1, v_2\}$ where $v_1 \in V(G_1)$ and $v_2 \in V(G_2)$ are dominating sets of $G_1 \vee \cdots G_k$. Hence, if the formula were false, then $\gamma(G) < \min\{2, \gamma(G_1), \ldots, \gamma(G_k)\}$, which means that $\gamma(G) = 1$ and $\gamma(G_1), \ldots, \gamma(G_k)$ are greater than 1 all of them. Therefore, G has a universal vertex but none of $G_1, \ldots, G_k$ has a universal vertex, which contradiction the fact that $G = G_1 \vee \cdots \vee G_k$. □

We now give a formula of the neighborhood number for the join of more than two graphs.

Corollary 3.11. *If* $G_1, G_2, \ldots, G_k$ *are graphs and* $k \geqslant 2$*, then*

$$\rho_n(G_1 \vee \cdots \vee G_k) = \min\{3, \gamma(G_1)+1, \cdots, \gamma(G_k)+1, \rho_n(G_1), \cdots, \rho_n(G_k)\}.$$

Proof. The formula is valid when $k = 3$ because Theorem 3.9 and Lemma 3.10 imply

$$\begin{aligned}
\rho_n(G_1 \vee G_2 \vee G_3) &= \min\{\gamma(G_1 \vee G_2)+1, \gamma(G_3)+1, \rho_n(G_1 \vee G_2), \rho_n(G_3)\} \\
&= \min\{\min\{2, \gamma(G_1), \gamma(G_2)\}+1, \gamma(G_3)+1, \\
&\qquad \min\{\gamma(G_1)+1, \gamma(G_2)+1, \rho_n(G_1), \rho_n(G_2)\}, \rho_n(G_3)\} \\
&= \min\{3, \gamma(G_1)+1, \gamma(G_2)+1, \gamma(G_3)+1, \rho_n(G_1), \rho_n(G_2)\}, \rho_n(G_3)\}
\end{aligned}$$

Moreover, if the formula is valid when $k = t$ for some $t \geqslant 3$, then it is also valid when $k = t+1$ since Theorem 3.9 and Lemma 3.10 imply

$$\begin{aligned}
\rho_n(G_1 \vee \cdots \vee G_{t+1}) &= \min\{\gamma(G_1 \vee \cdots \vee G_t)+1, \gamma(G_{t+1})+1, \\
&\qquad \rho_n(G_1 \vee \cdots \vee G_t), \rho_n(G_{t+1})\} \\
&= \min\{\min\{2, \gamma(G_1), \cdots, \gamma(G_t)\}+1, \gamma(G_{t+1})+1, \\
&\qquad \min\{3, \gamma(G_1)+1, \cdots, \gamma(G_t), \rho_n(G_1), \cdots, \rho_n(G_t)\}, \\
&\qquad \rho_n(G_{t+1})\} \\
&= \min\{3, \gamma(G_1)+1, \cdots, \gamma(G_{t+1})+1, \rho_n(G_1), \ldots, \rho_n(G_{t+1})\}.
\end{aligned}$$

By induction, the formula is valid for every $k \geqslant 3$. □

Theorem 3.12. *If* G *and* H *are graphs, then*

$$\alpha_n(G \vee H) = \min\{\alpha_2(G), \alpha_2(H)\}. \tag{3.2}$$

Proof. Let us first note that if a neighborhood-independent set of $G \vee H$ has size larger than 1, then it must have no edges belonging to $E(G)$ or $E(H)$. For in the join all edges between vertices of G are in the closed neighborhood of any vertex of H and likewise between the edges of H and the vertices of G. Similarly it cannot have any vertices, for every vertex in G is in the closed neighborhood of every vertex of H.

Now, let us prove that $\alpha_n(G \vee H) \geqslant \min\{\alpha_2(G), \alpha_2(H)\}$, by finding a neighborhood-independent set of $G \vee H$ of that size. Without loss of generality, suppose $\alpha_2(G) \leqslant \alpha_2(H)$. Let I_G be an 2-independent set of G and I_H be one of H, both of size $\alpha_2(G)$. Clearly as both I_H and I_G are independent sets in H and G, then they are also independent sets in $G \vee H$ and so $I_H \cup I_G$ induces a complete bipartite subgraph of $G \vee H$. Let M be a perfect matching between I_G and I_H in $G \vee H$. Clearly $|M| = |I_G| = |I_H| = \alpha_2(G)$. We will proceed to show that M is a neighborhood-independent set.

Suppose to the contrary that there are two edges in M, e_1 and e_2, such that there exists a vertex u of $V(G \vee H)$ satisfying $e_1, e_2 \subseteq N[u]$. Let us write $e_1 = v_1w_1$ and $e_2 = v_2w_2$, with $v_1, v_2 \in I_G$ and $w_1, w_2 \in I_H$. As u is a vertex of the join then u must belong to $V(G)$ or $V(H)$. If $u \in V(G)$, then v_1uv_2 is a path of length 2 from v_1 to v_2, in G. If $u \in V(H)$, then w_1uw_2 is a path of length 2 in H that connects w_1 and w_2. In both cases we reach a contradiction, because both I_G and I_H were 2-independent sets. Therefore, M must be a neighborhood-independent set of size $\alpha_2(G)$ and the inequality $\alpha_n(G \vee H) \geqslant \min\{\alpha_2(G), \alpha_2(H)\}$ must hold.

Now, if $\alpha_n(G \vee H) = 1$, then by the previous inequality we have the equality we were looking for. Let us then suppose that $\alpha_n(G \vee H) > 1$, which by the first observation of this proof implies that any neighborhood-independent set of the join must be a matching between vertices of G and H. Let M be any neighborhood-independent set of size $\alpha_n(G \vee H)$. We define Y_H and Y_G as the sets of vertices of H and G respectively such that $Y_H = \{w \in V(H) \colon \exists e \in M, w \in e\}$ and $Y_G = \{v \in V(G) \colon \exists e \in M, v \in e\}$. Clearly $|Y_H| = |Y_G|$, for every edge in M has one vertex in G and one in H. We shall see now that both are 2-independent sets.

Suppose again by contradiction that there are two vertices in Y_G, v_1 and v_2, such that $d_G(v_1, v_2) \leqslant 2$. This implies that there must exist a vertex $u \in V(G)$ such that $v_1, v_2 \in N_G[u]$ which clearly also means that $v_1, v_2 \in N_{G\vee H}[u]$. If we now take w_1 and w_2 in Y_H such that $v_i w_i \in M$ for each $i \in \{1, 2\}$, then clearly $v_1 w_1$ and $v_2 w_2$ cannot be neighborhood-independent edges, because if $u \in V(G)$, then both $w_1, w_2 \in N_{G\vee H}[u]$. This contradicts the fact that M is a neighborhood-independent set. The contradiction proves that Y_G must be a 2-independent set of G. By the same reasoning Y_H must be a 2-independent set of H. Hence as $|Y_G| \leqslant \alpha_2(G)$ and $|Y_H| \leqslant \alpha_2(H)$, then $|M| = |Y_H| = |Y_G| \leqslant \min\{\alpha_2(G), \alpha_2(H)\}$ and therefore $\alpha_n(G \vee H) \leqslant \min\{\alpha_2(G), \alpha_2(H)\}$, proving the reverse inequality and the theorem. □

We state the following immediate consequence for future reference.

Corollary 3.13. *If* $G_1, \dots, G_k$ *are graphs and* $k \geqslant 3$, *then*

$$\alpha_n(G_1 \vee \cdots \vee G_k) = 1.$$

Proof. Since every two vertices of $G_1 \vee \cdots \vee G_{k-1}$ are at distance at most two, $\alpha_2(G_1 \vee \cdots G_{k-1}) = 1$. Hence, Theorem 3.12 implies that $\alpha_n(G_1 \vee \cdots \vee G_k) = \min\{\alpha_2(G_1 \vee \cdots \vee G_{k-1}), \alpha_2(G_k)\} = 1$. □

As we have already stated, minimally non-neighborhood-perfect graphs were defined in [57]. Here we shall characterize the minimum non-neighborhood-perfect graphs that have a disconnected complement (that is, that they are formed by the join of two non-null graphs).

For that purpose we shall first define a subclass of neighborhood-perfect graphs, the strongly neighborhood-perfect graphs.

Definition 3.14. *We shall say that a graph* G*, is* ***strongly neighborhood-perfect*** *if* $\alpha_2(G') = \rho_n(G')$ *for every induced subgraph* G' *of* G*.*

Definition 3.15. *We shall say that a graph* G*, is* ***minimally non-strongly neighborhood-perfect*** *when* $\alpha_2(G) < \rho_n(G)$*, but* $\alpha_2(G') = \rho_n(G')$ *for every proper induced subgraph* G'*, of* G*. That is, it is not strongly neighborhood-perfect, but all its proper induced subgraphs are.*

Observation 3.16. Clearly all strongly neighborhood-perfect graphs are neighborhood-perfect. It follows from the string of inequalities: $\alpha_2(G) \leqslant \gamma(G) \leqslant \alpha_n(G) \leqslant \rho_n(G)$, that holds for every graph G, and the equality demanded by the definition of strongly neighborhood-perfect graphs. Moreover it is also true that if G is neighborhood-perfect, then it is strongly neighborhood-perfect if and only if $\alpha_2(G') = \alpha_n(G')$ for every induced subgraph G'.

We shall see which graphs satisfy that $\alpha_2(G') = \alpha_n(G')$ for every induced subgraph G'. But before giving this characterization we shall prove a useful general property of chordal P_k-free graphs.

Lemma 3.17. *Any* k*-walk* W *in a* P_k*-free chordal graph must have at least two vertices that are* 2 *steps from each other in* W *and either are adjacent or the same vertex.*

Proof. Let G be a P_k-free chordal graph and W be a k-walk in G. Since G is P_k-free, then W cannot be an induced path. This means that there must exist an integer p such that $p \geqslant 2$ and there are at least two vertices of W which are p steps from each other and are either adjacent in G or the same vertex in G. We choose p as small as possible. If we show that $p = 2$, then the assertion of the lemma follows.

Let us suppose by contradiction that p is greater that 2. But if we take the two vertices in W that are p steps from each other and consider the sub-walk of W

joining them (of length p). As the minimality of p implies that the vertices that are at fewer than p steps from each other in W are different and nonadjacent, this sub-walk must induce C_p or C_{p+1} in G, depending on whether the two vertices are the same or adjacent. But as the graph was chordal and p was greater than 2, this results in a contradiction, proving the lemma. □

Lemma 3.18. *A graph G satisfies $\alpha_2(G') = \alpha_n(G')$ for every induced subgraph G' if and only if G is P_6-free chordal.*

Proof. If G is a graph, let $S \subseteq V(G) \cup E(G)$ be a neighborhood-independent set of size $\alpha_n(G)$ and of minimum number of edges. We shall show that S must contain only vertices and therefore be a 2-stable set of G, proving the lemma (for $\alpha_2(G) \leqslant \alpha_n(G)$ is clearly true for all graphs).

Assume to the contrary that there is an edge $e = xy \in S$. As e cannot be replaced by x in S, maintaining the neighborhood-independence (for S had minimum number of edges), then there must exist an $s \in S$ (an edge or vertex), such that $N[x] \cap N[s] \neq \emptyset$. But as $e, s \in S$, then $N[x] \cap N[y] \cap N[s] = \emptyset$, which means that there is a vertex $x' \in N[x] \cap N[s]$ such that $x' \notin N[y]$. Moreover $x \in N[x] \cap N[y]$, which implies that $x \notin N[s]$, meaning that there must be a vertex x'', such that $x'' \notin N[x]$ and either $x'' \in s$ if s is an edge or $x'' = s$ if s is a vertex. But as $x'' \in s$ (or $x'' = s$), and $x' \in N[s]$, then $x'' \in N[x']$ and $x'' \notin N[x]$. By a symmetry argument, there must be vertices y' and y'', such that $y' \in N[y] - N[x]$ and $y'' \in N[y'] - N[y]$. But then $x''\ x'\ x\ y\ y'\ y''$ form a 6-walk where no two vertices that are two steps from each other are adjacent or the same. This together with Lemma 3.17, results in a contradiction, proving that no edge can belong to S and therefore S must be a 2-independent set of size $\alpha_n(G)$. □

Using the previous characterization, we shall state the following corollary, fully characterizing strongly neighborhood-perfect graphs by forbidden induced subgraphs.

Corollary 3.19. *If G is a graph, the following statements are equivalent:*

1. G *is strongly neighborhood-perfect*
2. G *is neighborhood-perfect* $\cap \{C_4, C_6, P_6\}$*-free*
3. G *is odd-sun-free* $\cap$ P_6*-free chordal*

Proof. Clearly, by Observation 3.16, G is strongly neighborhood-perfect if and only if G is neighborhood-perfect and $\alpha_2(G') = \alpha_n(G')$ for every induced subgraph G', which, by Lemma 3.18, holds if and only if G is neighborhood-perfect and P_6-free chordal. But, as all odd holes are forbidden induced subgraphs of neighborhood-perfect graphs [70], then G is neighborhood-perfect and P_6-free chordal if and only if it is neighborhood-perfect and $\{C_4, C_6, P_6\}$-free, proving (1) if and only if (2). Moreover, by [70] a chordal graph is neighborhood-perfect if and only if it is odd-sun-free, clearly implying (2) if and only if (3). □

We shall now prove the main result of this section, a characterization of minimally non-neighborhood-perfect graphs with disconnected complement (ie., formed by the join of two non-null subgraphs).

Theorem 3.20. *The only minimally non-neighborhood-perfect graphs with disconnected complement are* $C_4 \vee 2K_1$, $C_6 \vee 3K_1$ *and* $P_6 \vee 3K_1$.

Proof. Clearly a graph with disconnected complement can be thought of as the join of two non-null graphs. Let us consider we have a minimally non-neighborhood-perfect graph $G \vee H$. By minimality G and H must be neighborhood-perfect, but as $G \vee H$ is minimally non-neighborhood-perfect, $\rho_n(G \vee H) \neq \alpha_n(G \vee H)$ which, by Theorem 3.9 and Theorem 3.12, implies that G or H must satisfy $\alpha_2 \neq \rho_n$ (because $\alpha_2(W) < \gamma(W) + 1$ is true for all graphs W).

We shall note that if a graph is neighborhood-perfect but does not satisfy $\alpha_2 = \rho_n$, then it is neighborhood-perfect but not **strongly** neighborhood-perfect, which, by (1) if and only if (2) in Corollary 3.19, means that the graph must contain a C_4, C_6 or P_6 as induced subgraph.

Let us then suppose that both G and H do not satisfy $\alpha_n = \rho_n$. This means that both must contain a C_4, C_6 or P_6 as induced subgraph. If one of them contains an induced C_4, then $G \vee H$ must have $C_4 \vee 2K_1 = \overline{3K_2}$ as a proper induced subgraph, contradicting the minimality of $G \vee H$. On the other hand if none contain an induced C_4, then they must contain an induced C_6 or P_6, meaning that both must have at least an independent set of size 3. Hence $G \vee H$ must contain a $P_6 \vee 3K_1$ or $C_6 \vee 3K_1$ as a proper induced subgraph, but by Theorem 3.9 and Theorem 3.12, both have $\rho_n = 3 \neq 2 = \alpha_n$, meaning that they are not neighborhood-perfect. In both cases we have found a contradiction, therefore it cannot occur that both G and H do not satisfy $\alpha_n = \rho_n$.

We need only to consider the case where one of the graphs does not satisfy $\alpha_2 = \rho_n$. Let us say that G has an induced subgraph G', isomorphic to C_4, C_6 or P_6. Now, if only G does not satisfy $\alpha_2(G) = \rho_n(G)$, then $\alpha_2(G) < \rho_n(G)$ and $\alpha_2(H) = \rho_n(H)$. Hence $\alpha_2(G) < \alpha_2(H) \leqslant \alpha(H)$, because if not $G \vee H$ would be neighborhood-perfect. Thus we can take $H' = (\alpha_2(G') + 1)K_1$ as an induced subgraph of H, for $\alpha_2(G') \leqslant \alpha_2(G) < \alpha(H)$. Then once again $G' \vee H'$ must be a $\overline{3K_2}$, $C_6 \vee 3K_1$ or $P_6 \vee 3K_1$. But now as $G' \vee H'$ is an induced subgraph of $G \vee H$, by minimality $G \vee H = G' \vee H'$, proving the theorem. □

3.3 P_4-tidy and P_4-sparse Graphs

In this section we shall characterize the neighborhood-perfect graphs, restricted to the class of P_4-tidy graphs and as a corollary we shall deduce the characterization restricted to the class of P_4-sparse graphs. For this we will strongly rely on the characterization of minimally non-neighborhood-perfect graphs with disconnected complement shown in the previous section.

Let us then begin by stating the values of $\alpha_n(G)$ and $\rho_n(G)$ for a connected and co-connected P_4-tidy graph.

Theorem 3.21. *If G is a nontrivial connected and co-connected P_4-tidy graph, then one of the following statements holds:*

1. *G is isomorphic to C_5, $\rho_n(G) = 3$ and $\alpha_n(G) = 2$.*
2. *G is isomorphic to P_5 or $\overline{P_5}$ and $\alpha_n(G) = \rho_n(G) = 2$.*

3. G *is a starfish with* t *ends or a fat starfish arising from one, and* $\alpha_n(G) = \rho_n(G) = t$.
4. G *is an urchin or a fat urchin with at least* 3 *ends, and* $\rho_n(G) = 2$, $\alpha_n(G) = 1$.

Proof. Since G is P_4-tidy, connected and co-connected, it follows by Theorem 2.2 that G is isomorphic to C_5, P_5, $\overline{P_5}$, a starfish, a fat starfish, an urchin or a fat urchin. The values of ρ_n and α_n for C_5, P_5 and $\overline{P_5}$ can be easily checked by simple inspection.

We shall then consider first the case where G is a starfish with partition (S, C, R), such that $|S| = t$, or a fat starfish arising from the substitution of a vertex c of C by a K_2, or by the substitution of a vertex s from S by a K_2 or $2K_1$. In all cases there is a neighborhood set of size t formed by taking t vertices from C. If G is a starfish without substitution of a vertex of C then we take all C, if on the other hand it is a starfish where a vertex c of C has been substituted by a K_2 or $2K_1$, we take only one of the vertices by which c has been substituted. If G is a fat starfish arising by substituting a vertex c of C by a $2K_1$, then $C - \{c\} \cup \{s\}$, where s was the only neighbor of c in S, is a neighborhood set of size t of G. Thus $\rho_n(G) \leqslant t$. Now in all previous cases, if we take t edges that connect S to C, we get a neighborhood-independent set of size t. In the cases where a vertex has been substituted by a K_2 or $2K_1$, we choose only one of the two edges from S to C involved and all the other edges from S to C. In the case of a starfish that is not fat, we take all edges from S to C. Thus we have found in all cases a neighborhood-independent set of size t, implying that $\alpha_n(G) \geqslant t$. And as $\alpha_n(G) \leqslant \rho_n(G)$, we have that $\alpha_n(G) = \rho_n(G) = t$.

Let us now note that an urchin (or fat urchin) of less than 3 ends is also a starfish (or fat starfish). Therefore if we assume without loss of generality that G is not a starfish, the only possibility remaining is that G is an urchin with at leas 3 ends.

If G is an urchin or fat urchin with partition (S, C, R), and $|S| = t \geqslant 3$, we shall see that $\alpha_n(G) = 1$ and $\rho_n(G) = 2$. As there is no universal vertex in G, then

$\rho_n(G) \geqslant 2$. Moreover, if we take two vertices of C, taking care of not taking any vertex from the substituting K_2 or $2K_1$ in case G is a fat urchin, we clearly obtain a neighborhood set. Hence clearly $\rho_n(G) = 2$. Now let us see that indeed we cannot have a neighborhood-independent set of size 2. This becomes clear if we observe that in all cases, if G is an urchin or a fat urchin, all vertices and edges are in at least the neighborhood of $t-1$ vertices of C. That is, except for the vertices in S (or, eventually, of the K_2 or $2K_1$ substituting a vertex of S), all the rest of the vertices are adjacent to all vertices in C, and these are adjacent to $t-1$ vertices of C. Moreover all edges between vertices of $R \cup C$ are in the neighborhood of t vertices of C, and all edges between vertices of S and C are in the neighborhood of $t-1$ vertices of C. Thus, if we take any two edges or vertices of G, as $t \geqslant 3$, then there must at least be one vertex of C that includes them both in its neighborhood. Therefore, $\alpha_n(G) = 1$. □

Theorem 3.22. *If G is a P_4-tidy graph, then it is neighborhood-perfect if and only if it is $\{$3-pyramid, 0-pyramid, $C_5\}$-free.*

Proof. If G is neighborhood-perfect, then it cannot contain as induced subgraph a 3-pyramid, 0-pyramid or C_5 because none of these graphs are neighborhood-perfect and the class of neighborhood-perfect graphs is hereditary. We must then only prove that if G does not contain those graphs then it must be neighborhood-perfect.

Suppose that G is a P_4-tidy graph which is not neighborhood-perfect. Then it must contain a minimally non-neighborhood-perfect graph as induced subgraph; let H be such subgraph. The minimality of H implies that it must be connected. If $\overline{H}$ is disconnected, then H is a minimally non-neighborhood-perfect graph with disconnected complement, which by Theorem 3.20 means that it must be $C_4 \vee 2K_1 =$ 3-pyramid , $C_6 \vee 3K_1$ or $P_6 \vee 3K_1$. But as the class of P_4-tidy graphs is hereditary, H must be P_4-tidy, which implies that it cannot be $C_6 \vee 3K_1$ or $P_6 \vee 3K_1$. This is because both graphs contain four vertices with at least two companion vertices, namely any consecutive four vertices of the

C_6 or the center vertices of the P_6, and therefore are not P_4-tidy. Hence if $\overline{H}$ is disconnected, then H can only be 3-pyramid.

Let us suppose now that both H and $\overline{H}$ are connected. As H is minimally non-neighborhood-perfect, then $\alpha_n(H)$ must be different from $\rho_n(H)$. Which means, by Theorem 3.21, that H must be a C_5 or an urchin or fat urchin with at least 3 ends. Lastly, if H is an urchin or fat urchin with at least 3 ends, then it must have $\alpha_n(H) = 1$ and $\rho_n(H) = 2$. But by Theorem 3.5, the only minimally non-neighborhood-perfect graphs with $\alpha_n(H) = 1$ are the 3-pyramid and 0-pyramid and the only one of these that is an urchin is the 0-pyramid.Therefore, as H is connected and co-connected, it must be a C_5 or a 0-pyramid.

We conclude that H must be isomorphic to 3-pyramid, C_5 or 0-pyramid and since, by construction, H is an induced subgraph of G, this proves the theorem. □

Corollary 3.23. *Let G be a P_4-sparse graph, then it is neighborhood-perfect if and only if it is {3-pyramid, 0-pyramid}-free.*

Proof. As the class of P_4-sparse graphs is subclass of the class of P_4-tidy graphs, this characterization is a direct consequence of Theorem 3.22 and the fact that the only forbidden induced subgraph of the class of neighborhood-perfect graphs restricted to P_4-tidy graphs that is not P_4-sparse is C_5. □

3.4 Tree-cographs

In this section we shall develop a characterization by forbidden induced subgraphs of those tree-cographs that are neighborhood-perfect. Tree-cographs were defined in Chapter 2. We shall work with the modular decomposition of a tree-cograph and strongly rely on the characterization of minimally non-neighborhood-perfect graphs with disconnected complement, given in Theorem 3.20.

Theorem 3.24. *If* G *is a connected and co-connected tree-cograph, then one of the following statements holds:*

1. G *is a tree and* $\rho_n(G) = \alpha_n(G) = \nu(G) = \tau(G)$*,*
2. G *is a connected co-tree and* $\rho_n(G) = 2$*.*

Proof. By the definition of tree-cographs, if G is connected and co-connected, then G must be a tree with connected complement or a connected co-tree.

If G is a tree then it is bipartite. It was already noted in [70, 89] that for any bipartite graph G, $\alpha_n(G) = \nu(G)$ and $\rho_n(G) = \tau(G)$, which by the König-Egerváry theorem implies that $\alpha_n(G) = \nu(G) = \tau(G) = \rho_n(G)$.

If G is a connected co-tree, then $\overline{G}$ has at least one leaf; that leaf and its only neighbor in $\overline{G}$ clearly form a neighborhood set of G of size 2. Moreover as G has connected complement, there cannot be a neighborhood set of size 1, for this would imply the existence of a universal vertex in G and an isolated vertex in $\overline{G}$. Hence $\rho_n(G) = 2$, proving the theorem. □

Corollary 3.25. *There are no connected and co-connected tree-cographs that are minimally non-neighborhood-perfect.*

Proof. If a graph G is connected and co-connected, then, by Theorem 3.24, G is a tree or a co-tree. Moreover a tree cannot be non-neighborhood-perfect. If G is a co-tree, then $\rho_n(G) = 2$, which means that if G is minimally non-neighborhood-perfect, then $\alpha_n(G)$ must be 1. But by Theorem 3.5, the only minimally non-neighborhood-perfect graphs with $\alpha_n(G) = 1$ are the 0-pyramid and the 3-pyramid, none of which are co-trees. Hence if G is a minimally non-neighborhood-perfect graph, G cannot be a connected and co-connected tree-cograph. □

Theorem 3.26. *If* G *is a tree-cograph, then* G *is neighborhood-perfect if and only if* G *is* $\{$*3-pyramid*$, P_6 \vee 3K_1\}$*-free.*

Proof. It is clear that if G is neighborhood-perfect, it cannot have 3-pyramid or $P_6 \vee 3K_1$ as induced subgraphs, for they are both minimally non-neighborhood-perfect graphs. We shall now prove that if it does not have those graphs as subgraphs, then it is neighborhood-perfect.

Suppose that G is not neighborhood-perfect. Hence G must contain an induced subgraph H that is minimally non-neighborhood-perfect. Clearly by minimality, H cannot be disconnected. If H has disconnected complement, then it is a minimally non-neighborhood-perfect graph with disconnected complement, and by Theorem 3.20, it must be $C_4 \vee 2K_1 =$ 3-pyramid, $C_6 \vee 3K_1$ or $P_6 \vee 3K_1$. But $C_6 \vee 3K_1$ is not a tree-cograph, because it is clearly neither a tree, nor a co-tree, nor the disjoint union of two tree-cographs nor the join of two tree-cographs. Thus if H has disconnected complement, it must be 3-pyramid or $P_6 \vee 3K_1$.

On the other hand if H has a connected complement, then it will be a connected and co-connected tree-cograph. However by Corollary 3.25, if H is minimally non-neighborhood-perfect, then it cannot be a connected and co-connected tree-cograph. Hence H can only be 3-pyramid or $P_6 \vee 3K_1$, and as H was by construction an induced subgraph of G, this proves the theorem. □

3.5 Subclasses of Hereditary Clique-Helly Graphs and Related Graph Classes: Relation with Clique-Perfectness

As was already stated in Chapter 1 it was shown in [69] that in the class of hereditary clique-Helly graphs, the parameters involved in the definitions of clique-perfectness coincide with those in the definition of neighborhood-perfectness. This clearly means that if a graph is hereditary clique-Helly, then it is clique-perfect if and only if it is neighborhood-perfect. This fact together with partial characterizations of clique-perfect graphs allows us to obtain several partial characterizations of neighborhood-perfect graphs.

For the sake of clarity, we shall first explicitly write the proof of the equivalence of both classes in the context of hereditary clique-Helly graphs.

Theorem 3.27 ([69]). *If* G *is a hereditary clique-Helly graph, then* $\alpha_n(G') = \alpha_c(G')$ *and* $\rho_n(G') = \tau_c(G')$ *for every induced subgraph* G' *of* G.

Proof. As proved by Prisner in [86], a graph is hereditary clique-Helly if and only if it is hereditary *maximal clique irreducible*,which means that every induced subgraph satisfies that every maximal clique has a *proper edge*, an edge that is not contained in any other maximal clique. The class of hereditary clique-Helly graphs is obviously hereditary, which means that it is sufficient to prove the equalities for G.

If $T = \{v_1, \dots, v_s\}$ is a clique transversal of G, then as every edge belongs to at least one maximal clique, T must also be a neighborhood set, implying $\tau_c(G) \geqslant \rho_n(G)$. Note that this last inequality holds for any graph G. Now let $N = \{w_1 \dots w_k\}$ be a neighborhood set of G. If e is a proper edge of a maximal clique C, then there must be at least one vertex w_i such that $e \in N[w_i]$, but then $w_i \in C$. As every maximal clique of G has at least one proper edge, then N must be a clique transversal of G. Therefore $\tau_c(G) \leqslant \rho_n(G)$, and $\tau_c(G) = \rho_n(G)$ follows.

Let S be a neighborhood-independent set, where $S = \{s_1, \dots, s_t\}$ and $s_j \in V(G) \cup E(G)$ for each $j \in \{1 \dots t\}$. Now for every s_j we take C_j to be the maximal clique in G that includes s_j. Clearly, by definition of neighborhood-independence, $\{C_1, \dots, C_t\}$ is a clique-independent set. Thus $\alpha_c(G) \geqslant \alpha_n(G)$. Note that this inequality hods for every graph G. On the other hand let $\{C_1, \dots, C_t\}$ is a set of pairwise independent maximal cliques of G. If we take a proper edge or each maximal clique, we will clearly obtain a neighborhood-independent set of edges. Therefore $\alpha_c(G) \leqslant \alpha_n(G)$, and $\alpha_c(G) = \alpha_n(G)$ follows. □

Now that we have written the proof, we shall proceed to use this result to prove several partial characterizations of neighborhood-perfect graphs.

3.5.1 Helly Circular-arc Graphs

In this subsection we will provide a characterization by forbidden induced subgraphs of neighborhood-perfect graphs restricted to the class of Helly circular-arc graphs. To do this we shall show the minimal forbidden subgraphs that characterize clique-perfectness of graphs in HCH and see that these graphs are also non-neighborhood-perfect.

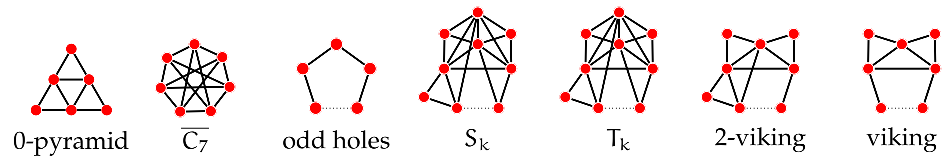

Figure 3.2: Minimal forbidden induced subraphs for the classes of clique-perfect and neighborhood-perfect graphs restricted to the class of HCA graphs. Dotted lines represent any induced path of odd lenght at least 1.

All of the following definitions and the theorem characterizing clique-perfectness of HCA graphs can be found in [13].

A viking is a graph G such that $V(G) = \{a_1, \dots, a_{2k+1}, b_1, b_2\}$, $k \geqslant 2$, $a_1 \dots a_{2k+1} a_1$ is an odd cycle with only one chord $a_2 a_4$; b_1 is adjacent to a_2 and a_3; b_2 is adjacent to a_3 and a_4, and there are no other edges in G.

A 2-viking is a graph G such that $V(G) = \{a_1, \dots, a_{2k+1}, b_1, b_2, b_3\}$, $k \geqslant 2$, $a_1 \dots a_{2k+1} a_1$ is an odd cycle with only two chords $a_2 a_4$ and $a_3 a_5$; b_1 is adjacent to a_2 and a_3; b_2 is adjacent to a_3 and a_4; b_3 is adjacent to a_4 and a_5, and there are no other edges in G.

Define the graph S_k, $k \geqslant 2$, as follows: $V(S_k) = \{a_1, \dots, a_{2k+1}, b_1, b_2, b_3\}$, $a_1 \dots a_{2k+1} a_1$ is an odd cycle with only one chord $a_3 a_5$; b_1 is adjacent to a_1 and a_2; b_2 is adjacent to a_4 and a_5; b_3 is adjacent to a_1, a_2, a_3 and a_4, and there are no other edges in S_k.

Define the graph T_k, $k \geqslant 2$, as follows: $V(S_k) = \{a_1, \dots, a_{2k+1}, b_1, b_2, b_3, b_4, b_5\}$, $a_1 \dots a_{2k+1} a_1$ is an odd cycle with only two chords, $a_2 a_4$ and $a_3 a_5$; b_1 is adjacent to a_1 and a_2; b_2 is adjacent to a_1,a_2 and a_3; b_3 is adjacent to a_1,a_2,a_3,a_4,b_2 and

b_4; b_4 is adjacent to a_3,a_4 and a_5; b_5 is adjacent to a_4 and a_5, and there are no other edges in T_k.

Then the following theorem holds:

Theorem 3.28 ([13]). *Let* G *be a HCA graph. Then* G *is clique-perfect if and only if it does not contain a* 0*-pyramid, an antihole of length seven, an odd hole, a viking, a* 2*-viking or one of the graphs* S_k *or* T_k*.*

Lemma 3.29 ([13]). *If* G *is an HCA graph that has an HCA representation with no two arcs covering the circle, then* G *is HCH.*

Finally, we shall state the characterization of the class of HCH graphs by forbidden induced subgraphs, due to Prisner, that was mentioned in Section 2.3.4.

Theorem 3.30 ([86]). *If* G *is a graph,* G *is hereditary clique-Helly if and only if it does not contain as an induced subgraph* 0*-pyramid,* 1*-pyramid,* 2*-pyramid, or* 3*-pyramid.*

Theorem 3.31. *If* G *is a Helly circular-arc graph, then* G *is neighborhood-perfect if and only if* G *is clique-perfect.*

Proof. It suffices to prove the following equivalence for each HCA graph H: H is minimally non-neighborhood-perfect if and only if H is minimally clique-imperfect. By Theorem 3.27, the equivalence holds whenever H is HCH. Since, by virtue of Theorem 3.28 and Theorem 3.30, the only non-HCH HCA minimally clique-imperfect graph is the 0-pyramid, which is also minimally non-neighborhood-perfect, it only remains to show that the only minimally non-HCH HCA minimally non-neighborhood-perfect graph is the 0-pyramid.

Let H be a non-HCH HCA minimally non-neighborhood-perfect graph. By Lemma 3.29 all HCA representations of H must have two arcs covering the circle. This clearly implies that $\rho_n(H) \leqslant 2$, because if we take in any HCA representation any vertices that represent two arcs that cover the whole circle, we have a neighborhood-covering set of size 2. This means that $\alpha_n(H) \leqslant 2$. If $\alpha_n(H) = 2$, then $\alpha_n(H) = \rho_n(H)$ and the theorem holds. Let us suppose then that

$\alpha_n(H) = 1$. But by Theorem 3.5 the only minimally non-neighborhood-perfect graphs with $\alpha_n(H) = 1$ are the 0-pyramid and 3-pyramid, and as was seen in [72] the 3-pyramid is not HCA. Hence H is the 0-pyramid, which completes the proof of the theorem. □

3.5.2 Gem-Free Circular-arc Graphs

In this subsection we shall prove a characterization of neighborhood-perfect graphs restricted to the class of gem-free circular-arc graphs. Circular-arc graphs were defined in Chapter 2 and the gem graph can be seen in Figure 2.1. There is a characterization of clique-perfect graphs restricted to the class of gem-free circular-arc graphs, in [19].

Theorem 3.32 ([19])**.** *If G is gem-free circular-arc, then G is clique-perfect if and only if it has no odd holes.*

Corollary 3.33. *If G is a gem-free circular-arc graph, then G is neighborhood-perfect if and only if G has neither odd holes nor 3-pyramid as induced subgraphs.*

Proof. The "only if" part of the statement is clear because odd holes and the 3-pyramid are both non-neighborhood-perfect. For the converse, we assume G is a circular-arc graph and has none of these as induced subgraphs, and prove that it is neighborhood-perfect. Since G is gem-free and 3-pyramid-free, it must be HCH, because the 0-pyramid, 1-pyramid and 2-pyramid all have an induced gem. Thus, by Theorem 3.32 it is clique-perfect, but this means that it must also be neighborhood-perfect, by Theorem 3.27. □

3.5.3 Subclasses of HCH Graphs

We shall now give two characterizations of neighborhood-perfect graphs restricted to two subclasses of HCH. Both are based in previously proven characterizations in clique-perfect graphs and are therefore immediate consequences of those characterizations and Theorem 3.27.

Let G be a graph and C be a cycle of G not necessarily induced. An edge of C is *non-proper* (or *improper*) if it forms a triangle with some vertex of C. If C has no improper edges, we will say that C is a *proper* cycle. Let us then define an r-*generalized sun* $r \geqslant 3$ as a graph G whose vertex set can be partitioned in two sets: a cycle C of r vertices, with all its non-proper edges $\{e_j\}_{j \in J}$ (where J can be empty) and a stable set $U = \{u_j\}_{j \in J}$, such that for each $j \in J$, u_j is only adjacent to the endpoint of e_j. An r-generalized sun is said to be *odd* if r is odd.

Diamond-Free Graphs

Theorem 3.34. *[10, 13] If G is a diamond-free graph, then it is clique-perfect if and only if no induced subgraph of G is an odd generalized sun or, equivalently, G has no proper odd cycles.*

Corollary 3.35. *If G is a diamond-free graph, then it is neighborhood-perfect if and only if no induced subgraph of G is an odd generalized sun or, equivalently, G has no proper odd cycles.*

Proof. The statement is a direct result of Theorem 3.34 and the fact that diamond-free graphs are HCH by Theorem 3.30. □

HCH Claw-Free Graphs

Theorem 3.36 ([12]). *If G is a hereditary clique-Helly and claw-free graph, then it is clique-perfect if and only if no induced subgraph of G is an odd hole or an antihole of length 7.*

Corollary 3.37. *If G is hereditary clique-Helly and claw-free, then it is neighborhood-perfect if and only if no induced subgraph of G is and odd hole or an antihole of length 7.*

3.5.4 Some Relations with Balanced Graphs

Balanced graphs were described in Chapter 1 as graphs with balanced clique-matrix. We state in this chapter some relations between the class of balanced graphs and neighborhood-perfect graphs. We begin by a theorem already mentioned in Chapter 1.

Theorem 3.38 ([8, 69])**.** *All balanced graphs are neighborhood-perfect.*

Proof. As was already stated in [20], it follows from Proposition 7 in [5] that balanced graphs are HCH. Moreover, by [8] every balanced graph G has $\tau_c(G) = \alpha_c(G)$, and as it is a hereditary class, this means it is clique-perfect. Therefore by Theorem 3.27, G must be neighborhood-perfect. □

We will now show two classes of graphs restricted to which neighborhood-perfect graphs coincide with balanced graphs, namely chordal graphs and paw-free graphs. The equivalence among statements 1, 3 and 4 for chordal graphs G was implicitly proved in [70]. In addition, the equivalence of statements 2 and 3 was implicitly stated in [10], the proof given here is due to Bonomo [11].

Theorem 3.39 ([11, 10, 70])**.** *If G is a chordal graph, then the following assertions are equivalent:*

1. *G is neighborhood-perfect.*
2. *G is clique-perfect.*
3. *G is balanced.*
4. *G is odd-sun-free.*

Proof. One of the main results of [70] is the equivalence between statements (1) and (3) for every chordal graph G. Moreover, in [70], the authors prove that if G is chordal, then it is a neighborhood-perfect graph, if and only if its clique-tree (K, E) is a balanced hypergraph. Note that, following their terminology, a clique-tree is defined as a hypergraph having as vertices the maximal cliques of G and as hyperedges the set of cliques that share a common vertex of G. Clearly if we take the incidence matrix of (K, E) and transpose it, we obtain the clique matrix of G, which is balanced if and only if (K, E) is. Thus if G is chordal, then it is neighborhood-perfect if and only if it is balanced. Hence, statements (1), (3) and (4) are equivalent.

The chain of inclusions balanced $\subseteq$ clique-perfect $\subseteq$ odd-sun-free, holds always, because we have already seen that odd-suns are not clique-perfect, and that balanced graphs are all clique-perfect. Hence, $(3) \Rightarrow (2) \Rightarrow (4)$. Since statements (3) and (4) are equivalent, this completes the proof of the theorem. □

Theorem 3.40. *If* G *is a paw-free graph, then the following assertions are equivalent:*

1. G *is neighborhood-perfect.*
2. G *is balanced.*
3. G *is perfect and HCH.*
4. G *has no odd holes and contains no induced* 3*-pyramid.*
5. *Each component of* G *is either bipartite or is the join of a complete bipartite graph and a complete graph.*

Proof. The equivalence of (2), (3), (4) and (5) was already proven in Theorem 3.3 of [20]. We shall then prove only the equivalence between (1) and (2). If G is balanced, then it is clearly neighborhood-perfect (Theorem 3.38). Conversely, if G is neighborhood-perfect then it cannot contain an odd hole nor an induced 3-pyramid, and, as a consequence of the equivalence of (4) and (2), G must be balanced. We have then proved the equivalence between (1) and (2), which completes the proof of the theorem. □

Chapter 4

Algorithmic and Complexity Results on Neighborhood-Perfect Graphs

In this chapter we will work with three types of problems. The first one is the problem of finding $\alpha_n(G)$ and $\rho_n(G)$ for a graph G. The next one is the problem of finding sets of vertices and edges that form a minimum neighborhood set and a maximum neighborhood-independent set. And finally we shall study the problem of recognizing neighborhood-perfect graphs. All three of these problems shall be examined in different graph classes. The first and second one were already mentioned and studied for several graph classes.

We shall begin this chapter by mentioning in Section 4.1 previously studied results on these problems. In Section 4.2 we shall present linear-time algorithms for the recognition problem when the input graph is restricted to be a P_4-tidy graph or a tree-cograph, as well as several polynomial-time algorithms that arise from characterizations seen in Chapter 3. In Section 4.3 we shall give two linear-time algorithms for the second problem mentioned (and therefore for the first one as well), when the input graph is a P_4-tidy or a tree-cograph. Finally in Section 4.4 we shall prove that the problems of determining $\alpha_n(G)$ and $\rho_n(G)$ are $\mathcal{NP}$-complete even if G is the complement of a bipartite graph.

4.1 Known Results

The problem of finding $\alpha_n(G)$ and $\rho_n(G)$ has been studied for graphs G in the classes of neighborhood-perfect chordal graphs and strongly chordal graphs, as well as in the class of cographs.

Theorem 4.1 ([70])**.** *There is a polynomial-time algorithm that, given any neighborhood-perfect chordal graph G, determines $\alpha_n(G)$ and $\rho_n(G)$.*

Theorem 4.2 ([27])**.** *There is a linear-time algorithm that, given a strongly chordal graph G with a strong elimination order, can compute $\alpha_n(G)$ and $\rho_n(G)$.*

Theorem 4.3 ([57])**.** *There is a linear-time algorithm that, given a cograph G, finds $\alpha_n(G)$ and $\rho_n(G)$.*

On the problem of determining the *optimal sets*, that is a minimum neighborhood set and a maximum neighborhood-independent set, we have found explicit results in interval graphs only. However the algorithm described in Theorem 4.3 can be easily modified to obtain said optimal sets in linear time for cographs.

Theorem 4.4 ([70])**.** *There is a linear-time algorithm, that given an interval graph G, determines a maximum neighborhood-independent set and a minimum neighborhood set of G.*

The time complexity of the problems of determining α_n and ρ_n have been studied in split graphs, planar graphs and line graphs.

Theorem 4.5 ([27])**.** *The problems of determining $\alpha_n(G)$ and $\rho_n(G)$ with G a split graph are $\mathcal{NP}$-complete.*

Theorem 4.6 ([56])**.** *The problems of determining $\alpha_n(G)$ and $\rho_n(G)$ with G a planar graph such that $\Delta(G) = 3$ are $\mathcal{NP}$-complete.*

Theorem 4.7 ([56]). *The problems of determining $\alpha_n(G)$ and $\rho_n(G)$ with G a line graph such that $\Delta(G) = 3$ are $\mathcal{NP}$-complete.*

Additionally a generalization of $\alpha_n(G)$ and $\rho_n(G)$ was defined in [65], of which we have already spoken briefly in Section 2.2. In this paper results on $\alpha_n(G, k)$ and $\rho_n(G, k)$ where given for G restricted to chordal graphs and strongly chordal graphs.

Theorem 4.8 ([65]). *The problems of determining $\alpha_n(G, k)$ and $\rho_n(G, k)$ for any fixed k are $\mathcal{NP}$-complete even if G is a chordal graph.*

Theorem 4.9 ([65]). *There is a linear-time algorithm, that given a strongly chordal graph G, determines $\alpha_n(G, k)$ and $\rho_n(G, k)$ for any fixed k, provided that a strong elimination order of G is given as input.*

A refinement of this result was given in [23], where a linear-time algorithm to find these parameters was presented if the input graph is restricted to the class of doubly chordal graphs without 0-pyramid (a superclass of strongly chordal graphs). Note that this algorithm does not require the strong elimination order to be given as an input, and that the best known algorithms for finding such orders have time complexity $\mathcal{O}(n^2)$ [93] or $\mathcal{O}(m \log(n))$ [83].

Theorem 4.10 ([23]). *There is a linear-time algorithm, that given a doubly chordal graph G, that does not have a 0-pyramid as an induced subgraph, determines $\alpha_n(G, k)$ and $\rho_n(G, k)$ for any fixed k.*

4.2 Recognition Algorithms

In this section we address the problem of recognizing neighborhood-perfect graphs and give several polynomial-time algorithms to solve it, restricted to certain graph classes. Our main results shall be that recognition problem is linear-time solvable when restricted to P_4-tidy graphs and tree-cographs.

4.2.1 P_4-tidy Graphs

Let us first remember that, as was said in Section 2.3.7, it follows from Theorem 2.2 that if h is an N-node of the modular decomposition tree of a P_4-tidy graph G, then $\pi(h)$ must be isomorphic to C_5, P_5, $\overline{P_5}$, a prime starfish or a prime urchin. Moreover in $\mathcal{O}(n_{\pi}(h))$ time it can be decided whether or not $\pi(h)$ is a starfish (resp. urchin) and, if affirmative, its partition can be found within the same time bound.

We shall present a linear-time algorithm that decides whether any given P_4-tidy graph is neighborhood-perfect or not. For this purpose, the algorithm performs a simple traversal of the modular decomposition tree of the input graph, which, we shall show, makes the algorithm terminate in $\mathcal{O}(n)$ time provided the modular decomposition tree is given as an input. The algorithm will strongly rely on the characterization by forbidden induced subgraphs proven in Theorem 3.22.

In order to simplify the recognition algorithm, we shall first define a boolean function $C : V(T(G)) \longrightarrow \{\mathtt{True}, \mathtt{False}\}$, where $T(G)$ is the modular decomposition tree of G, and, for each node h of $T(G)$, $C(h) = \mathtt{True}$ if and only if $G[h]$ contains an induced C_4. We shall prove that, given as input the modular decomposition tree $T(G)$ of any P_4-tidy graph G, Algorithm 1 can be implemented so as to compute $C(h)$ for each node h of $T(G)$ in $\mathcal{O}(n)$ overall time. Once we have proved so, we shall use Algorithm 1 as a subroutine in Algorithm 2, which recognizes those neighborhood-perfect graphs in the class of P_4-tidy graphs.

Algorithm 1: Computes $C(h)$ for every node of $T(G)$, with G P_4-tidy graph

```
Input: A $P_4$-tidy graph G and its modular decomposition tree $T(G)$
Output: The modular decomposition tree $T(G)$ with the value $C(h)$
        attached to each node $h$ of it, where $C(h) = \mathsf{True}$ if and only if
        $G[h]$ contains an induced $C_4$
1  Step 1:
2  Traverse the nodes of $T(G)$ in post-order, and in each node $h$ do:
3      if $h$ is a leaf then $C(h) := \mathsf{False}$
4      else if ($C(h')$ is True for any child $h'$ of $h$) or
5      ($h$ is a P-node having at least two nonleaf children) or
6      ($\pi(h)$ is $\overline{P_5}$) or
7      ($\pi(h)$ is a starfish or an urchin and any vertex of its body represents $2K_1$)
       then
8          $C(h) := \mathsf{True}$
9      else $C(h) := \mathsf{False}$
10 Step 2:
11     Output $C(h)$ for every node $h$ of $T(G)$
```

Algorithm 2: Recognition of neighborhood-perfectness of P_4-tidy

```
Input: A $P_4$-tidy graph $G$
Output: Determines whether or not $G$ is a neighborhood-perfect graph
Initialization: Build the modular decomposition tree $T(G)$ of $G$ and compute
        $C(h)$ for every node $h$ of $T(G)$ using Algorithm 1
1  Step 1:
2  Traverse every node $h$ of $T(G)$ in any order and do:
3      if $h$ is an N-node then
4          if $\pi(h)$ is a $C_5$ or an urchin with at least 3 ends then
5              output "G is not neighborhood-perfect" and stop
6          else if $\pi(h)$ is a fat starfish such that a vertex of its body represents $2K_1$ and
           $C(h_r)$ is True where $h_r$ is the only vertex of its head then
7              if $C(h_r)$ is True for $h_r$ the child representing the head of $\pi(h)$ then output
               "G is not neighborhood-perfect" and stop
8      else if $h$ is an S-node then
9          if $h$ has at least three nonleaf children then
10             output "G is not neighborhood-perfect" and stop
11         else if $h$ has exactly two nonleaf children $h_1$ and $h_2$ and at least one of $C(h_1)$
           and $C(h_2)$ is True then
12             output "G is not neighborhood-perfect" and stop
13 Step 2:
14     output "G is neighborhood-perfect"
```

Below, we prove that these two algorithms are indeed correct and run in linear time.

Theorem 4.11. *Algorithm 1 correctly computes $C(h)$ for every node h of any given modular decomposition tree $T(G)$ in $\mathcal{O}(n)$ time, whenever G is a P_4-tidy graph.*

Proof. Clearly the algorithm sets $C(h)$ correctly for each leaf h of $T(G)$. Let h be any nonleaf node of $T(G)$ and suppose, without loss of generality, that the algorithm correctly sets $C(h')$ for each of the nodes h' visited before h. It is then easy to check that if the algorithm sets $C(h)$ to True, $G[h]$ contains an induced C_4. Conversely, suppose that $G[h]$ contains an induced C_4 and we shall prove that the algorithm correctly sets $C(h)$ to True. Thus, if any vertex h' of $\pi(h)$ represents a graph containing an induced C_4, then $C(h')$ is set to True and consequently also $C(h)$ is set to True. Hence, we assume without loss of generality, that every vertex of $\pi(h)$ represents a C_4-free graph. Since $G[h]$ contains an induced C_4, Theorem 2.1 implies that $\pi(h)$ is $\overline{P_5}$, an edgeless graph, a complete graph, a starfish or an urchin. If $\pi(h)$ contains an induced C_4, necessarily $\pi(h)$ is isomorphic to $\overline{P_5}$ and the algorithm correctly sets $C(h)$ to True. Thus, we assume without loss of generality, that $\pi(h)$ is C_4-free. In particular, $G[h]$ is not $\overline{P_5}$. If there is a nonsimplicial vertex h' of $\pi(h)$ representing a non-complete graph, then $G[h]$ is a starfish or an urchin and h' is a vertex of the body representing a non-complete graph; if so, Theorem 2.2 implies that h' represents $2K_1$ and the algorithm correctly sets $C(h)$ to True. Hence, we assume without loss of generality, that every nonsimplicial vertex of $\pi(h)$ represents a complete graph. We conclude that each induced C_4 of $G[h]$ arises from two adjacent simplicial vertices h_1 and h_2 of $\pi(h)$, each of which represents a non-complete graph. Necessarily, h is a P-node and h_1 and h_2 are nonleaves. Also in this case the algorithm correctly sets C(h) to True. This completes the proof of the correctness of the algorithm.

As for the complexity of the algorithm, it is clear that each node is seen only once, and that every node is traversed after all of its children. Hence, as $T(G)$ has at most $2n$ nodes, the algorithm can easily be implemented to check for every node h if $C(h')$ is True for some child h' or if h is a P-node with at least two nonleaf children, all in $\mathcal{O}(n)$ time. Moreover, $\pi(h)$ is $\overline{P_5}$, an urchin or starfish only if h is an N-node. In $\mathcal{O}(n_\pi(h))$ time it can be checked if any N-node of a P_4-tidy graph is P_5, C_5, $\overline{P_5}$ or an urchin or starfish and in these cases find their partitions. Thus, it can be verified for all nodes h if $\pi(h)$ is $\overline{P_5}$ or if it is a starfish or urchin with a vertex of its body representing a $2K_1$, in $\mathcal{O}(\sum_{h \text{ N-node}} n_\pi(h))$ time. As was already stated in Chapter 2, the sum of $n_\pi(h)$ for all N-nodes is at most $2n$. Therefore the whole algorithm can be implemented in $\mathcal{O}(n)$ time. $\square$

Theorem 4.12. *Algorithm 2 correctly determines if a P_4-tidy graph G is neighborhood-perfect, in linear time. Moreover, it works in $\mathcal{O}(n)$ time if the modular decomposition tree of G is given as part of the input.*

Proof. In order to prove that Algorithm 2 correctly decides neighborhood-perfectness of any given P_4-tidy graph, we shall prove that it outputs that the graph is neighborhood-perfect if and only if it is $\{C_5, 0\text{-pyramid}, 3\text{-pyramid}\}$-free. This together with Theorem 3.22 will imply the correctness of the algorithm.

Suppose that Algorithm 2 outputs that G is not neighborhood-perfect. Hence the algorithm stopped in Step 1. If it stopped in line 5, then clearly G contains an induced C_5 or 0-pyramid. stopped in line 7 or line 10, then h is an S-node and consequently each of its nonleaf children represents a non-complete graph. On the one hand, if the algorithm stopped in line 7, then any set consisting of a pair of nonadjacent vertices of each of the three nonleaf children of h induces $2K_1 \vee 2K_1 \vee 2K_1 =$ 3-pyramid in G. On the other hand, if the algorithm stopped in line 10, then the vertices of an induced C_4 of the graph represented by h_1 or h_2 together with a pair of nonadjacent vertices of the graph represented by the other one induce $C_4 \vee 2K_1 =$ 3-pyramid in G as well. We conclude that if the algorithm outputs that G is not neighborhood-perfect, then G contains an induced C_5, 0-pyramid or 3-pyramid.

Let us now prove that, conversely, if G contains any of the three forbidden induced subgraphs, then the algorithm outputs that G is not neighborhood-perfect.

Suppose first that G contains an induced C_5 or an induced 0-pyramid. By Theorem 2.1, there is some N-node h of $T(G)$ such that $\pi(h)$ contains an induced C_5 or 0-pyramid. By Theorem 2.2, $\pi(h)$ is C_5 or an urchin with at least three ends and the algorithms outputs that G is not neighborhood-perfect in Line 5. Finally, let us consider the case when G contains an induced 3-pyramid. Let h be a node of $T(G)$ such that $G[h]$ contains an induced 3-pyramid but none of the graphs represented by its children does. Clearly h cannot be a P-node, so it must be an N-node or S-node. If h is an N-node and $G[h]$ contains an induced 3-pyramid, then $\pi(h)$ must be an urchin or a starfish. However, if $\pi(h)$ were an urchin, it would contain a 0-pyramid. Hence, without loss of generality, let us suppose that it is not an urchin. Suppose $\pi(h)$ is a starfish with partition (S, C, R) with the nodes of S and C being leafs of $T(G)$ and R consisting on a single node h_r. By hypothesis, clearly the 3-pyramid cannot be entirely in h_r, C, or S. Since every vertex of a graph represented by a node in S has degree at most 2 in $G[h]$, no vertices of graphs represented by nodes in S can be vertices of any induced 3-pyramid of $G[h]$. Now, as each vertex of a graph represented by a vertex of C are adjacent to every vertex of the graph represented by h_r, and 3-pyramid has no universal vertex, then each induced 3-pyramid must have at least two nonadjacent vertices belonging to graphs represented by a vertex of C. But this is only possible if $G[h]$ is a fat urchin where some node of C represents $2K_1$. If this is the case, then an induced 3-pyramid can only be formed if there is an induced C_4 in the graph represented by h_r. To conclude if h is an S-node, since 3-pyramid $= C_4 \vee 2K_1 = 2K_1 \vee 2K_1 \vee 2K_1$, the only two possibilities for $G[h]$ to have an induced 3-pyramid while none of its children have it, are that there are more than three children representing non-complete graphs or two children, one containing a C_4 and the other one representing a non-complete graph. In all cases the algorithm outputs that G is not neighborhood-perfect, which completes the proof of the correctness of the algorithm.

The time complexity of the algorithm can easily be seen to be $\mathcal{O}(n + m)$ and $\mathcal{O}(n)$ if the decomposition tree is given. It was already mentioned in Chapter 2 that the modular decomposition tree of P_4-tidy graphs can be found in linear time and within the same time bound a partition of the N-nodes that correspond to urchins or starfish can be given. It was already proven in Theorem 4.11 that Algorithm 1 runs in $\mathcal{O}(n)$ time. In Step 1 we traverse every node h of $T(G)$, and all the operations corresponding to each node h can be carried out in $\mathcal{O}(n_\pi(h))$ time once that $C(h)$ has been determined. Hence, as the sum of $n_\pi(h)$ over all nodes h is at most $2n$, the algorithm runs in $\mathcal{O}(n+m)$ time and even in $\mathcal{O}(n)$ time if the modular decomposition tree $T(G)$ is already given in the input. $\square$

4.2.2 Tree-cographs

In this subsection we shall present an algorithm, similar to the one proposed in Section 4.2.1, to decide neighborhood-perfectness of tree-cographs in linear time.

As was already pointed out in Chapter 2, the N-nodes of the modular decompositions of tree-cographs represent only trees and complement of trees. Moreover neighborhood-perfect tree-cographs were characterized in Theorem 3.26 as tree-cographs having no 3-pyramid or $P_6 \vee 3K_1$ as induced subgraphs. We shall use this characterization and the modular decomposition of tree-cographs to achieve a linear-time recognition algorithm.

We shall first define two functions defined on the nodes h of the modular decomposition tree $T(G)$ of a graph G. Let $P : V(T(G)) \longrightarrow \{\mathrm{True}, \mathrm{False}\}$, such that $P(h) = \mathrm{True}$ if and only if $G[h]$ has an induced P_6. And let $\alpha : V(T(G)) \longrightarrow \mathbb{N}$, such that $\alpha(h) = \alpha(G[h])$.

Algorithm 3 computes both $P(h)$ and $\alpha(h)$ for all nodes in a modular decomposition tree $T(G)$ of a tree-cograph. It computes as well $C(h)$ as was defined in Section 4.2.1, all in $\mathcal{O}(n)$ time, given the modular decomposition tree. It uses the fact that computing $\alpha(T)$ can be done in $\mathcal{O}(|V(T)|$ time), for any tree T [90].

Algorithm 3: Computes $\alpha(h)$, $P(h)$ and $C(h)$ for every node h of $T(G)$, with G a tree-cograph

```
Input: A $P_4$-tidy graph $G$ and its modular decomposition tree $T(G)$
Output: $C(h)$, $P(h)$ and $\alpha(h)$ for every node $h$ of $T(G)$
1  Step 1:
2  Traverse the nodes of $T(G)$ in post-order, and in each node $h$ do:
3      if $h$ is a leaf then $C(h) := P(h) :=$ False and $\alpha(H) := 1$
4      else if $h$ is a P-node with children $h_1, \dots, h_k$ then
5          $C(h) := \bigvee_{i=1}^{k} C(h_i)$, $P(h) := \bigvee_{i=1}^{k} P(h_i)$, $\alpha(h) := \sum_{i=1}^{k} \alpha(h_i)$
6      else if $h$ is a S-node with children $h_1, \dots, h_k$ then
7          $\alpha(h) := \max\{\alpha(h_i) : 1 \leqslant i \leqslant k\}$, $P(h) := \bigvee_{i=1}^{k} P(h_i)$,
8          if $h$ has at least two nonleaf children then $C(h) :=$ True
9          else $C(h) := \bigvee_{i=1}^{k} C(h_i)$
10     else if $\pi(h)$ is a tree with children $h_1, \dots, h_k$ then
11         compute $\alpha(G[h])$ in linear time and assign it to $\alpha(h)$ $C(h) :=$ False
12         if the longest path in $\pi(h)$ is of length at least 6 then $P(h) :=$ True
13         else $P(h) :=$ False
14     else if $\pi(h)$ is a co-tree with children $h_1, \dots, h_k$ then
15         $\alpha(h) := 2$, $P(h) :=$ False
16         if $\overline{\pi(h)}$ has an induced matching of size at least 2 then
17             $C(h) :=$ True
18         else $C(h) :=$ False
19 Step 2:
20     Output $C(h)$, $P(h)$ and $\alpha(h)$ for every node $h$ of $T(G)$
```

Algorithm 4 is a linear-time algorithm, that uses Algorithm 3, to determine whether any given tree-cograph is neighborhood-perfect.

Algorithm 4: Recognition of neighborhood-perfectness of tree-cograph

```
Input: A tree-cograph $G$
Output: Determines whether $G$ is a neighborhood-perfect graph
Initialization: Build the modular decomposition tree $T(G)$ of $G$ and compute
        $C(h)$, $P(h)$, and $\alpha(h)$ for every node $h$ of $T(G)$ using Algorithm 3
Step 1:
Traverse every node $h$ of $T(G)$ in any order and do:
    if $h$ is an S-node then
        if $h$ has at least three nonleaf children then
            output "$G$ is not neighborhood-perfect" and stop
        else if $h$ has exactly two nonleaf childen $h_1$ and $h_2$ then
            if $C(h_1)$ or $C(h_2)$ is True then
                output "$G$ is not neighborhood-perfect" and stop
            else if $P(h_1)$ is True and $\alpha(h_2) \geqslant 3$ or vice versa then
                output "$G$ is not neighborhood-perfect" and stop
    if $h$ is an N-node, with $\pi(h)$ a co-tree then
        if $\overline{G[h]}$ contains an induced matching of size at least 3 then
            output "$G$ is not neighborhood-perfect" and stop
Step 2:
    output "$G$ is neighborhood-perfect"
```

We shall proceed to prove that both Algorithm 3 and Algorithm 4 are both correct and run in the previously stated time bounds.

Theorem 4.13. *Algorithm 3 correctly computes $C(h)$, $P(h)$ and $\alpha(h)$ for every node h of a given modular decomposition tree $T(G)$ in $\mathcal{O}(n + m)$ time, whenever G is a tree-cograph.*

Proof. The nodes of $T(G)$ are traversed in post-order, meaning that when the algorithm computes the functions C, P, and α for h, all the children of h have already been processed. It is clear that if h is a leaf, the functions are correctly computed. Let us prove then that for each node h that is not a leaf, the functions are correctly computed, assuming they were correctly computed for the children of h.

If h is a P-node, then clearly the maximum independent set of $G[h]$ is the union of the maximum independent sets of each connected component, moreover it contains an induced P_6 or C_4 if and only if one of the connected components has one.

If h is an S-node, then clearly the maximum independent set of $G[h]$ is an independent set of one of the graphs represented by its children. It is as well clear that as P_6 has a connected complement, it must be contained in one of the connected components of $\overline{G[h]}$, which are the graphs represented by the children of h. As for the C_4, since it can be formed by the join of two $2K_1$, it can be and induced subgraph of $G[h]$ if and only if it is and induced subgraph of the graph represented by some of the children of h or if there are two nonleaf children of G (because the join of one non-edge from each of the graphs represented by them form an induced C_4). Thus the only case that remains to be considered is when h is an N-node.

If h is an N-node, with $\pi(h)$ a tree, then, as $G[h]$ is a tree, it cannot contain an induced C_4, and it contains an induced P_6 if and only if there are two vertices at distance 5 or more. If h is an N-node and $\pi(h)$ is a co-tree with connected complement, then $\alpha(G[h]) = 2$ because it cannot be greater than 2 ($\overline{\pi(h)}$ would contain a C_3) and if it were 1, then $\pi(h)$ would be complete and therefore have a disconnected complement. Similarly $G[h]$ cannot contain an induced P_6, because it has three independent vertices that would form a C_3 in the complement of $\pi(h)$. Finally as $C_4 = \overline{2K_2}$, $\pi(h) = G[h]$ contains an induced C_4 if and only if $\overline{\pi(h)}$ contains an induced matching of size 2.

To prove that the algorithm runs in $\mathcal{O}(n + m)$ time, we shall see that for every node of $T(G)$, it performs $\mathcal{O}(n_\pi(h))$ operations, except for the N-nodes h with $\pi(h)$ isomorphic to a co-tree, in which the number of operations is in $\mathcal{O}(n_\pi(h) + m_\pi(h))$. As mentioned in Chapter 2 the sum of $n_\pi(h)$ over all nodes of $T(G)$ is at most $2n$, since all edges in $\pi(h)$, for h an N-node are in one-to-one correspondence with edges of G, and two graphs represented by two different N-nodes are vertex-disjoint, the sum of $m_\pi(h)$ for all N-nodes with $\pi(h)$ a co-tree must be at most m.

It is clear that if h is a leaf, a P-node, or an S-node, then the number of operations is proportional to $n_\pi(h)$. If h is an N-node with $\pi(h)$ isomorphic to a tree, then using any of the algorithms in [80, 81, 90] a maximum cardinality independent set can be found in $\mathcal{O}(n_\pi(h))$ time. And using the algorithm, suggested by Dijkstra in the sixties and formally proved in [25], to find a maximum path in trees it can be easily tested if the longest path in $\pi(h)$ has size greater or equal to 6 in $\mathcal{O}(n_\pi(h))$ time. The last case to consider is the if h is an N-node, with $\pi(h)$ isomorphic to a co-tree. Because $\pi(h)$ has $\mathcal{O}((n_\pi(h)^2)$ edges, then in $\mathcal{O}(m_\pi(h))$ time it can be complemented. Once complemented, in $\mathcal{O}(n_\pi(h)$ time the size of the greatest induced matching can be determined using any of the algorithms in [48, 106, 55]. This fact together with the observations made in the last paragraph imply that the whole algorithm can be implemented to run in $\mathcal{O}(n+m)$ time. □

Theorem 4.14. *Algorithm 4 correctly determines whether any given tree-cograph G is neighborhood-perfect, in $\mathcal{O}(n+m)$ time.*

Proof. To prove the correctness of this algorithm, we shall apply the same reasoning as in the proof of Theorem 4.12, but using the subgraph characterization of neighborhood-perfect graphs among tree-cographs proved in Theorem 3.26. We will then prove that the algorithm outputs that the graph G is neighborhood-perfect if and only if G is $\{(P_6 \vee 3K_1), \text{3-pyramid}\}$-free.

Let see first that if the algorithm outputs that the graph is not neighborhood-perfect, then it must contain one of the forbidden induced subgraphs. It must stop in Step 1. If it stops in line 5, then clearly $G[h]$ must contain an induced 3-pyramid $= 2K_1 \vee 2K_1 \vee 2K_1$, if it stops in line 8 then it must contain an induced $C_4 \vee 2K_1 =$ 3-pyramid. Moreover if it stops in line 10, then one of the two children of h contains an induced P_6 and the other one has an independent set of size at least 3, implying that $G[h]$ contains an induced $P_6 \vee 3K_1$. Lastly if it stops in line 13, then $G[h]$ is a co-tree that includes a $\overline{3K_2} = 3$-pyramid.

To conclude the if and only if proof, suppose now that G contains one of the two forbidden induced subgraphs, and let us see that the algorithm must then output that G is not neighborhood-perfect. Clearly if G contains one of the forbidden induced subgraphs, then there must be a node h of $T(G)$ such that $G[h]$ contains the induced subgraph, but none of its children does. Clearly h cannot be a P-node. Moreover, h cannot be an N-node with $\pi(h)$ isomorphic to a tree, because both forbidden graphs have cycles. Thus h must be a S-node or an N-node with $\pi(h)$ isomorphic to a co-tree. If h is a S-node and $G[h]$ contains an induced 3-pyramid, then, as was shown in the proof of Theorem 4.12, either h has three nonleaf children or has exactly two nonleaf children one of which contains an induced C_4. On the other hand if h is an S-node but $G[h]$ contains an induced $P_6 \vee 3K_1$, then as both P_6 and $3K_1$ are not the join of any other graph, there must be two children of h, one representing a graph containing an induced P_6 and the other one a graph having an independent set of size at least 3. All of these cases are considered in lines 5, 8 and 10. Finally if h is an N-node, with $\pi(h)$ a co-tree, then clearly $G[h]$ cannot contain an induced $3K_1 \vee P_6$, because the complement of a $3K_1$ would be a C_3, and $G[h]$ is a co-tree. If i$G[h]$ contains an induced 3-pyramid, then it could only be because in the complement of $\pi(h)$ there is an induced $3K_2$, which is the same as saying that $\overline{\pi(h)}$ has an induced matching of size at least 3. Again this is tested in line 13. So we have proved that if G contains one of the forbidden induced subgraphs, then the algorithm outputs that G is not neighborhood-perfect, concluding the proof of the if and only if.

To see that the algorithm runs in $\mathcal{O}(n+m)$ time, we shall use the same argument as in Theorem 4.13. First recall that as was mentioned in Chapter 2 we can construct the modular tree in linear time and, as was already proven, run Algorithm 3 in linear time. Now, for every node h in $T(G)$, if h is a P-node or a N-node with $\pi(h)$ a tree, the algorithm does no operations. If h is an S-node, then it clearly can determine determine the number of children h_i of h and check the values of $C(h_i)$, $P(h_i)$ and $\alpha(h_i)$ for all of them, in $\mathcal{O}(n_\pi(h))$ time. Finally if h is an N-node, with $\pi(h)$ a co-tree, then, as $m_\pi(h) \in\mathcal{O}((n_\pi(h))^2)$, we can complement $\pi(h)$ in $\mathcal{O}(m_\pi(h))$ time. Once complemented, we can use any of the linear-time algorithms in [48, 106, 55], to compute a maximum induced matching of $\overline{\pi(h)}$ in $\mathcal{O}(n_\pi(h))$ time. Thus the algorithm makes at most a number of operations proportional to $n_\pi(h)$ for every node h and to $m_\pi(h)$ for the N-nodes with $\pi(h)$ a co-tree, which implies that it runs in $\mathcal{O}(n+m)$ time for the whole graph. □

4.2.3 Recognition in Other Graph Classes

In this subsection we shall present several results on the recognition problem of neighborhood-perfect graphs in different graphs classes, based on the characterizations proved in Section 3.5, namely we shall prove that it can be solved in polynomial time restricted to the classes of paw-free graphs, diamond-free graphs, claw-free HCH graphs, HCA graphs and balanced graphs.

Corollary 4.15. *There is a linear-time algorithm, that given any paw-free graph G, decides whether G is neighborhood-perfect.*

Proof. By Theorem 3.40, G is neighborhood-perfect if and only if each component of G is either bipartite or is the join of a complete bipartite graph and a complete graph. Based on this characterization it is clear that we can in linear time check if any component of G is bipartite or if the non universal vertices of the component form a complete bipartite graph. This algorithm was already proposed to recognize balanced paw-free graphs in [20]. □

Corollary 4.16. *There is a polynomial-time algorithm, that given any diamond-free graph G, determines whether G is neighborhood-perfect.*

Proof. As was proven in Corollary 3.35, in the class of diamond-free graphs, neighborhood-perfectness and clique-perfectness coincide. Moreover in [20], it was proven that restricted to the class of diamond-free graphs, clique-perfectness and balancedness are the same. Recalling from Chapter 2 that recognizing balanced graphs is polynomial-time solvable, recognizing neighborhood-perfect graphs in the class of diamond-free graphs can be done in polynomial time. □

Corollary 4.17. *There is a polynomial-time algorithm to determine, given any chordal graph* G*, if* G *is neighborhood-perfect.*

Proof. By Theorem 3.39, G is neighborhood-perfect if and only if it is balanced. Thus, as there is a polynomial-time algorithm to determine whether or not G is balanced, there is one to determine if it is neighborhood-perfect. □

Corollary 4.18. *There is a polynomial-time algorithm to determine, given any claw-free HCH graph* G*, if* G *is neighborhood-perfect.*

Proof. By Corollary 3.37, claw-free HCH neighborhood-perfect graphs are exactly claw-free HCH clique-perfect graphs. It was also proven in [12] that if G is claw-free HCH, then it is clique-perfect if and only if it is perfect. Thus, restricted to the class of claw-free HCH graphs, clique-perfect, neighborhood-perfect and perfect graphs coincide. Therefore as there is a polynomial-time algorithm to recognize perfect graphs, there is a polynomial-time algorithm to determine if G is neighborhood-perfect. □

Corollary 4.19. *There is a polynomial-time algorithm to determine, given a hereditary circular-arc graph* G*, if* G *is neighborhood-perfect.*

Proof. As was proved in Theorem 3.31, G is neighborhood-perfect if and only if it is clique-perfect. But in [13], a $\mathcal{O}(n^{19})$-time algorithm is given to recognize clique-perfectness of graphs in the HCA class, therefore the same algorithm can be used to recognize whether any given HCA graph is neighborhood-perfect. □

4.3 Algorithms for Computing Optimal Sets of Vertices and Edges

In this section we shall present two new linear-time algorithms to compute a maximum neighborhood-independent set, a minimum neighborhood set, a maximum 2-independent set, and a minimum dominating set of P_4-tidy and tree-cographs. In this subsection we will refer a maximum neighborhood-independent set, a minimum neighborhood set, a maximum 2-independent set and a minimum dominating set of a graph as *optimal sets* of the graph. As in the previous section, we shall strongly use the properties of the modular decomposition trees of these two classes.

First we shall present an algorithm that given a subroutine that computes the optimal sets of graphs represented by the N-nodes of the modular decomposition tree (meaning that it computes a maximum neighborhood-independent set, a minimum neighborhood-independent set, a domination sets), finds optimal sets for the graphs represented by all the remaining nodes of the modular decomposition tree. This algorithm will be used for both classes of graphs, changing only the routine that finds optimal sets for the graph represented by the N-nodes (which have a different characterization in each class). It is also interesting to note that given any other graph class with a known characterization of its modular decomposition tree, one needs only to find a routine that finds optimal sets for the graph represented by the N-nodes from optimal sets of its children, to obtain an algorithm that finds optimal sets in the whole graph.

Given a graph G and its modular decomposition tree $T(G)$, for any node h of $T(G)$, let $R_n(h)$ be a list of vertices of G that form a neighborhood set of $G[h]$ of minimum size, $A_n(h)$ be a list of vertices and edges forming a maximum neighborhood-independent set of $G[h]$, $A_2(h)$ be a list of vertices forming a 2-independent set of maximum size of $G[h]$, and $D(h)$ be a list of vertices of G constituting a minimum dominating set of $G[h]$. We will call these four lists, *optimal lists* for the node h. Algorithm 5 will show how to recursively obtain optimal lists for each node h, thus obtaining these lists for the root of $T(G)$, which we shall call $A_n(G)$, $R_n(G)$, $A_2(G)$ and $D(G)$, respectively. For this purpose, Algorithm 5 will assume that we have a subroutine that given any N-node and optimal lists for the children of the N-node, correctly obtains the lists for the N-node. In all the following algorithms, we shall denote the concatenation of lists $l_1, \dots, l_k$, with $1 \leqslant i \leqslant k$ as $\sum_{i=1}^{k} l_i$. To denote the concatenation of two lists l_1 and l_2, we will use $l_1 + l_2$. We will denote a list by listing its elements between '$\langle$' and '$\rangle$'; for instance, a list whose elements are x, y, z will be denoted by $\langle x, y, z\rangle$. If l is a list, we will denote by $l[i]$ its i-th element.

Below, we prove that Algorithm 5 correctly calculates the desired lists, given that the subroutine used to calculate the lists in the N-nodes works correctly. Moreover we shall prove that if the N-nodes' subroutine works in linear time with respect to $\pi(h)$ for a N-node h, then the Algorithm 5 works in linear time with respect to G.

Theorem 4.20. *Algorithm 5 obtains correctly* $A_n(G)$, $R_n(G)$, $A_2(G)$ *and* $D(G)$, *given that the subroutine for N-nodes is correct.*

Proof. The algorithm traverses $T(G)$ in post-order, meaning that before reaching a node h, all its children have their optimal lists already computed. It is clear that if h is a leaf, then $G[h]$ is a single vertex and then all optimal lists associated with h consist in precisely that vertex. Let us now see that if we suppose the algorithm correctly builds optimal lists for all the children $h_1, \dots, h_k$ of an S-node or P-node, then it correctly computes them for the node itself. If h is a P-node, then, since the graphs represented by its children are the components of $G[h]$, it is clear that all optimal sets required can be obtained by simply joining the lists of the optimal sets for the children. If h is an S-node, it is easy to see that each of the lists $A_2(h)$, $D(h)$, $A_n(h)$, $R_n(h)$ represent a dominating set, a 2-independent set, a neighborhood-independent set and a neighborhood set of $G[h]$, respectively. Moreover, since the lengths of these lists match the optimal values (according to Theorems 3.9 and 3.12, lemma 3.10, and corollaries 3.11 and 3.13), the lists build in this way are optimal lists. □

Algorithm 5: Computes $A_n(G)$, $R_n(G)$, $A_2(G)$, $D(G)$ of a graph G, if a subroutine to find optimal lists for the graphs represented by N-nodes of its modular decomposition tree is given

```
Input: A graph $G$
Output: $A_n(G)$, $R_n(G)$, $A_2(G)$ and $D(G)$
Initialization: Construct $T(G)$, the modular decomposition tree of $G$
1  Step 1:
2  Traverse the nodes of $T(G)$ in post-order, and in each node $h$ do:
3      if $h$ is a leaf, representing only $v \in V(G)$ then
4          $A_n(h) := \langle v \rangle$, $R_n(h) := \langle v \rangle$, $A_2(h) := \langle v \rangle$, $D(h) := \langle v \rangle$
5      else if $h$ is a P-node with children $h_1, \dots, h_k$ then
6          $R_n(h) = \sum_{i=1}^{k} R_n(h_i)$, $A_2(h) = \sum_{i=1}^{k} A_2(h_i)$, $D(h) = \sum_{i=1}^{k} D(h_i)$,
           $A_n(h) = \sum_{i=1}^{k} A_n(h_i)$
7      else if $h$ is an S-node with children $h_1, \dots, h_k$ then
8          $A_2(h) := \langle v \rangle$ where $v$ is an arbitrary vertex of $G[h]$
9          $D(h) :=$ a list of minimum length among $D(h_1), \dots, D(h_k)$, $\langle v_1, v_2 \rangle$
           for any $v_1 \in V(G[h_1])$ and $v_2 \in V(G[h_2])$
10         if $k = 2$ then
11             $A_n(h) := \langle (A_2(h_1)[i], A_2(h_2)[i]) : , 1 \leqslant i \leqslant$
               $\min\{|A_2(h_1)|, |A_2(h_2)|\} \rangle$
12         else
13             $A_n(h) := \langle (v_1, v_2) \rangle$ for any $v_1 \in V(G[h_1])$ and $v_2 \in V(G[h_2])$
14         $R^* :=$  a list of minimum length among
                     $D^*(h_1), \dots, D^*(h_k), R_n(h_1), \dots, R_n(h_k)$,
                     where $D^*(h_i) = D(h_i) + \langle v \rangle$ for any $v \in G[h] \setminus G[h_i]$
15         if $k = 2$ then
16             $R_n(h) := R^*$
17         else
18             $R_n(h) :=$ a list of minimum length between $R^*$ and $\{v_1, v_2, v_3\}$,
               where $v_i \in V(G[h_i])$ for $i \in \{1, 2, 3\}$
19     else if $h$ is an N-node then
20         Use a graph class specific subroutine to calculate $A_n(h)$, $R_n(h)$,
           $A_2(h)$ and $D(h)$
21 Step 2:
22     Output $A_n(G)$, $R_n(G)$, $A_2(G)$, $D(G)$
```

Lemma 4.21. *Let $c(h)$ be the number of edges made in line 11 if $k = 2$, for h and all the descendants of h in a modular decomposition tree $T(G)$. Hence, for every node h,*

$c(h) + \alpha_2(h) \leqslant n(h)$, *where* $\alpha_2(h) = \alpha_2(G[h])$.

Proof. To prove this statement, we shall use a structural induction in $T(G)$. First, let us see that for each leaf h, clearly $c(h) = 0$, $\alpha_2(h) = 1$, and $n(h) = 1$. Now, let us suppose that we have a node h, not a leaf, and that the statement holds for every child $h_i, 1 \leqslant i \leqslant k$ of h. If h is not an S-node, then clearly $c(h) = \sum_{i=1}^{k} c(h_i)$. As every $G[h_i] \subseteq G[h]$, the inequality $|I \cap V(h_i)| \leqslant \alpha_2(h_i)$ must hold for every 2-independent set I of $G[h]$. Hence, by the induction hypothesis, $c(h) + \alpha_2(h) \leqslant \sum_{i=1}^{k} c(h_i) + \sum_{i=1}^{k} \alpha_2(h_i) \leqslant \sum_{i=1}^{k} n(h_i) = n(h)$.

If h is an S-node and $k > 2$, then $\alpha_2(h) = 1$, implying $c(h)+\alpha_2(h) = (\sum_{i=1}^{k} c(h_i))+1 \leqslant \sum_{i=1}^{k} c(h_i) + \alpha_2(h) \leqslant \sum_{i=1}^{k} n(h_i) = n(h)$. Hence, suppose that h is an S-node with two children and suppose, without loss of generality, that $\alpha_2(h_1) \leqslant \alpha_2(h_2)$ and consequently $c(h) = c(h_1) + c(h_2) + \alpha_2(h_1)$. Thus, since $\alpha_2(h) = 1$ (because h is an S-node), then $c(h) + \alpha_2(h) = c(h_1) + c(h_2) + \alpha_2(h_1) + 1 \leqslant c(h_1) + c(h_2) + \alpha_2(h_1) + \alpha_2(h_2) \leqslant n(h_1) + n(h_2) = n(h)$. □

Theorem 4.22. *Algorithm 5 works in $\mathcal{O}(n + m)$ time, if the subroutine for N-nodes works in $\mathcal{O}(n_\pi(h) + m_\pi(h))$ time, for every N-node h.*

Proof. All nodes are traversed exactly once, so let us see that for every leaf, S-node and P-node, the algorithm performs $\mathcal{O}(n_\pi(h))$ operations. If h is a leaf, then it only creates four lists of size 1. If h is a P-node, then the algorithm concatenates four times $n_\pi(h)$ lists. Which, if we suppose is done by loosing the original lists, can be achieved in $\mathcal{O}(n_\pi(h))$ time. If h is an S-node, then to obtain $D(h)$, $A_2(h)$, $A_n(h)$, and $R_n(h)$, clearly it performs at most $\mathcal{O}(n_\pi(h))$ operations plus the time of building the edges in line 11, if h is an S-node with exactly two children. Since, by Lemma 4.21, the number of edges made in all S-nodes is $\mathcal{O}(n)$, the sum of $n_\pi(h)$ for every node h in $T(G)$ is at most $2n$, the sum of all $m_\pi(h)$ for all N-nodes is at most m, and finding the modular decomposition tree can be done in $\mathcal{O}(n + m)$ time, the whole algorithm can be implemented to run in $\mathcal{O}(n + m)$ time. □

4.3.1 P_4-tidy Graphs

In this subsection we shall show an algorithm to find in $\mathcal{O}(n_{\pi(h)})$ time the optimal sets for an N-node of the modular decomposition tree $T(G)$ of a P_4-tidy graph G.

Algorithm 6: Computes $A_n(h)$, $R_n(h)$, $A_2(h)$, $D(h)$, for any given N-node h of the modular decomposition tree $T(G)$ of a P_4-tidy graph G

```
Input: A N-node $h$ of a modular decomposition tree of a $P_4$-tidy graph $G$
Output: $A_n(h)$, $R_n(h)$, $A_2(h)$ and $D(h)$
1  Step 1:
2      if $\pi(h)$ is isomorphic to $C_5 = v_1 \dots v_5 v_1$ then
3          $A_n := \langle v_1v_2, v_4v_5\rangle$, $R_n(h) := \langle v_1, v_3, v_5\rangle$, $A_2(h) := \langle v_1\rangle$, $D(h) := \langle v_1, v_2\rangle$
4      else if $\pi(h)$ is isomorphic to $P_5 = v_1 \dots v_5$ then
5          $A_n := \langle v_1v_2, v_4v_5\rangle$, $R_n(h) := D(h) := \langle v_2, v_4\rangle$, $A_2(h) := \langle v_1, v_4\rangle$
6      else if $\pi(h)$ is isomorphic to $\overline{P_5}$ with $\overline{\pi(h)} = v_1 \dots v_5$ then
7          $A_n := \langle v_1v_5, v_2v_3\rangle$, $R_n(h) := D(h) = \langle v_1, v_2\rangle$, $A_2(h) = \langle v_1\rangle$
8      else if $\pi(h)$ is a starfish with partition $(S, C, R)$ where $C = \{c_1, \dots, c_k\}$,
       $S = \{s_1, \dots, s_k\}$ and $c_1s_1, \dots, c_ks_k$ are the legs of $\pi(h)$ then
9          Let $v_i \in V(G[c_i])$ and $w_i \in V(G[s_i])$ for each $i \in \{1, \dots, k\}$
10         $A_2(h) := \langle w_1, \dots, w_k\rangle$, $D(h) := \langle v_1, \dots, v_k\rangle$, $A_n(h) := \langle v_1w_1, \dots, v_kw_k\rangle$,
           $R_n(h) := \langle v_1, v_2, \dots, v_k\rangle$
11         if $\pi(h)$ is a fat starfish with $c_i \in C$ representing $2K_1$ then
12             Replace $v_i$ in $R_n(h)$ with $w_i$
13     else if $\pi(h)$ is an urchin with partition $(S, C, R)$ where $C = \{c_1, \dots, c_k\}$,
       $S = \{s_1, \dots, s_k\}$ and $c_1s_1, \dots, c_ks_k$ are the legs of $\overline{\pi(h)}$ then
14         $A_2(h) := \langle v_1\rangle$, $D(h) := \langle v_1, v_2\rangle$, $A_n := \langle v_1w_2\rangle$ and $R_n(h) := \langle v_1, v_2\rangle$ for any
           $v_1 \in V(G[c_1])$, $v_2 \in V(G[c_2])$ and $w_2 \in V(G[s_2])$.
15 Step 2:
16     Output $A_n(h)$, $R_n(h)$, $A_2(h)$, $D(h)$
```

Theorem 4.23. *Algorithm 6 correctly finds* $A_n(h)$, $R_n(h)$, $A_2(h)$, $D(h)$, *for any given N-node* h *of the modular decomposition tree* $T(G)$ *of any* P_4*-tidy graph* G.

Proof. It can be checked by simple inspection that if $\pi(h)$ is isomorphic to C_5, P_5, or $\overline{P_5}$, optimal lists are chosen (recall that if this is the case $G[h] = \pi(h)$). If $\pi(h)$ is a starfish, then clearly the lists $A_2(h)$, $D(h)$, $A_n(h)$ and $R_n(h)$ computed by the algorithm correspond to a 2-independent set, a dominating set, a neighborhood-independent set and a neighborhood set of $G[h]$, respectively. Moreover, such lists are optimal lists because $A_2(h)$ has the same length as $D(h)$, and $A_n(h)$ has the same length as $R_n(h)$. If $\pi(h)$ is an urchin, clearly we cannot dominate all vertices with only one vertex, but if we take two vertices belonging to different graphs represented by vertices of C, we obtain a minimum dominating set, as well as a minimum neighborhood set. In an urchin all vertices are at most at distance two from each other, and thus all 2-independent sets of $G[h]$ have size 1. It is also easy to see that if $\pi(h)$ is an urchin then no two edges can be neighborhood-independent; so, the maximum neighborhood-independent set must be composed of at most one edge. Hence, if $\pi(h)$ is an urchin, then the lists $A_2(h)$, $D(h)$, $A_n(h)$ and $R_n(h)$ build by the algorithm are optimal lists. Therefore, as G is P_4-tidy, we have seen that for all possible scenarios the algorithm correctly computes the optimal sets. □

Theorem 4.24. *Algorithm 6 works in $\mathcal{O}(n_\pi(h))$ time, for h an N-node in the modular decomposition tree of any P_4-tidy graph G.*

Proof. As we was already seen in Chapter 2, if G is a P_4-tidy, we can decide in $\mathcal{O}(n_\pi(h))$ time whether $\pi(h)$ is isomorphic to P_5, C_5, $\overline{P_5}$, or is a starfish or urchin, and in the latter two cases obtain its decomposition. It is clear that if $\pi(h)$ is isomorphic to a P_5, C_5, $\overline{P_5}$, the algorithm performs a constant number of operations. If $\pi(h)$ is a starfish, then once it has obtained C and S, and determined if there is a replaced vertex of C (all in $\mathcal{O}(n_\pi(h))$) time, it does only constant time assignments and it generates $|C|$ edges, all of which can be done in $\mathcal{O}(n_\pi(h))$ time. Finally if $\pi(h)$ is an urchin, then once again it performs a constant number of operations. Therefore in all possible cases it runs in $\mathcal{O}(n_\pi(h))$ time. □

4.3.2 Tree-cographs

In this subsection we will present an algorithm to find the optimal sets of N-nodes in a modular decomposition tree of a tree-cograph. To this purpose we shall first give a characterization of co-trees with $\alpha_n > 1$. This characterization will allow us to easily identify these graph and find a neighborhood-independent set of maximum size, all in linear time.

Lemma 4.25. *If* G *is a graph, then* $\alpha_n(G) > 1$ *if and only if* G *has two edges* xy *and* wz *such that* $\{x, y, w, z\}$ *is a total dominating set of* $\overline{G}$*.*

Proof. By definition, $\alpha_n(G) > 1$, if and only if there are two neighborhood-independent edges xy and wz in G. Moreover, any two edges xy and zw of G, neighborhood-independent satisfy that every vertex is at least nonadjacent in G to at least one vertex in $\{x, y, w, z\}$ different from itself, or equivalently $\{x, y, w, z\}$ is a total dominating set of $\overline{G}$. □

Lemma 4.26. *If* G *is a co-tree with* $\alpha_n(G) > 1$*, then* T' *must be a path, where* T *is* $\overline{G}$ *and* T' *is the graph* T*, with all its leafs erased.*

Proof. If G is a co-tree, then clearly T is a tree and hence T' must be also a tree. Let us suppose by contradiction that T' is not a path. As paths are trees with at most two leaves, then T' has by our supposition three different leafs x, y, z and, as $|V(T')| > 2$, these three vertices must form an independent set of T' (and thus of T). The fact that these three vertices are in T' implies that they were not leaves in T, but as they are leaves of T', then they must have been adjacent to leaves in T. Given a tree, all vertices adjacent to leafs must be in all total dominating sets, because they are the only vertices that can dominate the leafs. Hence x, y and z are in all total dominating sets of T. Since $\alpha_n(G) > 1$, Lemma 4.25 implies that there must be a vertex w such that $\{x, y, w, z\}$ is a total dominating set of T and without loss of generality xy, wz are edges of G. But this clearly implies a contradiction, because z is not strongly dominated in T by $\{x, y, w, z\}$. The contradiction came from the supposition that T' was not a path. □

Before stating the characterization, we shall present an inequality that will be used in the proof of Theorem 4.28.

Theorem 4.27 ([30]). *The following inequality holds for every tree* T*:*

$$\gamma_t(T) \geqslant (n(T) + 2 - l(T))/2.$$

Where $n(T)$ *is* $|V(T)|$, $l(T)$ *is the number of leafs of* T *and* $\gamma_t(T)$ *is the total dominating number of* T.

Theorem 4.28. *If* G *is a co-tree, then* $\alpha_n(G) > 1$ *if and only if* T' *is either* P_2, P_3, P_4, *or* P_5 *or* P_6 *with no central vertex of* T' *adjacent to a leaf of* T, *where* $T = \overline{G}$ *and* T' *is the graph* T *with all its leaves erased.*

Proof. Let us first prove that if G is a co-tree with $\alpha_n(G) > 1$, then T' is as described above. By Lemma 4.26, T' must be a path. Clearly T' cannot have only 1 vertex, because T would be a star and $\alpha_n(G)$ would be one. As we have already seen in Lemma 4.25, $\gamma_t(T) \leqslant 4$ if $T = \overline{G}$. Thus, Theorem 4.27 implies that $6 \geqslant n(T) - l(T)$, but $n(T) - l(T) = n(T')$. Therefore T' is P_i with $2 \leqslant i \leqslant 6$. If $T' = P_5$ or $T' = P_6$, then suppose by contradiction that there is a leaf in T adjacent to any central vertex of T'. As was already mentioned in the proof of Lemma 4.26, this means that there is a central vertex of T' that must be in every total dominating set of T, this is also always true for both leafs of T'. But then there cannot be a total dominating set of T of size 4, because all three vertices are nonadjacent in T and there is no vertex that is adjacent to all three at the same time. This leads to a contradiction because we have already proved that $\gamma_t \leqslant 4$. Hence no central vertex of T' can be adjacent to a leaf of T if T' is a P_5 or P_6.

To prove the converse implication, if T' is P_2, P_3 or P_4, simply take all vertices of T' plus two, one, or zero leaves of T, respectively, adjacent to different leaves of T', and we shall have a total dominating set of T of size 4. If this set is $\{x, y, w, z\}$, then clearly we can always take xy and wz to be non-edges of T and thus edges of G, and by Lemma 4.25, $\alpha_n(G) > 1$. If T' is P_5 or P_6, we can take all vertices of T', except for the central vertices of the path. As no central vertex is adjacent to leafs of T, then clearly these four vertices must be a total dominating set of T. Once again it is easy to check that we can find two non-edges of T among these four vertices, and therefore $\alpha_n(G) > 1$. $\square$

Corollary 4.29. *It is easy to decide in $\mathcal{O}(n + m)$ time whether a co-tree G has $\alpha_n(G) > 1$ and if so find a neighborhood-independent set of G size 2.*

Proof. We use the characterization presented in Theorem 4.28. We can easily complement G and remove the vertices with degree 1. If the resulting tree is a path of length 2 to 6, then $\alpha_n(G) > 1$ and, following the instructions of the proof of Theorem 4.28, we can obtain the two neighborhood-independent edges of G. As G is a co-tree, $m \in \mathcal{O}(n^2)$ meaning that we can complement G in $\mathcal{O}(m)$ time. Deciding whether a tree becomes a path of bounded size by removing its leaves and, if so, also computing the corresponding path, can all be done in $\mathcal{O}(n)$. Finally obtaining the edges following the instructions of Theorem 4.28 can be easily done in $\mathcal{O}(n)$ time. $\square$

Now that we have given this characterization, we shall prove that Algorithm 7 finds the optimal sets for an N-node of the modular decomposition tree of a tree-cograph, all in $\mathcal{O}(n_\pi(h)+m_\pi(h))$ time. In line 8, we check if $\overline{\pi(h)}$ has a total dominating set of size 2. Let us see why this allows us to find the 2-independent set $\pi(h)$ that we need.

Lemma 4.30. *If G is a graph, then $\{v_1, v_2\} \subseteq V(G)$ is a 2-independent set of G if and only if it is a total dominating set of $\overline{G}$.*

Proof. The set $S = \{v_1, v_2\}$ is a 2-independent set of G if and only if $N_G[v_1] \cap N_G[v_2] = \emptyset$. But this means that in $\overline{G}$ no vertex can be nonadjacent to both v_1 and v_2, which is to say that all vertices of $\overline{G}$ must be adjacent to v_1 or v_2. Therefore S is a 2-independent set of G if and only if S is a total dominating set of $\overline{G}$ of size 2. $\square$

Algorithm 7: Computes $A_n(h)$, $R_n(h)$, $A_2(h)$, $D(h)$, for a given N-node h of the modular decomposition tree $T(G)$ of a tree-cograph G

```
Input: A N-node $h$ of a modular decomposition tree of a tree-cograph $G$
Output: $A_n(h)$, $R_n(h)$, $A_2(h)$ and $D(h)$
Step 1:
    if $\pi(h)$ is a tree then
        $A_2(h) :=$ a maximum 2-independent set of $G[h]$
        $D(h) :=$ a minimum dominating set of $G[h]$
        $A_n(h) :=$ a maximum matching of $G[h]$
        $R_n(h) :=$ a minimum vertex cover of $G[h]$
    else if $\pi(h)$ is a co-tree then
        if $\overline{\pi(h)}$ has a total dominating set of size 2 then $A_2(h) :=$ a total dominating set of $\overline{G[h]}$ of size 2
        else $A_2(h) := \langle v_1 \rangle$ for any $v_1 \in G[h]$
        if $\alpha_n(\pi(h)) > 1$ then $A_n(h) := \{e_1, e_2\}$ with $e_1, e_2$ neighborhood-independent edges of $G[h]$
        else $A_n(h) = \{e_1\}$ with $e_1$ any edge of $\pi(h)$
        $D(h) := \langle v_l, v_n \rangle$, $R_{\setminus}(h) := \langle v_l, v_n \rangle$, with $v_l$ a leaf of $\overline{G[h]}$ and $v_n$ its only neighbor in $\overline{G[h]}$
Step 2:
    Output $A_n(h)$, $R_n(h)$, $A_2(h)$, $D(h)$
```

Theorem 4.31. *Algorithm 7 correctly finds* $A_n(h)$, $R_n(h)$, $A_2(h)$ *and* $D(h)$*, for any given N-node* h *of the modular decomposition tree of a tree-cograph* G.

Proof. If G is a tree-cograph, then an N-node h of its modular decomposition is a tree with connected complement or a connected co-tree. In both cases $\pi(h)$ is isomorphic to $G[h]$, thus we can find the optimal sets analyzing $\pi(h)$. If $\pi(h)$ is a tree, then as was already seen in [70] a maximum matching of $G[h]$ is also a maximum neighborhood-independent edge set and a minimum vertex cover is a minimum neighborhood-cover set. Hence if $\pi(h)$ is a tree, then clearly the algorithm computes the correct values for the optimal sets. On the other hand if $\pi(h)$ is a co-tree, then as was seen in Lemma 4.30, if we find a total dominating set of size 2 in $\overline{G[h]}$, we will have a 2-independent set of size 2 of $G[h]$. Clearly a co-tree cannot have an independent set of size three, thus $\alpha_n(\pi(h)) \leqslant 2$. Clearly if there are no 2-independent sets of size 2, then any node is a maximum 2-independent set. It was already stated in Corollary 4.29 that there is a linear-time algorithm to determine if $\alpha_n(\pi(h)) > 1$ and if this is the case to find a neighborhood-independent set of size 2. Thus in line 10, we correctly obtain $A_n(h)$. Note that $\alpha_n(G[h]) \leqslant 2$, because if we take a leaf of $\overline{G[h]}$ and its only neighbor in $\overline{G[h]}$, we clearly have a neighborhood set as well as a dominating set of $G(h)$. Moreover if there were a dominating set or neighborhood set of size 1, then that would mean an isolated vertex in $\overline{G[h]}$, which would contradict the fact that it is a tree. □

Theorem 4.32. *Algorithm 7 can be implemented to run in* $\mathcal{O}(n_\pi(h) + m_\pi(h))$ *time.*

Proof. It is clear that in linear time it can be determined if $\pi(h)$ is a tree. Moreover, if $\pi(h)$ is not a tree, then it must be a co-tree because all N-nodes of a tree-cograph are trees or co-trees. If $\pi(h)$ is a tree, algorithms for finding minimum vertex cover sets, minimum dominating sets and maximum matchings in linear time can be found in [80, 81, 90]. Obtaining a 2-independent maximum set of a tree can also be done efficiently with an algorithm very similar to the one mentioned in [81] for independent sets. We explicitly state here, for the sake of completion, this linear-time algorithm for finding a 2-independent maximum set of a tree T:

Given a tree T, we regard it as a directed tree with an arbitrary root vertex r and traverse its vertices in post-order. For every vertex i, we determine $\text{Use}(i)$ and $\text{NUse}(i)$, where $\text{Use}(i)$ is a maximum 2-independent set using vertex i and $\text{NUse}(i)$ is defined analogously but without using i. Clearly, if i is not a leaf, then $\text{Use}(i) = i \cup \bigcup(\text{ NUse(j) } : \text{j is a child of i })$ and $\text{NUse}(i) = \bigcup(\max\{\text{ Use(j) , NUse(j) } : \text{j is child of i })\}$, where $\max\{A, B\}$ denotes a set with maximum number of vertices among A and B. If i is a leaf, then clearly

$\text{Use}(i) = \{i\}$ and $\text{NUse}(i) = \emptyset$. Hence, $\text{Use}(i)$ and $\text{NUse}(i)$ for all vertices i can be determined in overall linear-time. Finally, $\max\{\text{Use}(i), \text{NUse}(i)\}$, which is a maximum 2-independent set of T, can be found in linear time.

Hence, using the algorithms mentioned above, which clearly run in $\mathcal{O}(n_\pi(h))$ time, we can obtain corresponding to a node h whenever $\pi(h)$ is a tree. If $\pi(h)$ is a co-tree, then, as was already mentioned, we can complement it in $\mathcal{O}(m_\pi(h))$ time, then using any of the algorithms mentioned in [68, 30, 63], we can obtain a maximum total dominating set of $\overline{\pi(h)}$, and if it is of size 2, we can obtain the corresponding total dominating set of $\overline{G[h]}$ and assign it to $A_2(h)$ (bearing in mind that $\pi(h)$ and $G[h]$ are isomorphic). Using the algorithm mentioned in Corollary 4.29, we can find in $\mathcal{O}(n_\pi(h) + m_\pi(h))$ time a maximum neighborhood-independent set of $G[h]$. Finally, having already complemented $\pi(h)$, finding a leaf of $\overline{G[h]}$ and its neighbor can be done easily in linear time. Therefore if $\pi(h)$ is a co-tree, the algorithm can also be implemented to run in $\mathcal{O}(n_\pi(h) + m_\pi(h))$ time. □

4.4 Complexity Results

As was already stated, to our knowledge no results have been explicitly given on the computational complexity of the recognition problem of the whole class of neighborhood-perfect graphs. On the other hand the problems of determining α_n and ρ_n were already proven to be $\mathcal{NP}$-complete even for the class of split graphs in [27]. In this section we present a prove of $\mathcal{NP}$-hardness of these two problems in the class of co-bipartite graphs. If X and Y are disjoint sets and $F \subseteq X \times Y$, we shall denote by (X, Y, F) the co-bipartite graph with vertex set $X \cup Y$ where X and Y are cliques and the edges between X and Y are those in F.

Theorem 4.33. *It is $\mathcal{NP}$-hard to determine the neighborhood-independence number in co-bipartite graphs.*

Proof. We shall prove the $\mathcal{NP}$-hardness of the problem, by showing a polynomial reduction of the problem of determining the size of a maximum independent set of a graph H. For that purpose, given any graph H, we will define a co-bipartite graph G such that $\alpha_n(\mathrm{G}) = \alpha(\mathrm{G})$.

Given any graph $\mathrm{H} = (\mathrm{V}, \mathrm{E})$, let $\mathrm{G} = (\mathrm{X}, \mathrm{Y}, \mathrm{F})$ where $\mathrm{X} = \{v' \colon v \in \mathrm{V}\}$, $\mathrm{Y} = \mathrm{V} \cup \mathrm{E}$ and $\mathrm{F} = \{v'e \colon v \in \mathrm{V}, e \in \mathrm{E} \text{ and } v \text{ is incident to } e\} \cup \{v'v \colon v \in \mathrm{V}\}$; that is, we connect every vertex in Y to its copy in X and every edge in Y to the copies of its endpoints in X. Let us first note that as there are no isolated vertices in G, then in order to determine the neighborhood-independence number we can restrict our attention to those neighborhood-independent sets consisting only of edges. Moreover, being X and Y cliques, there is some maximum neighborhood-independent set having all its edges in F.

Given an independent set $\mathrm{S} \in \mathrm{V}$ of H, let I be the subset of F defined by $\mathrm{I} = \{v'v \colon v \in \mathrm{S}\}$. It is easy to see that I is a neighborhood-independent set because given two different edges $v'v$ and $w'w$ of I, there is no vertex adjacent to all four vertices. In fact, the only vertices in X adjacent to v and w are v' and w' respectively and if there were an element of Y adjacent to v, v', w, and w', then it would necessarily be an edge e of H joining v to w, which contradicts the fact that S is an independent set of H. This contradictions proves that I is a neighborhood-independent set and hence $\alpha_n(\mathrm{G}) \geqslant \alpha(\mathrm{H})$.

Conversely, let I be a neighborhood-independent set of edges in G such that $\mathrm{I} \subseteq \mathrm{F}$. We shall see that $\mathrm{S} = \{v \in \mathrm{V} \colon v'y \in \mathrm{I}\}$ is an independent set of H. Suppose, for a contradiction, that there is an edge e of H joining two vertices v and w of S. By definition, there are $y_1, y_2 \in \mathrm{Y}$ such that $v'y_1, w'y_2 \in \mathrm{F}$ and, by construction, e is adjacent in G to all the four endpoints of $v'y_1$ and $w'y_2$, which contradicts the fact that F is a neighborhood-independent set. This contradictions shows that S is an independent set of H and therefore $\alpha(\mathrm{H}) \geqslant \alpha_n(\mathrm{G})$. This completes the proof of the polynomial reduction of the maximum independent set problem to the maximum neighborhood-independent set problem in co-bipartite graphs. $\square$

To prove the following theorem we will use the following result from [42].

Theorem 4.34 ([42]). *Given a graph* G*, it is* $\mathcal{NP}$*-hard to approximate the Minimum Vertex Cover to within any factor smaller than* $10\sqrt{5} - 21 = 1.3606\ldots$.

Theorem 4.35. *It is* $\mathcal{NP}$*-hard to determine the neighborhood number in co-bipartite graphs.*

Proof. To prove that the problem is $\mathcal{NP}$-hard, we shall use Theorem 4.34, and show that a polynomial-time reduction from a $\frac{4}{3}$-approximation of the Minimum Vertex Cover problem can be easily obtained. For that purpose, given a graph H, we will show to build a co-bipartite graph G such that $\beta(H) \leqslant \rho_n(G) \leqslant \beta(H)+1$. Namely, given any graph $H = (V, E)$, let $G = (X, Y, F)$ where $X = V$, $Y = E$ and $F = \{ve \in V \times E \colon v \text{ is incident to } e \text{ in } H\}$; that is, every vertex in X is joined to the edges in Y to which it is incident in H.

Given a set vertex cover $C \subseteq V$ of H, then C together with any element of Y is clearly neighborhood set of G. In fact, all the edges of the cliques X and Y will clearly be covered by any vertex of X and the vertex of Y, respectively. Moreover all edges of F will be covered because if $ve \in F$, then $e = vw$ (in H) for some $w \in V$. Hence, since C was a vertex cover of H, v or w must be in C and both cover the edge ve in G (because v, w, e is a triangle in G). Thus $\rho_n(G) \leqslant \beta(H)+1$.

To check the remaining inequality, let $S \subseteq X \cup Y$ be a neighborhood set of G with minimum cardinality. If e is any element in $S \cap Y$, then e is covering in G only two edges of F, namely the ve and we, where $vw = e$ (in H). Thus if we replace e by v or w in S, this set that arises still covers all the the edges of F. If we apply this procedure successively for all vertices in $S \cup Y$, we will obtain at the end a vertex set of $S' \subseteq X$ that is a neighborhood-covering set of F and has size less than or equal to $\rho_n(G)$. It turns out that $S' \subseteq V$ will be a vertex cover of H, because for any edge $e \in E$, where $e = vw$ (in H), v or w will be in S' for these are the only vertices in X that cover $ve \in F$. As S' is a a vertex cover of H whose size is less than or equal to $\rho_n(G)$, $\beta(H) \leqslant \rho_n(G)$.

Now that we have proved that this co-bipartite graph G satisfies $\beta(H) \leqslant \rho_n(G) \leqslant \beta(H) + 1$, it is easy to give a polynomial-time reduction to the problem of approximating $\beta(H)$ within a factor of $\frac{4}{3}$. Given a graph H, we can in polynomial (linear) time decide whether it has a vertex cover of size 1 or 2 and, if so, we transform H into an arbitrary co-bipartite graph whose corresponding maximum neighborhood set has size 1 or 2, respectively. If $\beta(H) \geqslant 3$ we construct in polynomial time G as described above. As proven before $\beta(H) \leqslant \rho_n(G) \leqslant \beta(H) + 1$, which as $\beta(H) \geqslant 3$ means that $1 \leqslant \frac{\rho_n(G)}{\beta(H)} \leqslant 1 + \frac{1}{3}$. This proves the reduction from the problem of approximating the Minimum Vertex Cover problem less than $10\sqrt{5} - 21 = 1.3606\ldots$, as desired. $\square$

Chapter 5

Conclusion and Final Remarks

Throughout this thesis we work on neighborhood-perfect graphs, a variation of perfect graphs. We study characterizations by forbidden induced subgraphs, restricted to superclasses of cographs and classes related to the class of hereditary clique-Helly graphs. Moreover we studied the recognition problem of neighborhood-perfect graphs and the problem of finding the parameters involved in the definition of neighborhood-perfectness, both restricted to these same graph classes.

In Chapter 3 we consider the characterizations mentioned above. The main results of this chapter are summarized in Table 5.1.

In Section 3.2 we studied how the join operation modifies the two parameters used in the definition of neighborhood-perfectness. In Theorem 3.9 and Theorem 3.12 we gave two equalities that determine how these parameters change with the join operation. Using these results, we found in Theorem 3.20 exactly all graphs that are minimally non-neighborhood-perfect with disconnected complement. This in turn allowed us to characterize by forbidden induced subgraphs the neighborhood-perfect graphs restricted to the classes of P_4-tidy graphs and tree-cographs. It seems interesting to note that the techniques used to find this characterizations could be extended to prove forbidden induced subgraph characterizations of neighborhood-perfectness in graph classes with known modular decomposition tree, provided that the minimally non-neighborhood-perfect graphs could be easily identified in the graphs represented by the N-nodes of this tree.

In Section 3.5 we proved several forbidden induced subgraph characterizations of neighborhood-perfect graphs restricted to classes that are either contained in the class of HCH graphs or strongly related to it. In most of these proofs, we use a result proved by Lehel and explicitly restated here in Theorem 3.27. This theorem shows how deeply related the classes of clique-perfect graphs and neighborhood-perfect graph are. Thus, using this relation and known results on clique-perfectness, we could prove several partial characterizations by forbidden induced subgraphs of neighborhood-perfect graphs. It is noteworthy that the relationship observed between neighborhood-perfect graphs and clique-perfect graphs could eventually be used as well to find characterizations of clique-perfectness, using results of neighborhood-perfectness. Furthermore, there are several graph classes for which clique-perfectness has been studied and in which we believe results on neighborhood-perfectness could be achieved by using similar techniques as those used in this section (e.g., complement of line graphs).

In Chapter 4, we studied the algorithmic problems associated with neighborhood-perfectness. In particular, we studied the problem of recognizing neighborhood-perfect graphs in Section 4.2 (the main results are summarized in Table 5.2) and, in the final two sections, the problems of finding $\alpha_n(G)$ and $\rho_n(G)$, for graphs G belonging to the classes of P_4-tidy graphs, tree-cographs and complement of bipartite graphs (results are summarized in Table 5.3). For P_4-tidy graphs and tree-cographs G, we gave as well linear-time algorithms that find the sets that achieve the optimum $\alpha_n(G)$ and $\rho_n(G)$. Similarly to the previous chapter, most

Graph Class	Minimal forbidden induced subgraphs for neighborhood-perfectness	Reference
P_4-tidy graphs	0-pyramid, 3-pyramid, C_5	Theorem 3.22
tree-cographs	3-pyramid, $P_6 \vee 3K_1$	Theorem 3.26
HCA graphs	0-pyramid, $\overline{C_7}$, odd holes, vikings, 2-vikings, S_k, T_k	Theorem 3.31
gem-free CA graphs	odd holes, 3-pyramid	Corollary 3.33
diamond-free graphs	odd generalized suns	Corollary 3.35
HCH claw-free graphs	odd holes, $\overline{C_7}$	Corollary 3.37

Table 5.1: Minimal forbidden induced subgraph characterizations for neighborhood-perfect graphs within the graph classes studied in Chapter 3.

Graph Class	Time complexity of recognition of neighborhood-perfect graphs	Reference
P_4-tidy graphs	linear-time	Theorem 4.12
tree-cographs	linear-time	Theorem 4.14
paw-free graphs	linear-time	Corollary 4.15
diamond-free graphs	polynomial-time	Corollary 4.16
chordal graphs	polynomial-time	Corollary 4.17
claw-free HCH graphs	polynomial-time	Corollary 4.18
HCA graphs	polynomial-time	Corollary 4.19

Table 5.2: Time complexity of the recognition problem of neighborhood-perfectness restricted to several graph classes, results appear in Chapter 4.

Graph Class	Time complexity of finding α_n and ρ_n	Reference
P_4-tidy graphs	linear-time	Theorem 4.24
tree-cographs	linear-time	Theorem 4.32
complement of bipartite graphs	$\mathcal{NP}$-hard	Theorems 4.33 and 4.35

Table 5.3: Time complexity of the problem of finding α_n and ρ_n restricted to two superclasses of cographs and the class of complement of bipartite graphs, results appear in Chapter 4.

of the algorithmic results on P_4-tidy graphs and tree-cographs are based on the identities proved in Theorems 3.9 and 3.12 and their corollaries. Using both of these theorems we were able to propose algorithms that work recursively on the modular decomposition trees of these graphs. We believe that this technique could be used to solve the recognition problem of neighborhood-perfect graphs in other classes that have a well understood modular decomposition tree. Moreover, we gave a general algorithm, that given any graph G and its modular decomposition tree $T(G)$, reduces the problem of finding optimal neighborhood sets and neighborhood-independent sets in G, to the problem of determining these sets, as well as a maximum 2-independent set and a minimum dominating set in the N-nodes of $T(G)$.

It is worth noting that a different approach for obtaining linear-time algorithms for P_4-tidy graphs (and, more generally, in graph classes having bounded clique-width) was introduced by Courcelle et al in [39]. This approach allows for linear-time solutions of recognition and optimization problems that are expressible in a certain monadic second-order logic. Given the characterizations proven in Theorems 3.22 and 3.26, it is easy to see that the recognition problem of neighborhood-perfectness in P_4-tidy graphs and tree-cographs can be expressed in this monadic second-order logic. As the class of tree-cographs also has bounded clique-width, Courcelle's metatheorem would imply the existence of a linear-time algorithm for the recognition problem of neighborhood-perfectness when the input graph is restricted to both tree-cographs and P_4-tidy graphs. Moreover, the problem of finding a minimum neighborhood-covering set can be seen to fall as well in the scope of their approach, implying a linear-time algorithm to solve this problem in any P_4-tidy graph or tree-cograph. Nevertheless, although Courcelle's metatheorem is of great theoretical importance, the algorithm obtained by it is far away from being practical; although it begins by building a modular decomposition tree of the input graph, the rest of the algorithm may have enormous hidden constants in the linear-time complexity (even if the input graph has small clique-width) [38]. This combinatorial explosion of the constants seems to be a consequence of the generality of the metatheorem, given that it requires only a monadic second-order formula and an input graph to solve the problem. This seems unavoidable if one wishes to obtain results for general monadic second-order formulas [47]. Therefore, it is clearly of interest to find more practical algorithms, that can work by only performing a simple transversal of the modular decomposition trees of the input graph.

Finally, in Section 4.4, we proved that determining $\alpha_n(G)$ and $\rho_n(G)$ is $\mathcal{NP}$-hard even if the graph G is the complement of a bipartite graph. The reduction used in this proof was inspired by similar reductions from [56] used to prove $\mathcal{NP}$-completeness of determining α_c and τ_c. We believe that by further exploring the similarities between the parameters of clique-perfectness and neighborhood-perfectness, new results that would help to understand the algorithmic complexity of determining this parameters could be found. Moreover it is worth studying if the results found for $\alpha_n(G)$ and $\rho_n(G)$ could be extended to the generalized parameters $\alpha_n(G,k)$ and $\rho_n(G,k)$. In particular we believe that the algorithmic results for P_4-tidy graphs and tree-cographs could be extended to these parameters in a fairly straightforward way.

Index

Bibliography

[1] T. Andreae, M. Schughart, and Z. Tuza. Clique-transversal sets of line graphs and complements of line graphs. *Discrete Math.*, 88(1):11–20, 1991. 13

[2] S. Baumann. A linear algorithm for the homogeneous decomposition of graphs. Report TUM M9615, Fakultät für Mathematik, Technische Universität München, Munich, Germany, 1996. 24

[3] M. Bellare, O. Goldreich, and M. Sudan. Free bits, PCPs, and nonapproximability—towards tight results. *SIAM J. Comput.*, 27(3):804–915 (electronic), 1998. 30

[4] C. Berge. Färbung von graphen, deren sämtliche bzw. deren ungerade kreise starr sind. *Wiss. Z. Martin-Luther-Univ. Halle-Wittenberg Math.-Natur. Reihe*, 10(114):88, 1961. 1, 16

[5] C. Berge. Balanced matrices. *Math. Programming*, 2(1):19–31, 1972. 21, 52

[6] C. Berge. Motivations and history of some of my conjectures. *Discrete Math.*, 165/166:61–70, 1997. 2

[7] C. Berge and V. Chvátal. Introduction. In *Topics on Perfect Graphs*, volume 88 of *North-Holland Mathematics Studies*, pages vii–xiv. Noth-Holland, Amsterdam, 1984. 21

[8] C. Berge and M. Las Vergnas. Sur un théorème du type König pour hypergraphes. *Ann. New York Acad. Sci.*, 175:32–40, 1970. 21, 52

[9] A. Bondy, G. Durán, M. C. Lin, and J. L. Szwarcfiter. Self-clique graphs and matrix permutations. *J. Graph Theory*, 44(3):178–192, 2003. 19

[10] F. Bonomo. *On subclasses and variations of perfect graphs*. Doctoral thesis, Departamento de Computación, FCEyN, Universidad de Buenos Aires, Buenos Aires, Argentina, 2005. 51, 52

[11] F. Bonomo, 2014. Personal communication. 52

[12] F. Bonomo, M. Chudnovsky, and G. Durán. Partial characterizations of clique-perfect graphs. I. Subclasses of claw-free graphs. *Discrete Appl. Math.*, 156(7):1058–1082, 2008. 19, 51, 71

[13] F. Bonomo, M. Chudnovsky, and G. Durán. Partial characterizations of clique-perfect graphs. II. Diamond-free and Helly circular-arc graphs. *Discrete Math.*, 309(11):3485–3499, 2009. 19, 20, 22, 48, 49, 51, 71

[14] F. Bonomo, G. Durán, L. N. Grippo, and M. D. Safe. Partial characterizations of circle graphs. *Discrete Appl. Math.*, 159(16):1699–1706, 2011. 27

[15] F. Bonomo, G. Durán, L. N. Grippo, and M. D. Safe. Probe interval graphs and probe unit interval graphs on superclasses of cographs. *Discrete Math. Theor. Comput. Sci.*, 15(2):177–194, 2013. 27

[16] F. Bonomo, G. Durán, M. C. Lin, and J. L. Szwarcfiter. On balanced graphs. *Math. Program.*, 105(2-3, Ser. B):233–250, 2006. 21

[17] F. Bonomo, G. Durán, M. D. Safe, and A. K. Wagler. Clique-perfectness of complements of line graphs. In *LAGOS'11—VI Latin-American Algorithms, Graphs and Optimization Symposium*, volume 37 of *Electron. Notes Discrete Math.*, pages 327–332. Elsevier Sci. B. V., Amsterdam, 2011. 19

[18] F. Bonomo, G. Durán, M. D. Safe, and A. K. Wagler. On minimal forbidden subgraph characterizations of balanced graphs. *Discrete Appl. Math.*, 161(13-14):1925–1942, 2013. 21

[19] F. Bonomo, G. Durán, M. D. Safe, and A. K. Wagler. Balancedness of subclasses of circular-arc graphs. *Discrete Math. Theor. Comput. Sci.*, 16(3):1–22, 2014. 21, 50

[20] F. Bonomo, G. Durán, M. D. Safe, and A. K. Wagler. Clique-perfectness and balancedness of some graph classes. *Int. J. Comput. Math.*, 91(10):2118–2141, 2014. 19, 21, 52, 53, 70

[21] F. Bonomo, G. Durán, F. Soulignac, and G. Sueiro. Partial characterizations of clique-perfect and coordinated graphs: superclasses of triangle-free graphs. *Discrete Appl. Math.*, 157(17):3511–3518, 2009. 19

[22] F. Bonomo, O. Schaudt, M. Stein, and M. Valencia-Pabon. b-coloring is NP-hard on co-bipartite graphs and polytime solvable on tree-cographs. In P. Fouilhoux, L. E. N. Gouveia, A. R. Mahjoub, and V. T. Paschos, editors, *Combinatorial Optimization - Third International Symposium, ISCO 2014, Lisbon, Portugal, March 5-7, 2014, Revised Selected Papers*, volume 8596 of *Lecture Notes in Computer Science*, pages 100–111. Springer, 2014. 27

[23] A. Brandstädt, V. D. Chepoi, and F. F. Dragan. Clique r-domination and clique r-packing problems on dually chordal graphs. *SIAM J. Discrete Math.*, 10(1):109–127, 1997. 3, 13, 57

[24] H. Buer and R. H. Möhring. A fast algorithm for the decomposition of graphs and posets. *Math. Oper. Res.*, 8(2):170–184, 1983. 23

[25] R. W. Bulterman, F. W. van der Sommen, G. Zwaan, T. Verhoeff, A. J. M. van Gasteren, and W. H. J. Feijen. On computing a longest path in a tree. *Inform. Process. Lett.*, 81(2):93–96, 2002. 68

[26] K. Cameron. Induced matchings. *Discrete Appl. Math.*, 24(1-3):97–102, 1989. 10

[27] G. J. Chang, M. Farber, and Z. Tuza. Algorithmic aspects of neighborhood numbers. *SIAM J. Discrete Math.*, 6(1):24–29, 1993. 3, 12, 13, 14, 56, 83

[28] M.-S. Chang, Y.-H. Chen, G. J. Chang, and J.-H. Yan. Algorithmic aspects of the generalized clique-transversal problem on chordal graphs. *Discrete Appl. Math.*, 66(3):189–203, 1996. 13, 15

[29] M. Chellali, O. Favaron, A. Hansberg, and L. Volkmann. k-domination and k-independence in graphs: a survey. *Graphs Combin.*, 28(1):1–55, 2012. 14

[30] M. Chellali and T. W. Haynes. A note on the total domination number of a tree. *J. Combin. Math. Combin. Comput.*, 58:189–193, 2006. 79, 83

[31] M. Chudnovsky, G. Cornuéjols, X. Liu, P. Seymour, and K. Vušković. Recognizing Berge graphs. *Combinatorica*, 25(2):143–186, 2005. 18

[32] M. Chudnovsky, N. Robertson, P. Seymour, and R. Thomas. The strong perfect graph theorem. *Ann. of Math. (2)*, 164(1):51–229, 2006. 2, 18

[33] V. Chvátal. On certain polytopes associated with graphs. *J. Combinatorial Theory Ser. B*, 18:138–154, 1975. 18

[34] S. A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the Third Annual ACM Symposium on Theory of Computing*, STOC '71, pages 151–158, New York, NY, USA, 1971. ACM. 28

[35] D. G. Corneil, H. Lerchs, and L. S. Burlingham. Complement reducible graphs. *Discrete Appl. Math.*, 3(3):163–174, 1981. 25, 27

[36] D. G. Corneil, Y. Perl, and L. Stewart. Cographs: recognition, applications and algorithms. In *Proceedings of the fifteenth Southeastern conference on combinatorics, graph theory and computing (Baton Rouge, La., 1984)*, volume 43, pages 249–258, 1984. 25

[37] D. G. Corneil, Y. Perl, and L. K. Stewart. A linear recognition algorithm for cographs. *SIAM J. Comput.*, 14(4):926–934, 1985. 25

[38] B. Courcelle. A multivariate interlace polynomial and its computation for graphs of bounded clique-width. *Electron. J. Combin.*, 15(1):Research Paper 69, 36, 2008. 90

[39] B. Courcelle, J. A. Makowsky, and U. Rotics. Linear time solvable optimization problems on graphs of bounded clique-width. *Theory Comput. Syst.*, 33(2):125–150, 2000. 90

[40] A. Cournier and M. Habib. A new linear algorithm for modular decomposition. In *Trees in algebra and programming—CAAP '94 (Edinburgh, 1994)*, volume 787 of *Lecture Notes in Comput. Sci.*, pages 68–84. Springer, Berlin, 1994. 24

[41] E. Dahlhaus, J. Gustedt, and R. M. McConnell. Efficient and practical algorithms for sequential modular decomposition. *J. Algorithms*, 41(2):360–387, 2001. 24

[42] I. Dinur and S. Safra. On the hardness of approximating minimum vertex cover. *Ann. of Math. (2)*, 162(1):439–485, 2005. 30, 84, 85

[43] G. Durán, M. C. Lin, and J. L. Szwarcfiter. On clique-transversals and clique-independent sets. *Ann. Oper. Res.*, 116:71–77, 2002. 13, 19

[44] P. Erdős, T. Gallai, and Z. Tuza. Covering the cliques of a graph with vertices. *Discrete Math.*, 108(1-3):279–289, 1992. 13

[45] F. Escalante. Über iterierte Clique-Graphen. *Abh. Math. Sem. Univ. Hamburg*, 39:59–68, 1973. 19

[46] J.-L. Fouquet and V. Giakoumakis. On semi-P_4-sparse graphs. *Discrete Math.*, 165/166:277–300, 1997. 24

[47] M. Frick and M. Grohe. The complexity of first-order and monadic second-order logic revisited. *Ann. Pure Appl. Logic*, 130(1-3):3–31, 2004. 91

[48] G. Fricke and R. Laskar. Strong matchings on trees. In *Proceedings of the Twenty-third Southeastern International Conference on Combinatorics, Graph*

Theory, and Computing (Boca Raton, FL, 1992), volume 89, pages 239–243, 1992. 68, 70

[49] D. R. Fulkerson. On the perfect graph theorem. In *Mathematical progamming (Proc. Advanced Sem., Univ. Wisconsin, Madison, Wis., 1972)*, pages 69–76. Math. Res. Center Publ., No. 30. Academic Press, New York, 1973. 2

[50] T. Gallai. Transitiv orientierbare Graphen. *Acta Math. Acad. Sci. Hungar*, 18:25–66, 1967. 23

[51] M. R. Garey and D. S. Johnson. *Computers and intractability*. W. H. Freeman and Co., San Francisco, Calif., 1979. 14, 30

[52] F. Gavril. Algorithms on circular-arc graphs. *Networks*, 4:357–369, 1974. 22

[53] V. Giakoumakis, F. Roussel, and H. Thuillier. On P_4-tidy graphs. *Discrete Math. Theor. Comput. Sci.*, 1(1):17–41 (electronic), 1997. 26

[54] M. C. Golumbic. *Algorithmic graph theory and perfect graphs*, volume 57 of *Annals of Discrete Mathematics*. Elsevier Science B.V., Amsterdam, second edition, 2004. 1

[55] M. C. Golumbic and M. Lewenstein. New results on induced matchings. *Discrete Appl. Math.*, 101(1-3):157–165, 2000. 68, 70

[56] V. Guruswami and C. P. Rangan. Algorithmic aspects of clique-transversal and clique-independent sets. *Discrete Appl. Math.*, 100(3):183–202, 2000. 3, 13, 19, 56, 57, 91

[57] A. Gyárfás, D. Kratsch, J. Lehel, and F. Maffray. Minimal non-neighborhood-perfect graphs. *J. Graph Theory*, 21(1):55–66, 1996. 3, 4, 32, 33, 37, 56

[58] M. Habib and C. Paul. A survey of the algorithmic aspects of modular decomposition. *Comput. Sci. Rev.*, 4(1):41–59, 2010. 24

[59] R. C. Hamelink. A partial characterization of clique graphs. *J. Combinatorial Theory*, 5:192–197, 1968. 19

[60] J. Håstad. Clique is hard to approximate within $n^{1-\epsilon}$. *Acta Math.*, 182(1):105–142, 1999. 30

[61] J. Håstad. Some optimal inapproximability results. *J. ACM*, 48(4):798–859, 2001. 30

[62] S. T. Hedetniemi and R. C. Laskar. Bibliography on domination in graphs and some basic definitions of domination parameters. *Discrete Math.*, 86(1-3):257–277, 1990. 14

[63] M. A. Henning and A. Yeo. *Total domination in graphs*. Springer Monographs in Mathematics. Springer, New York, 2013. 83

[64] C. T. Hoang. *Perfect graphs*. Ph.D. thesis, School of Computer Science, McGill University, Montreal, Canada, 1985. 25

[65] S.-F. Hwang and G. J. Chang. k-neighborhood-covering and -independence problems for chordal graphs. *SIAM J. Discrete Math.*, 11(4):633–643, 1998. 12, 13, 16, 57

[66] B. L. Joeris, M. C. Lin, R. M. McConnell, J. P. Spinrad, and J. L. Szwarcfiter. Linear-time recognition of Helly circular-arc models and graphs. *Algorithmica*, 59(2):215–239, 2011. 22

[67] R. M. Karp. Reducibility among combinatorial problems. In *Complexity of computer computations (Proc. Sympos., IBM Thomas J. Watson Res. Center, Yorktown Heights, N.Y., 1972)*, pages 85–103. Plenum, New York, 1972. 12, 15, 28

[68] 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. 83

[69] J. Lehel. Neighbourhood-perfect line graphs. *Graphs Combin.*, 10(4):353–361, 1994. 3, 4, 5, 19, 20, 33, 46, 47, 52

[70] J. Lehel and Z. Tuza. Neighborhood perfect graphs. *Discrete Math.*, 61(1):93–101, 1986. 2, 3, 4, 12, 13, 18, 32, 40, 45, 52, 56, 81

[71] L. A. Levin. Universal enumeration problems. *Problemy Peredači Informacii*, 9(3):115–116, 1973. 28

[72] M. C. Lin and J. L. Szwarcfiter. Characterizations and linear time recognition of Helly circular-arc graphs. In *Computing and combinatorics*, volume 4112 of *Lecture Notes in Comput. Sci.*, pages 73–82. Springer, Berlin, 2006. 22, 50

[73] M. C. Lin and J. L. Szwarcfiter. Faster recognition of clique-Helly and hereditary clique-Helly graphs. *Inform. Process. Lett.*, 103(1):40–43, 2007. 19

[74] L. Lovász. A characterization of perfect graphs. *J. Combinatorial Theory Ser. B*, 13:95–98, 1972. 1, 17

[75] L. Lovász. Normal hypergraphs and the perfect graph conjecture. *Discrete Math.*, 2(3):253–267, 1972. 1

[76] C. Lund and M. Yannakakis. On the hardness of approximating minimization problems. *J. Assoc. Comput. Mach.*, 41(5):960–981, 1994. 30

[77] A. Lyons. Acyclic and star colorings of cographs. *Discrete Appl. Math.*, 159(16):1842–1850, 2011. 27

[78] R. M. McConnell. Linear-time recognition of circular-arc graphs. *Algorithmica*, 37(2):93–147, 2003. 22

[79] R. M. McConnell and J. P. Spinrad. Modular decomposition and transitive orientation. *Discrete Math.*, 201(1-3):189–241, 1999. 24

[80] S. Mitchell, S. Hedetniemi, and S. Goodman. Some linear algorithms on trees. In *Proceedings of the Sixth Southeastern Conference on Combinatorics, Graph Theory, and Computing (Florida Atlantic Univ., Boca Raton, Fla., 1975)*, pages 467–483. Congressus Numerantium, No. XIV. Utilitas Math., Winnipeg, Man., 1975. 68, 82

[81] S. L. Mitchell, E. J. Cockayne, and S. T. Hedetniemi. Linear algorithms on recursive representations of trees. *J. Comput. System Sci.*, 18(1):76–85, 1979. 68, 82

[82] S. D. Nikolopoulos, L. Palios, and C. Papadopoulos. Counting spanning trees using modular decomposition. *Theoret. Comput. Sci.*, 526:41–57, 2014. 27

[83] R. Paige and R. E. Tarjan. Three partition refinement algorithms. *SIAM J. Comput.*, 16(6):973–989, 1987. 57

[84] C. H. Papadimitriou and K. Steiglitz. *Combinatorial optimization: algorithms and complexity*. Dover Publications, Inc., Mineola, NY, 1998. 30

[85] C. H. Papadimitriou and M. Yannakakis. Optimization, approximation, and complexity classes. *J. Comput. System Sci.*, 43(3):425–440, 1991. 29

[86] E. Prisner. Hereditary clique-Helly graphs. *J. Combin. Math. Combin. Comput.*, 14:216–220, 1993. 20, 47, 49

[87] F. S. Roberts and J. H. Spencer. A characterization of clique graphs. *J. Combinatorial Theory Ser. B*, 10:102–108, 1971. 19

[88] M. D. Safe. *On structural characterizations of graph classes related to perfect graphs and the Kőnig property*. Doctoral thesis, Departamento de Computación, FCEyN, Universidad de Buenos Aires, Buenos Aires, Argentina, 2011. 21

[89] E. Sampathkumar and P. S. Neeralagi. The neighbourhood number of a graph. *Indian J. Pure Appl. Math.*, 16(2):126–132, 1985. 2, 12, 45

[90] C. Savage. Depth-first search and the vertex cover problem. *Inform. Process. Lett.*, 14(5):233–235, 1982. 64, 68, 82

[91] D. Seinsche. On a property of the class of n-colorable graphs. *J. Combinatorial Theory Ser. B*, 16:191–193, 1974. 25

[92] P. Seymour. How the proof of the strong perfect graph conjecture was found. *Gaz. Math.*, 109:69–83, 2006. 2

[93] J. P. Spinrad. Doubly lexical ordering of dense 0-1 matrices. *Inform. Process. Lett.*, 45(5):229–235, 1993. 57

[94] M. Tedder, D. Corneil, M. Habib, and C. Paul. Simpler linear-time modular decomposition via recursive factorizing permutations. In *Automata, languages and programming. Part I*, volume 5125 of *Lecture Notes in Comput. Sci.*, pages 634–645. Springer, Berlin, 2008. 24

[95] G. Tinhofer. Strong tree-cographs are Birkhoff graphs. *Discrete Appl. Math.*, 22(3):275–288, 1988/89. 27

[96] S. Tsukiyama, M. Ide, H. Ariyoshi, and I. Shirakawa. A new algorithm for generating all the maximal independent sets. *SIAM J. Comput.*, 6(3):505–517, 1977. 20, 21

[97] A. Tucker. Matrix characterizations of circular-arc graphs. *Pacific J. Math.*, 39:535–545, 1971. 22

[98] A. Tucker. Perfect graphs and an application to optimizing municipal services. *SIAM Rev.*, 15:585–590, 1973. 1

[99] A. Tucker. Structure theorems for some circular-arc graphs. *Discrete Math.*, 7:167–195, 1974. 22

[100] A. Tucker. Coloring a family of circular arcs. *SIAM J. Appl. Math.*, 29(3):493–502, 1975. 22

[101] A. Tucker. An efficient test for circular-arc graphs. *SIAM J. Comput.*, 9(1):1–24, 1980. 22

[102] Z. Tuza. Covering all cliques of a graph. *Discrete Math.*, 86(1-3):117–126, 1990. 13

[103] V. V. Vazirani. *Approximation algorithms*. Springer-Verlag, Berlin, 2001. 30

[104] D. B. West. *Introduction to graph theory*. Prentice Hall, Inc., Upper Saddle River, NJ, 1996. 7, 11

[105] G. Zambelli. A polynomial recognition algorithm for balanced matrices. *J. Combin. Theory Ser. B*, 95(1):49–67, 2005. 21

[106] M. Zito. Linear time maximum induced matching algorithm for trees. *Nordic J. Comput.*, 7(1):58–63, 2000. 68, 70