Ann Oper Res (2020) 286:55–86
https://doi.org/10.1007/s10479-017-2712-z

S.I.:CLAIO 2016

Neighborhood covering and independence on P_4-tidy graphs and tree-cographs

Guillermo Durán[1,2,3] · **Martín Safe**[4] · **Xavier Warnes**[1,5]

Published online: 28 November 2017

Abstract Given a simple graph G, a set $C \subseteq V(G)$ is a neighborhood cover set if every edge and vertex of G belongs to some $G[v]$ with $v \in C$, where $G[v]$ denotes the subgraph of G induced by the closed neighborhood of the vertex v. Two elements of $E(G) \cup V(G)$ are neighborhood-independent if there is no vertex $v \in V(G)$ such that both elements are in $G[v]$. A set $S \subseteq V(G) \cup E(G)$ is neighborhood-independent if every pair of elements of S is neighborhood-independent. Let $\rho_n(G)$ be the size of a minimum neighborhood cover set and $\alpha_n(G)$ of a maximum neighborhood-independent set. Lehel and Tuza defined neighborhood-perfect graphs G as those where the equality $\rho_n(G') = \alpha_n(G')$ holds for every induced subgraph G' of G. In this work we prove forbidden induced subgraph characterizations of the class of neighborhood-perfect graphs, restricted to two superclasses of cographs: P_4-tidy graphs and tree-cographs. We give as well linear-time algorithms for solving the recognition problem of neighborhood-perfect graphs and the problem of finding a minimum neighborhood cover set and a maximum neighborhood-independent set in these same classes. Finally we prove that although for complements of trees finding these optimal sets can be achieved in linear-time, for complements of bipartite graphs it is NP-hard.

✉ Xavier Warnes
xwarnes@stanford.edu

Guillermo Durán
gduran@dm.uba.ar

Martín Safe
msafe@uns.edu.ar

1 Instituto de Cálculo and Departamento de Matemática, Facultad de Ciencias Exactas y Naturales, Universidad de Buenos Aires, Buenos Aires, Argentina

2 Departamento de Ingeniería Industrial, Facultad de Ciencias Físicas y Matemáticas, Universidad de Chile, Santiago, Chile

3 CONICET, Buenos Aires, Argentina

4 Departamento de Matemática, Universidad Nacional del Sur, Bahía Blanca, Argentina

5 Graduate School of Business, Stanford University, Stanford, CA, USA

Keywords Forbidden induced subgraphs · Neighborhood-perfect graphs · P_4-tidy graphs · Tree-cographs · Recognition algorithms · Co-bipartite graphs

Mathematics Subject Classification 05C17 · 05C69 · 05C75

1 Introduction

A graph is *perfect* if, for every induced subgraph, the maximum size of a clique equals the minimum number of colors needed to color its vertices such that no two adjacent vertices have the same color. 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. This characterization was proved by Chudnovsky, Robertson, Seymour and Thomas in 2002 (Chudnovsky et al. 2006), settling affirmatively a conjecture posed more than 40 years before by Berge (1961). The minimal forbidden induced subgraphs of perfect graphs are the chordless cycles of odd length having at least 5 vertices, called *odd holes* C_{2k+1}, and their complements, the *odd antiholes* $\overline{C}_{2k+1}$.

During the nearly half a century in which this characterization remained a conjecture, many graph classes were defined analogously to perfect graphs by the equality of two parameters [e.g. clique perfect graphs by Guruswami and Rangan (2000), coordinated graphs by Bonomo et al. (2007), neighborhood-perfect graphs by Lehel and Tuza (1986)].

Neighborhood-perfect graphs were defined by Lehel and Tuza (1986), by the equality of two parameters for all induced subgraphs. Given a simple graph G, a set $C \subseteq V(G)$ is a *neighborhood-covering set* (or *neighborhood set*) if each edge and each vertex of G belongs to some $G[v]$ with $v \in C$, where $G[v]$ denotes the subgraph of G induced by the closed neighborhood of the vertex v. Two elements of $E(G) \cup V(G)$ are *neighborhood-independent* if there is no vertex $v \in V(G)$ such that both elements are in $G[v]$. 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. Let $\rho_{\mathrm{n}}(G)$ be the size of a minimum neighborhood-covering set and $\alpha_{\mathrm{n}}(G)$ of a maximum neighborhood-independent set. Clearly, $\rho_{\mathrm{n}}(G) \geq \alpha_{\mathrm{n}}(G)$ for every graph G. When $\rho_{\mathrm{n}}(G') = \alpha_{\mathrm{n}}(G')$ for every induced subgraph G' of G, G is called a *neighborhood-perfect* graph. Since odd holes and odd antiholes are not neighborhood-perfect (Lehel and Tuza 1986), the Strong Perfect Graph Theorem implies that all neighborhood-perfect graphs are also perfect.

Neighborhood-perfect graphs have been characterized by forbidden induced subgraphs, when restricted to the classes of chordal graphs (Lehel and Tuza 1986), line graphs (Lehel 1994) and cographs (Gyárfás et al. 1996). The characterizations presented here are an extension of this last result. Furthermore, Lehel and Tuza proved that finding $\alpha_{\mathrm{n}}(G)$ and $\rho_{\mathrm{n}}(G)$ can be done in polynomial time if G is a chordal neighborhood-perfect graph. If G is strongly chordal, interval or a cograph (i.e., P_4-free), then linear-time algorithms that find the above mentioned parameters have been given (Brandstädt et al. 1997; Gyárfás et al. 1996; Lehel and Tuza 1986). On the other hand it was proven that the problems of finding these parameters are NP-complete over a class of split graphs with degree constraints (Chang et al. 1993). Although it follows from previous works (e.g. Gyárfás et al. 1996; Lehel 1994; Lehel and Tuza 1986) that deciding whether a graph is neighborhood-perfect can be accomplished in polynomial-time if the input graph belongs to several different graph classes, the computational complexity of recognizing neighborhood-perfect graphs in general is unknown.

The work is organized as follows. In Sect. 2, we give some preliminary definitions and results, including an introduction to modular decomposition and the structure of the classes of P_4-tidy graphs and tree-cographs. In Sect. 3, we give formulas for α_n and ρ_n for the join of two or more graphs and determine all the minimally non-neighborhood-perfect graphs whose complement is disconnected. In Sect. 4, we prove our structural results, which consist in minimal forbidden induced subgraph characterizations of the class of neighborhood-perfect graphs when restricted to the classes of P_4-tidy graphs and tree-cographs, respectively. In Sect. 5, we give our algorithmic results, which consist in linear-time recognition algorithms for neighborhood-perfectness of P_4-tidy graphs and tree-cographs, linear-time algorithms for computing α_n and ρ_n for any given P_4-tidy graph or tree-cograph, and a proof that the problems of computing α_n and ρ_n become NP-hard for complements of bipartite graphs.

2 Preliminaries

Before we formulate the results, a few definitions that will be used later on are required. For all undefined terminology we refer to West (2001). All graphs in this work are finite, undirected, and have no loops or multiple edges. Let G be a graph. We shall denote by $V(G)$ its vertex set and $E(G)$ its edge set, by m_G the size of $E(G)$ and by n_G the size of $V(G)$ (omitting the subscript G when it is clear by context). The complement of G shall be denoted by $\overline{G}$, the neighborhood of a vertex v by $N_G(v)$, and the closed neighborhood $N_G(v) \cup \{v\}$ by $N_G[v]$. We denote by $G[W]$ the subgraph of G induced by $W \subseteq V(G)$. A vertex of G is said to be *pendant* if it is adjacent to exactly one vertex of G, *universal* if it is adjacent to all other vertices, *isolated* if it is not adjacent to any other vertex, and *simplicial* if its neighborhood is a clique. If H is a graph, then G is H-free if G contains no induced subgraphs isomorphic to H. If $\mathscr{H}$ is a collection of graphs, then G is $\mathscr{H}$-free if G is H-free for all $H \in \mathscr{H}$. The chordless path of k vertices is denoted P_k, and the chordless cycle of k vertices C_k. A cycle is odd if it has an odd number of vertices. The graph K_n is the complete graph of n vertices. If H is a graph and t is a nonnegative integer, then tH denotes the disjoint union of t copies of H. The *join* of two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ (where $V_1 \cap V_2 = \emptyset$) is the graph $G_1 \vee G_2 = (V_1 \cup V_2, E_1 \cup E_2 \cup \{uv \mid u \in V_1, v \in V_2\})$. The disjoint union of two graphs G_1 and G_2 is denoted by $G_1 + G_2$. The size of a set S is denoted by $|S|$. We shall consider a *clique* (resp. *independent set*) to be a set of pairwise adjacent (resp. nonadjacent) vertices, and a *2-independent set* to be a set of vertices such that for every pair of vertices of the set, there is no path of length 2 or less that connects them in G. We shall use *maximum* to denote of maximum size, while *maximal* shall denote inclusion-wise maximal (analogously with *minimum* and *minimal*). Given a graph G, we shall denote by $\alpha(G)$ the size of any maximum stable set of G and by $\alpha_2(G)$ the size of a maximum 2-independent set of G. A set of vertices of G is said to be a *dominating set* if every vertex in G belongs to the set or is adjacent to a vertex of the set. A *total dominating set* of G is a set of vertices such that every vertex of G is adjacent to at least one vertex of the set. We shall denote by $\gamma(G)$, and $\gamma_\mathrm{t}(G)$ the sizes of any minimum dominating set and any minimum total dominating set, respectively, of G. A *matching* of a graph is a set of edges such that every node of G is incident with at most one edge of the set. A *vertex cover* of a graph G is a set of vertices W such that every edge of G is incident to at least one vertex of W. We shall note by $\beta(G)$ the size of any minimum vertex cover of G and by $\nu(G)$ the size of any maximum matching of G.

A useful string of inequalities that will be used later is that for any graph G, $\alpha_2(G) \leq \gamma(G) \leq \alpha_\mathrm{n}(G) \leq \rho_\mathrm{n}(G)$. The first inequality comes from the fact that different vertices in a

2-independent set cannot be dominated by the same vertex of a dominating set. The second inequality is derived by observing that, given any maximum neighborhood-independent set, one can construct a dominating set of the same cardinality by simply taking a vertex from every edge and all the vertices in the neighborhood-independent set.

We say that a graph is *co-connected* if its complement is connected. The *anticomponents* of a graph are its maximal co-connected induced subgraphs (equivalently they are the complements of the components of G). This means that if $H_1 \dots H_k$ are the anticomponents of H, then $H = H_1 \vee \cdots \vee H_k$.

A graph G is said to be *chordal* if it is C_k-free, for every $k \geq 4$. A *k-sun* (or *trampoline of order k*) is a chordal graph G having $2k$ vertices $v_1, \dots, v_k, w_1, \dots, w_k$ such that, for each $i \in \{1, \dots, k\}$, $N_G(w_i) = \{v_i, v_{i+1}\}$ while $\{v_{i-1}, v_{i+1}\} \subseteq N_G(v_i)$ (where subindices are taken modulo k). Notice that each vertex v_i may be adjacent to some other vertices in $\{v_1, \dots, v_k\}$ apart from v_{i-1} and v_{i+1}; for instance, it is possible for $\{v_1, \dots, v_n\}$ to be a clique of G. A k-sun is said to be *odd* if k is odd.

As stated in Sect. 1, neighborhood-perfect graphs have been characterized by forbidden induced subgraphs restricted to the class of chordal graphs. The minimal forbidden induced subgraphs are exactly the odd suns defined above.

Theorem 1 (Lehel and Tuza 1986) *A chordal graph G is neighborhood-perfect if and only if it contains no induced odd sun.*

A graph G was defined to be minimally non-neighborhood-perfect in Gyárfás et al. (1996), if G is not neighborhood-perfect, but all proper induced subgraphs of G are. We now state for future reference a result proven in Gyárfás et al. (1996) that determines exactly which graphs G are minimally non-neighborhood-perfect and have $\alpha_{\mathrm{n}}(G) = 1$.

Theorem 2 (Gyárfás et al. 1996) *If G is a minimally non-neighborhood-perfect graph and $\alpha_{\mathrm{n}}(G) = 1$, then G is a 3-sun or $\overline{3K_2}$.*

2.1 Modular decomposition

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 every leaf one of 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 associate with each leaf the only vertex the module represents, then $M(h)$ corresponds to the set of leafs 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, and called a *P-node*, *S-node* or *N-node*, respectively. If $G[h]$ is disconnected, h is a P-node; if $\overline{G[h]}$ is disconnected, h is an S-node; and if both $G[h]$ and

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

In all of these cases, it holds that $\{M(h_1), \ldots, M(h_k)\}$ is a disjoint partition of the nodes in $M(h)$ (Buer and Möhring 1983; Gallai 1967).

Let h be a node of $T(G)$ and let $h_1, \ldots, h_k$ be its children. We shall denote by $\pi(h)$ the graph having vertex set $\{h_1, \ldots, h_k\}$ 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 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. Note that each $\pi(h)$ must be a prime graph, since all $M(h_i)$ are maximal strong modules, and if $\pi(h)$ had a nontrivial module of more than one vertex, the modules $M[h_i]$ corresponding to these vertices would form a module of $G[h]$. We shall denote by $\pi(G)$ the set $\{\pi(h) : h \textit{ is an N-node of } T(G)\}$. 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 3 (Fouquet and Giakoumakis 1997) *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 note $|V(G[h])|$ by $n(h)$, for every $h \in V(T(G))$. If h is an N-node, then we shall note $|V(\pi(h))|$ by $n_\pi(h)$ and $|E(\pi(h))|$ by $m_\pi(h)$. A fact that will be used in what follows is that since $T(G)$ has n nonadjacent leaves and each internal node has at least two children, $T(G)$ must have less than $2n$ nodes. An important property that we shall use extensively is that the sum of $n_\pi(h)$, over all N-nodes h of $T(G)$, is at most $2n$ (Baumann 1996).

In this work 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 $T(G)$ (Cournier and Habib 1994; Dahlhaus et al. 2001; McConnell and Spinrad 1999; Tedder et al. 2008), moreover in Baumann (1996) 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 Habib and Paul (2010).

2.2 Structure of P_4-tidy graphs and tree-cographs

We shall now introduce the two classes in which we will be studying neighborhood-perfectness. Both of these classes are generalizations of the class of cographs.

A graph $G = (V, E)$ is P_4*-tidy* if for every vertex set A inducing a P_4 in G there is at most one vertex $v \in V \setminus A$ such that $G[A \cup \{v\}]$ contains at least two induced P_4's. There is a structure theorem for P_4-tidy graphs that extends Seinsche's theorem in cographs (Seinsche 1974), in terms of 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 $t \geq 2$.
- The set R is allowed to be empty. If it is not, then the vertices of R can induce any graph and every vertex of R is 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 urchins are the complement of starfish and vice versa. Given G, a starfish or 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 $2K_1$ (where each vertex of the K_2 or $2K_1$ would have the same adjacencies as the vertex they substituted).

Theorem 4 (Giakoumakis et al. 1997) *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.*

Let G be a P_4-tidy graph and h be an N-node of $T(G)$, the modular decomposition tree of G. Theorem 4 implies that $\pi(h)$ must be isomorphic to C_5, P_5, $\overline{P_5}$, a prime starfish, or a prime urchin. Moreover, if $\pi(h)$ is isomorphic to a prime urchin or a prime starfish, each of the children of h in $T(G)$ is a leaf except for at most one child h_R that represents the head and/or another child representing $2K_1$ or K_2. As was seen in Giakoumakis et al. (1997), in $\mathcal{O}(n_\pi(h))$ time, it can be decided whether or not $\pi(h)$ is a starfish (resp. an urchin) and, if this is the case, find its partition.

The class of *tree-cographs* is another superclass of the class of cographs. Tree-cographs were introduced in Tinhofer (1988) by the following recursive definition:

Definition 1 1. Every tree is a tree-cograph.
2. If G is a tree-cograph, then $\overline{G}$ is a tree-cograph.
3. The disjoint union of tree-cographs is a tree-cograph.

This definition implies that if G is a tree-cograph, then either G or $\overline{G}$ is disconnected, or G is a tree or the complement of a tree. Hence, Definition 1 implies that if G is a tree-cograph and h is an N-node of $T(G)$, then $\pi(h)$ is a tree or the complement of a tree.

3 Parameters and minimal forbidden induced subgraphs

In order to effectively use the inherent structure of P_4-tidy graphs and tree-cographs, we first explore how the join operation modifies α_{n} and ρ_{n}. As a consequence of these results, we will determine ahead in this section all the minimally non-neighborhood-perfect graphs whose complement is disconnected. As a byproduct, we will also characterize the class of graphs that arises by requiring $\alpha_2 = \rho_{\mathrm{n}}$ for every induced subgraph, which we will call the class of *strongly neighborhood-perfect graph.*

Lemma 1 *If G and H are graphs, then*

$$\rho_{\mathrm{n}}(G \vee H) = \min\{\gamma(H) + 1, \gamma(G) + 1, \rho_{\mathrm{n}}(H), \rho_{\mathrm{n}}(G)\}.$$

Proof It is immediate to see that

$$\rho_{\mathrm{n}}(G \vee H) \leq \min\{\gamma(H)+1, \gamma(G)+1, \rho_{\mathrm{n}}(H), \rho_{\mathrm{n}}(G)\},$$

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 $\min\{\gamma(H)+1, \gamma(G)+1, \rho_{\mathrm{n}}(H), \rho_{\mathrm{n}}(G)\}$. Hence, as S has fewer vertices than $\rho_{\mathrm{n}}(G)$ and $\rho_{\mathrm{n}}(H)$, 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 $|S_H| \leq |S| - 1$ and $|S_G| \leq |S| - 1$. 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. □

Lemma 2 *If G and H are graphs, then*

$$\alpha_{\mathrm{n}}(G \vee H) = \min\{\alpha_2(G), \alpha_2(H)\}.$$

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 $\alpha_{\mathrm{n}}(G \vee H) \geq \min\{\alpha_2(G), \alpha_2(H)\}$, by finding a neighborhood-independent set of $G \vee H$ of that size. Without loss of generality, suppose $\alpha_2(G) \leq \alpha_2(H)$. 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, respectively, 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 $\alpha_{\mathrm{n}}(G \vee H) \geq \min\{\alpha_2(G), \alpha_2(H)\}$ must hold.

Now, if $\alpha_{\mathrm{n}}(G \vee H) = 1$, then by the previous inequality we have the equality we were looking for. Let us then suppose that $\alpha_{\mathrm{n}}(G \vee H) > 1$, which by the first observations 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 $\alpha_{\mathrm{n}}(G \vee H)$. We define Y_H and Y_G as the sets of vertices of H and G respectively such that $Y_H = \{w \in V(H)\colon$ there exists $e \in M$ such that $w \in e\}$ and $Y_G = \{v \in V(G)\colon$ there exists $e \in$

M such that $v \in e\}$. 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 $d_G(v_1, v_2) \leq 2$. 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 $|Y_G| \leq \alpha_2(G)$ and $|Y_H| \leq \alpha_2(H)$, then $|M| = |Y_H| = |Y_G| \leq \min\{\alpha_2(G), \alpha_2(H)\}$ and therefore $\alpha_{\mathrm{n}}(G \vee H) \leq \min\{\alpha_2(G), \alpha_2(H)\}$, proving the reverse inequality and the theorem. □

Now we shall extend this results to the join of more than two graphs. For this purpose we state the following lemma, which is easy to prove but still useful.

Lemma 3 *If $G_1, \ldots, G_k$ are graphs and $k \geq 2$, then*

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

Proof Let $G = G_1 \vee \cdots \vee G_k$. Clearly, $\gamma(G) \leq \min\{2, \gamma(G_1), \ldots, \gamma(G_k)\}$ 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 $G_1 \vee \cdots \vee G_k$. 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 1 *If $G_1, \ldots, G_k$ are graphs and $k \geq 3$, then*

$$\rho_{\mathrm{n}}(G_1 \vee \cdots \vee G_k) = \min\{3, \gamma(G_1) + 1, \ldots, \gamma(G_k) + 1, \rho_{\mathrm{n}}(G_1), \ldots, \rho_{\mathrm{n}}(G_k)\}.$$

Proof The formula is valid when $k = 3$ because Lemmas 1 and 3 imply

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

Moreover, if the formula is valid when $k = t$ for some $t \geq 3$, then it is also valid when $k = t + 1$ since Lemma 3 implies

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

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

And now we state the following immediate consequence of Lemma 2, for future reference.

Corollary 2 *If $G_1, \dots, G_k$ are graphs and $k \geq 3$, then*

$$\alpha_{\mathrm{n}}(G_1 \vee \cdots \vee G_k) = 1.$$

Proof Since every two vertices of $G_1 \vee \cdots \vee G_{k-1}$ are at distance at most two, $\alpha_2(G_1 \vee \cdots \vee G_{k-1}) = 1$. Hence, Lemma 2 implies that $\alpha_{\mathrm{n}}(G_1 \vee \cdots \vee G_k) = \min\{\alpha_2(G_1 \vee \cdots \vee G_{k-1}), \alpha_2(G_k)\} = 1$. □

Using the previous two results, we will characterize which graphs that are formed by the join of two non-null subgraphs are minimally non-neighborhood-perfect or equivalently which are the only minimally non-neighborhood-perfect graphs that have a disconnected complement (Theorem 5). For this purpose, we shall first define a subclass of neighborhood-perfect graphs, the strongly neighborhood-perfect graphs.

Definition 2 We shall say that a graph G, is *strongly neighborhood-perfect* if $\alpha_2(G') = \rho_{\mathrm{n}}(G')$ for every induced subgraph G' of G.

Definition 3 We shall say that a graph G, is *minimally non-strongly neighborhood-perfect* when $\alpha_2(G) < \rho_{\mathrm{n}}(G)$, but $\alpha_2(G') = \rho_{\mathrm{n}}(G')$ for every proper induced subgraph G', of G. That is, it is not strongly neighborhood-perfect, but all its proper induced subgraphs are.

Remark 1 Clearly all strongly neighborhood-perfect graphs are neighborhood-perfect. It follows from the string of inequalities: $\alpha_2(G) \leq \gamma(G) \leq \alpha_{\mathrm{n}}(G) \leq \rho_{\mathrm{n}}(G)$, which was showed in Sect. 2 to hold 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 $\alpha_2(G') = \alpha_{\mathrm{n}}(G')$ for every induced subgraph G'.

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

Lemma 4 *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 $p \geq 2$ and there are at least two vertices of W which are p steps from each other in W 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. We take two vertices in W that are p steps from each other and either are adjacent or the same vertex in G, and consider the sub-walk of W of length p joining them. 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 5 *A graph G satisfies $\alpha_2(G') = \alpha_{\mathrm{n}}(G')$ for every induced subgraph G' of G if and only if G is P_6-free chordal.*

Proof First we observe that $\alpha_2(P_6) = 2$, while $\alpha_n(P_6) = 3$. Hence, G must be P_6-free for the equality of the parameters to hold for every induced subgraph.

Now, let G be P_6-free chordal, and 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 $\alpha_2(G) \leq \alpha_n(G)$ is 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 a vertex), such that $N[x] \cap N[s] \neq \emptyset$. Thus, there is a vertex $x' \in N[x] \cap N[s]$, but as $e, s \in S$, then $x' \notin N[y]$. 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 4, 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 coro, fully characterizing strongly neighborhood-perfect graphs by forbidden induced subgraphs.

Corollary 3 *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 Remark 1, 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 5, 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 (Lehel and Tuza 1986), 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 Theorem 1 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 use the previous characterization to prove which are the only minimally non-neighborhood-perfect graphs that have a disconnected complement.

Theorem 5 *The only minimally non-neighborhood-perfect graphs with disconnected complement are* $C_6 \vee 3K_1$, $P_6 \vee 3K_1$, *and* $C_4 \vee 2K_1 = \overline{3K_2}$.

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 Lemmas 1 and 2, 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, as was shown in Sect. 2).

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, 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 $\alpha_{\mathrm{n}} = \rho_{\mathrm{n}}$. 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 Lemmas 1 and 2, both have $\rho_{\mathrm{n}} = 3 \neq 2 = \alpha_{\mathrm{n}}$, 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 $\alpha_{\mathrm{n}} = \rho_{\mathrm{n}}$.

We need only to consider the case where one of the graphs does not satisfy $\alpha_2 = \rho_{\mathrm{n}}$. Let us say that G does not satisfy $\alpha_2(G) = \rho_{\mathrm{n}}(G)$ and, consequently, G has an induced subgraph G' isomorphic to C_4, C_6 or P_6. Now, as only G does not satisfy $\alpha_2(G) = \rho_{\mathrm{n}}(G)$, then $\alpha_2(G) < \rho_{\mathrm{n}}(G)$ and $\alpha_2(H) = \rho_{\mathrm{n}}(H)$. Hence $\alpha_2(G) < \alpha_2(H) \leq \alpha(H)$ 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 $\alpha_2(G') \leq \alpha_2(G) < \alpha(H)$. 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. □

4 Structural characterizations

In this section we shall characterize by minimal forbidden induced subgraphs the class of neighborhood-perfect graphs, restricted to the classes of P_4-tidy graphs (Theorem 6) and tree-cographs (Theorem 7). For this we will strongly rely on the characterization of minimally non-neighborhood-perfect graphs with disconnected complement shown in the previous section. We will use Theorem 5 together with the fact that every disjoint union of neighborhood-perfect graphs is neighborhood-perfect and the structures of P_4-tidy and tree-cographs given by Theorem 4 and Definition 1, respectively, to prove these characterizations.

We will first show the characterization restricted to P_4-tidy graphs. For this, let us begin by determining the values of $\alpha_{\mathrm{n}}(G)$ and $\rho_{\mathrm{n}}(G)$ for any connected and co-connected P_4-tidy graph G.

Lemma 6 *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, $\rho_{\mathrm{n}}(G) = 3$ and $\alpha_{\mathrm{n}}(G) = 2$.*
2. *G is isomorphic to P_5 or $\overline{P_5}$ and $\alpha_{\mathrm{n}}(G) = \rho_{\mathrm{n}}(G) = 2$.*
3. *G is a starfish with t ends or a fat starfish arising from one by substituting a vertex of $S \cup C$ by K_2 or $2K_1$, and $\alpha_{\mathrm{n}}(G) = \rho_{\mathrm{n}}(G) = t$.*
4. *G is an urchin or a fat urchin with at least 3 ends, and $\rho_{\mathrm{n}}(G) = 2$, $\alpha_{\mathrm{n}}(G) = 1$.*

Proof Since G is P_4-tidy, connected and co-connected, it follows by Theorem 4 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 such a starfish by the substitution of a vertex c of C by a K_2, or $2K_1$, or by the substitution of a vertex s from S by a K_2 or $2K_1$. If G is a starfish or a fat starfish with the substitution of a vertex from S then all C is a neighborhood cover set of size t, if on the other hand it is a fat starfish where a vertex c of C has been substituted by a K_2, we take only one of the vertices by which c has been substituted and the rest of C. 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 cover set of size t of G. Thus $\rho_n(G) \leq t$. 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 $\alpha_n(G) \geq t$. And as $\alpha_n(G) \leq \rho_n(G)$, 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 least 3 ends.

If G is an urchin or fat urchin with partition (S, C, R), and $|S| = t \geq 3$, we shall see that $\alpha_n(G) = 1$ and $\rho_n(G) = 2$. As there is no universal vertex in G, then $\rho_n(G) \geq 2$. 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 $t \geq 3$, then there must at least be one vertex of C that includes them both in its neighborhood. Therefore, $\alpha_n(G) = 1$. □

We then state and prove the forbidden induced subgraph characterization of neighborhood-perfect graphs restricted to the class of P_4-tidy graphs.

Theorem 6 *If G is a P_4-tidy graph, then it is neighborhood-perfect if and only if it is $\{\overline{3K_2}, 3{-}sun, C_5\}$-free.*

Proof If G is neighborhood-perfect, then it cannot contain as induced subgraph a $\overline{3K_2}$, 3-sun 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 is not neighborhood-perfect then it must contain $\overline{3K_2}$, 3-sun, or C_5 as an induced subgraph.

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 any 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 5 means that it must be $C_4 \vee 2K_1 = \overline{3K_2}$, $C_6 \vee 3K_1$ 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 $\overline{3K_2}$.

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 Lemma 6, 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 2, the only minimally non-neighborhood-perfect graphs with $\alpha_n(H) = 1$ are the $\overline{3K_2}$ and 3-sun and the only one of these that is an urchin is the 3-sun. Therefore, as H is connected and co-connected, it must be a C_5 or a 3-sun.

We conclude that H must be isomorphic to $\overline{3K_2}$, C_5 or 3-sun and since, by construction, H is an induced subgraph of G, this proves the theorem. □

Having proved Theorem 6, we proceed to prove the characterization of neighborhood-perfect graphs restricted to the class of tree-cographs. We shall work with the structural definition of a tree-cograph and strongly rely on the characterization of minimally non-neighborhood-perfect graphs with disconnected complement given in Theorem 5.

Lemma 7 *If G is a connected and co-connected tree-cograph, then one of the following statements holds:*

1. *G is a tree and $\rho_{\mathrm{n}}(G) = \alpha_{\mathrm{n}}(G) = \nu(G) = \beta(G)$,*
2. *G is a connected complement of a tree and $\rho_{\mathrm{n}}(G) = 2$.*

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 complement of a tree.

If G is a tree then it is bipartite. It was already noted in Lehel and Tuza (1986) and Sampathkumar and Neeralagi (1985) that for any bipartite graph G, $\alpha_{\mathrm{n}}(G) = \nu(G)$ and $\rho_{\mathrm{n}}(G) = \beta(G)$, which by the König-Egerváry theorem implies that $\alpha_{\mathrm{n}}(G) = \nu(G) = \beta(G) = \rho_{\mathrm{n}}(G)$.

If G is a connected complement of a 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 $\rho_{\mathrm{n}}(G) = 2$, proving the theorem. □

Corollary 4 *There are no connected and co-connected tree-cographs that are minimally non-neighborhood-perfect.*

Proof If a graph G is a connected and co-connected tree-cograph, then, by definition, G is a tree or the complement of a tree. By Lemma 7, a tree cannot be non-neighborhood-perfect. Moreoveer, if G is the complement of a tree, then $\rho_{\mathrm{n}}(G) = 2$, which means that if G is minimally non-neighborhood-perfect, then $\alpha_{\mathrm{n}}(G)$ must be 1. But by Theorem 2, the only minimally non-neighborhood-perfect graphs with $\alpha_{\mathrm{n}}(G) = 1$ are the 3-sun and the $\overline{3K_2}$, none of which are complements of trees. Hence if G is a minimally non-neighborhood-perfect graph, G cannot be a connected and co-connected tree-cograph. □

We are then ready to state and prove the characterization of neighborhood-perfect graphs restricted to the class of tree-cographs.

Theorem 7 *If G is a tree-cograph, then G is neighborhood-perfect if and only if G is $\{\overline{3K_2}, P_6 \vee 3K_1\}$-free.*

Proof It is clear that if G is neighborhood-perfect, it cannot have $\overline{3K_2}$ 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 5, it must be $C_4 \vee 2K_1 = \overline{3K_2}$, $C_6 \vee 3K_1$ 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 the complement of a 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 $\overline{3K_2}$ 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 4, if H is minimally non-neighborhood-perfect, then it cannot be a connected and co-connected tree-cograph. Hence H can only be $\overline{3K_2}$ or $P_6 \vee 3K_1$, and as H was by construction an induced subgraph of G, this proves the theorem. □

5 Algorithms and complexity results

In this section we shall present linear-time algorithms to solve the recognition problems of neighborhood-perfect graphs, as well the problems of finding an optimal neighborhood-independent set and neighborhood-covering set, restricted to the classes of P_4-tidy graphs and tree-cographs. Moreover, we shall prove that although the problem of determining $\alpha_{\mathrm{n}}(G)$ and $\rho_{\mathrm{n}}(G)$ can be solved in linear time for complements of trees, it becomes $\mathcal{NP}$-hard for complements of bipartite graphs.

5.1 Recognition algorithms

By using the particular structure of the modular decomposition trees of P_4-tidy graphs and tree-cographs, and Theorems 6 and 7, we will show two linear-time algorithms that solve the recognition problem of neighborhood-perfect graphs restricted to these two classes. Both of these algorithms work on the modular decomposition tree of the input graph.

We shall begin by describing a linear-time algorithm that decides whether or not a P_4-tidy graph is neighborhood-perfect. Let us first remember that, as was said in Sect. 2, it follows from Theorem 4 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.

Our recognition algorithm for neighborhood-perfect graphs, restricted to the class of P_4-tidy graphs, 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 6.

In order to simplify the recognition algorithm, we shall first define a boolean function $C : V(T(G)) \to \{True, False\}$, where $T(G)$ is the modular decomposition tree of G, and, for each node h of $T(G)$, $C(h) = True$ 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 neighborhood-perfect graphs in the class of P_4-tidy graphs.

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

Theorem 8 *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, by induction, 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

Algorithm 1: Computes $C(h)$ for every node of $T(G)$, with G a 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 $C(h) = True$ if and only if $G[h]$ contains an induced C_4

Step 1:
Traverse the nodes of $T(G)$ **in post-order, and in each node** h **do:**
- **if** *h is a leaf* **then** $C(h) := False$
- **else if** *$C(h')$ is True for any child h' of h* **or**
  h is a P-node having at least two nonleaf children **or**
  $\pi(h)$ is $\overline{P_5}$ **or**
  $\pi(h)$ is a starfish or an urchin and any vertex of its body represents $2K_1$ **then**
  - $C(h) := True$
- **else** $C(h) := False$

Step 2:
- 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

Step 1:
Traverse every node h **of** $T(G)$ **in any order and do:**
- **if** *h is an N-node* **then**
  - **if** *$\pi(h)$ is a C_5 or an urchin with at least 3 ends* **then**
    - **output** "G is not neighborhood-perfect" and **stop**
  - **else if** *$\pi(h)$ is a fat starfish such that a vertex of its body represents $2K_1$ and $C(h_r)$ is True for h_r the child representing the head of $\pi(h)$* **then**
    - **output** "G is not neighborhood-perfect" and **stop**
- **else 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 children h_1 and h_2 and at least one of $C(h_1)$ and $C(h_2)$ is True* **then**
    - **output** "G is not neighborhood-perfect" and **stop**

Step 2:
- **output** "G is neighborhood-perfect"

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 3 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 4 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 nonleafs. 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 h 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 N-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 time $\mathcal{O}(\sum_{h \text{ an N-node}} n_\pi(h))$. As was already stated in Sect. 2, the sum of $n_\pi(h)$ for all N-nodes h is at most $2n$. Therefore the whole algorithm can be implemented in $\mathcal{O}(n)$ time. □

Theorem 9 *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 $\{C_5, \text{3-sun}, \overline{3K_2}\}$-free. This together with Theorem 6 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. There are thus four possible cases. If it stopped in line 5, then G contains an induced C_5 or 3-sun, because every urchin with at least 3 ends contains a 3-sun. If it stopped in line 7 then $\pi(h)$ is a fat starfish, such that the vertices of h_r induce a C_4 and the and a vertex of the body induces a $2K_1$ in G. But then, there is an induced $2K_1 \vee C_4 = \overline{3K_2}$ in G. In the other two possible cases, the h that causes the algorithm to stop 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 10, then any set consisting of a pair of nonadjacent vertices of each of the three nonleaf children of h induces $2K_1 \vee 2K_1 \vee 2K_1 = \overline{3K_2}$ in G. On the other hand, if the algorithm stopped in line 12, 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 $C_4 \vee 2K_1 = \overline{3K_2}$ in G as well. We conclude that if the algorithm outputs that G is not neighborhood-perfect, then G contains an induced C_5, 3-sun or $\overline{3K_2}$.

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 3-sun. By Theorem 3, there is some N-node h of $T(G)$ such that $\pi(h)$ contains an induced C_5 or 3-sun. By Theorem 4, $\pi(h)$ 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 $\overline{3K_2}$. Let h be a node of $T(G)$ such that $G[h]$ contains an induced $\overline{3K_2}$ 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 an S-node. If h is an N-node and $G[h]$ contains an induced $\overline{3K_2}$, then $\pi(h)$ must be an urchin or a starfish. However, if $\pi(h)$ were an urchin, it would contain a 3-sun. Hence, because we have already covered the case with an induced 3-sun, 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, the $\overline{3K_2}$ 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 $\overline{3K_2}$ 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 $\overline{3K_2}$ has no universal vertex, then each induced $\overline{3K_2}$ 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 $\overline{3K_2}$ 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 $\overline{3K_2} = C_4 \vee 2K_1 = 2K_1 \vee 2K_1 \vee 2K_1$, the only two possibilities for $G[h]$ to have an induced $\overline{3K_2}$ 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. That the Initialization can be performed in $\mathcal{O}(n+m)$ time follows from the remarks made in Sect. 2.2, and Theorem 8 (in particular if the modular decomposition is already given, the Initialization can be performed in $\mathcal{O}(n)$ time). In Step 1 we perform $\mathcal{O}(n_\pi(h))$ operations in every node h of $T(G)$. 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. □

Having presented the algorithm for P_4-tidy graphs we shall give another one to decide neighborhood-perfectness of tree-cographs in linear-time.

As was already pointed out in Sect. 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 7 as tree-cographs having no $\overline{3K_2}$ 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 $P : V(T(G)) \to \{True, False\}$, such that $P(h) = True$ if and only if $G[h]$ has an induced P_6. And let $\alpha : V(T(G)) \to \mathbb{N}$, 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 above, all in $\mathcal{O}(n)$ time, given the modular decomposition tree. It uses the fact that computing $\alpha(T)$ can be done in time $\mathcal{O}(|V(T)|)$, for any tree T (Savage 1982).

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

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

Theorem 10 *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.

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 $T(G)$

Step 1:
Traverse the nodes of $T(G)$ **in post-order, and in each node** h **do:**
  if *h is a leaf* **then** $C(h) := P(h) := False$ and $\alpha(H) := 1$
  else if *h is a P-node with children* $h_1, \ldots, h_k$ **then**
    $C(h) := \bigvee_{i=1}^{k} C(h_i)$, $P(h) := \bigvee_{i=1}^{k} P(h_i)$, $\alpha(h) := \sum_{i=1}^{k} \alpha(h_i)$
  else if *h is a S-node with children* $h_1, \ldots, h_k$ **then**
    $\alpha(h) := \max\{\alpha(h_i) \colon 1 \leq i \leq k\}$, $P(h) := \bigvee_{i=1}^{k} P(h_i)$,
    if *h has at least two nonleaf children* **then** $C(h) := True$
    else $C(h) := \bigvee_{i=1}^{k} C(h_i)$
  else if $\pi(h)$ *is a tree with children* $h_1, \ldots, h_k$ **then**
    compute $\alpha(G[h])$ in linear-time and assign it to $\alpha(h)$, $C(h) := False$
    if *the longest path in* $\pi(h)$ *is of length at least* 6 **then** $P(h) := True$
    else $P(h) := False$
  else if $\pi(h)$ *is the complement of a tree with children* $h_1, \ldots, h_k$ **then**
    $\alpha(h) := 2$, $P(h) := False$
    if $\overline{\pi(h)}$ *has an induced matching of size at least* 2 **then**
      $C(h) := True$
    else $C(h) := False$

Step 2:
  Output $C(h)$, $P(h)$ and $\alpha(h)$ for every node h of $T(G)$

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 children* 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* $\alpha(h_2) \geq 3$ *or vice versa* **then**
        output "G is not neighborhood-perfect" and **stop**
  if *h is an N-node, with* $\pi(h)$ *the complement of a 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"

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

If h is an S-node, then 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 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 an induced subgraph of $G[h]$ if and only if it is an 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 the complement of a 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 the complement of a tree, in which the number of operations is in $\mathcal{O}(n_\pi(h) + m_\pi(h))$. As mentioned in Sect. 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 $\pi(h)$ the complement of a 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 Mitchell et al. (1975, 1979) and Savage (1982) 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 Bulterman et al. (2002), 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 when h is an N-node, with $\pi(h)$ isomorphic to the complement of a tree. Because $\pi(h)$ has $\Theta((n_\pi(h)^2)$ edges, then it can be complemented in $\mathcal{O}(m_\pi(h))$ time. Once complemented, the size of the greatest induced matching can be determined in $\mathcal{O}(n_\pi(h)$ time using any of the algorithms in Fricke and Laskar (1992); Golumbic and Lewenstein (2000) and Zito (2000). This fact together with the observations made in the previous paragraph imply that the whole algorithm can be implemented to run in $\mathcal{O}(n+m)$ time. □

Theorem 11 *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 9, but using the subgraph characterization of neighborhood-perfect graphs among tree-cographs proved in Theorem 7. We will then prove that the algorithm outputs that the graph G is neighborhood-perfect if and only if G is $\{P_6 \vee 3K_1, \overline{3K_2}\}$-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 $G[h]$ must contain an induced $\overline{3K_2} = 2K_1 \vee 2K_1 \vee 2K_1$; if it stops in line 8 then it must contain an induced $C_4 \vee 2K_1 = \overline{3K_2}$. 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 the complement of a tree that contains an induced $\overline{3K_2}$.

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. 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 the complement of a tree. If h is a S-node and $G[h]$ contains an induced $\overline{3K_2}$, then, as was shown in the proof of Theorem 9, 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 8, 5, and 10. Finally if h is an N-node, with $\pi(h)$ the complement of a tree, then $G[h]$ 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 the complement of a tree. If $G[h]$ contains an induced $\overline{3K_2}$, 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 10. First recall that as was mentioned in Sect. 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 an N-node with $\pi(h)$ a tree, the algorithm does no operations. If h is an S-node, then it clearly can 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 $\pi(h)$ the complement of a tree, then, as $m_\pi(h) \in \Theta(n_\pi(h)^2)$, we can complement $\pi(h)$ in $\mathcal{O}(m_\pi(h))$ time. Once complemented, we can use any of the linear-time algorithms in Fricke and Laskar (1992), Zito (2000) and Golumbic and Lewenstein (2000), 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 N-node h and to $m_\pi(h)$ for the N-nodes with $\pi(h)$ the complement of a tree, which implies that it runs in $\mathcal{O}(n+m)$ time for the whole graph. □

5.2 Optimal sets algorithms

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 graphs and tree-cographs. 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 $R_{\mathrm{n}}(h)$ be a list of vertices of G that form a neighborhood set of $G[h]$ of minimum size, $A_{\mathrm{n}}(h)$ 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 also for each node h, thus obtaining these lists for the root of $T(G)$, which we shall call $A_{\mathrm{n}}(G)$, $R_{\mathrm{n}}(G)$, $A_2(G)$ 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 $l_1, \ldots, l_k$, with $1 \leq i \leq k$ 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 an N-node h, then the Algorithm 5 works in linear time with respect to G.

Theorem 12 *Algorithm 5 obtains correctly $A_{\mathrm{n}}(G)$, $R_{\mathrm{n}}(G)$, $A_2(G)$ 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 $h_1, \ldots, h_k$ 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 $A_2(h)$, $D(h)$, $A_{\mathrm{n}}(h)$, $R_{\mathrm{n}}(h)$ 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 Lemmas 1, 2, 3, and Corollaries 1, 2), the lists build in this way are optimal lists. □

Lemma 8 *Let $c(h)$ be the number of edges made in line 11 of Algorithm 5 if $k = 2$, for h and all the descendants of h in a modular decomposition tree $T(G)$. Then, for every node h, $c(h) + \alpha_2(h) \leq n(h)$, 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, $1 \leq i \leq k$ 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

Algorithm 5: Computes $A_{\mathrm{n}}(G)$, $R_{\mathrm{n}}(G)$, $A_2(G)$, $D(G)$ 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: $A_{\mathrm{n}}(G)$, $R_{\mathrm{n}}(G)$, $A_2(G)$ and $D(G)$
Initialization: Construct $T(G)$, the modular decomposition tree of G

Step 1:
Traverse the nodes of $T(G)$ in post-order, and in each node h do:
  if *h is a leaf, representing only $v \in V(G)$* **then**
    $A_{\mathrm{n}}(h) := \langle v\rangle$, $R_{\mathrm{n}}(h) := \langle v\rangle$, $A_2(h) := \langle v\rangle$, $D(h) := \langle v\rangle$
  else if *h is a P-node with children $h_1, \ldots, h_k$* **then**
    $R_{\mathrm{n}}(h) := \sum_{i=1}^{k} R_{\mathrm{n}}(h_i)$, $A_2(h) := \sum_{i=1}^{k} A_2(h_i)$, $D(h) := \sum_{i=1}^{k} D(h_i)$,
    $A_{\mathrm{n}}(h) := \sum_{i=1}^{k} A_{\mathrm{n}}(h_i)$
  else if *h is an S-node with children $h_1, \ldots, h_k$* **then**
    $A_2(h) := \langle v\rangle$ where v is an arbitrary vertex of $G[h]$
    $D(h) :=$ a list of minimum length among $D(h_1), \ldots, D(h_k)$, $\langle v_1, v_2\rangle$ for any $v_1 \in V(G[h_1])$ and $v_2 \in V(G[h_2])$
    if $k = 2$ **then**
      $A_{\mathrm{n}}(h) := \langle (A_2(h_1)[i], A_2(h_2)[i]) \colon 1 \le i \le \min\{|A_2(h_1)|, |A_2(h_2)|\}\rangle$
    else
      $A_{\mathrm{n}}(h) := \langle (v_1, v_2)\rangle$ for any $v_1 \in V(G[h_1])$ and $v_2 \in V(G[h_2])$
    $R^* :=$ a list of minimum length among $D^*(h_1), \ldots, D^*(h_k)$, $R_{\mathrm{n}}(h_1), \ldots, R_{\mathrm{n}}(h_k)$, where $D^*(h_i) = D(h_i) + \langle v\rangle$ for any $v \in G[h] \setminus G[h_i]$
    if $k = 2$ **then**
      $R_{\mathrm{n}}(h) := R^*$
    else
      $R_{\mathrm{n}}(h) :=$ a list of minimum length between R^* and $\{v_1, v_2, v_3\}$, where $v_i \in V(G[h_i])$ for $i \in \{1, 2, 3\}$
  else if *h is an N-node* **then**
    Use a graph class specific subroutine to calculate $A_{\mathrm{n}}(h)$, $R_{\mathrm{n}}(h)$, $A_2(h)$ and $D(h)$

Step 2:
  Output $A_{\mathrm{n}}(G)$, $R_{\mathrm{n}}(G)$, $A_2(G)$, $D(G)$

inequality $|I \cap V(h_i)| \le \alpha_2(h_i)$ must hold for every 2-independent set I of $G[h]$. Hence, by the induction hypothesis, $c(h)+\alpha_2(h) \le \sum_{i=1}^{k} c(h_i)+\sum_{i=1}^{k} \alpha_2(h_i) \le \sum_{i=1}^{k} n(h_i) = n(h)$.

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

Theorem 13 *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 $D(h)$, $A_2(h)$, $A_\mathrm{n}(h)$, and $R_\mathrm{n}(h)$, 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 8, 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 time $\mathcal{O}(n+m)$, the whole algorithm can be implemented to run in $\mathcal{O}(n+m)$ time. □

Now that we have the "general" algorithm, 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 $A_\mathrm{n}(h)$, $R_\mathrm{n}(h)$, $A_2(h)$, $D(h)$, for a given N-node h of a modular decomposition tree $T(G)$ of a P_4-tidy graph G

Input: An N-node h of a modular decomposition tree of a P_4-tidy graph G
Output: $A_\mathrm{n}(h)$, $R_\mathrm{n}(h)$, $A_2(h)$ and $D(h)$

Step 1:
- **if** *$\pi(h)$ is isomorphic to $C_5 = v_1 \dots v_5 v_1$* **then**
  - $A_\mathrm{n} := \langle v_1v_2, v_4v_5\rangle$, $R_\mathrm{n}(h) := \langle v_1, v_3, v_5\rangle$, $A_2(h) := \langle v_1\rangle$, $D(h) := \langle v_1, v_3\rangle$
- **else if** *$\pi(h)$ is isomorphic to $P_5 = v_1 \dots v_5$* **then**
  - $A_\mathrm{n} := \langle v_1v_2, v_4v_5\rangle$, $R_\mathrm{n}(h) := D(h) := \langle v_2, v_4\rangle$, $A_2(h) := \langle v_1, v_4\rangle$
- **else if** *$\pi(h)$ is isomorphic to $\overline{P_5}$ with $\overline{\pi(h)} = v_1 \dots v_5$* **then**
  - $A_\mathrm{n} := \langle v_1v_5, v_2v_4\rangle$, $R_\mathrm{n}(h) := D(h) = \langle v_1, v_2\rangle$, $A_2(h) = \langle v_1\rangle$
- **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**
  - Let $v_i \in V(G[c_i])$ and $w_i \in V(G[s_i])$ for each $i \in \{1, \dots, k\}$
  - $A_2(h) := \langle w_1, \dots, w_k\rangle$, $D(h) := \langle v_1, \dots, v_k\rangle$, $A_\mathrm{n}(h) := \langle v_1w_1, \dots, v_kw_k\rangle$, $R_\mathrm{n}(h) := \langle v_1, \dots, v_k\rangle$
  - **if** *$\pi(h)$ is a fat starfish with $c_i \in C$ representing $2K_1$* **then**
    - Replace v_i in $R_\mathrm{n}(h)$ with w_i
- **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**
  - $A_2(h) := \langle v_1\rangle$, $D(h) := \langle v_1, v_2\rangle$, $A_\mathrm{n} := \langle v_1w_2\rangle$ and $R_\mathrm{n}(h) := \langle v_1, v_2\rangle$ for any $v_1 \in V(G[c_1])$, $v_2 \in V(G[c_2])$ and $w_2 \in V(G[s_2])$.

Step 2:
- Output $A_\mathrm{n}(h)$, $R_\mathrm{n}(h)$, $A_2(h)$, $D(h)$

Theorem 14 *Algorithm 6 correctly finds $A_\mathrm{n}(h)$, $R_\mathrm{n}(h)$, $A_2(h)$, $D(h)$, 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 $A_2(h)$, $D(h)$, $A_\mathrm{n}(h)$ and $R_\mathrm{n}(h)$ 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 $A_\mathrm{n}(h)$ has the same length as $R_\mathrm{n}(h)$. 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 $R_n(h)$ built 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 15 *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 have already seen in Sect. 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. □

Now we shall 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 give a characterization of connected complements of trees with $\alpha_n > 1$. This characterization will allow us to easily identify these graphs and find a neighborhood-independent set of maximum size, all in linear-time. Notice that a total dominating set is a dominating set inducing a subgraph with no isolated vertices, and $\gamma_t(G)$ denotes the minimum size of any total dominating set of G.

Lemma 9 *If G is a connected graph then, $\alpha_n(G) > 1$ if and only if G has at least two neighborhood-independent edges.*

Proof Clearly if G has two neighborhood-independent edges, then $\alpha_n(G) > 1$. Now, if $\alpha_n(G) > 1$, let S be a maximum neighborhood-independent set with maximum amount of edges. Let us assume by contradiction that S has less than two edges of G. Because $|S| > 1$, then S must have at least one vertex. If we take any such vertex x, because G is connected, there is an edge xy in $E(G)$. Clearly $xy \notin S$, moreover because x is neighborhood-independent to all elements of S, then xy must be as well. Therefore $S \setminus \{x\} \cup \{xy\}$ is a maximum cardinality neighborhood-independent set with one more edge than S. □

Lemma 10 *If G is a connected 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 Lemma 9, $\alpha_n(G) > 1$, 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 $\overline{G}$. □

For the following results we will denote by T' a tree obtained from a tree T by deleting all leaves of T.

Lemma 11 *If G is a connected complement of a tree with $\alpha_n(G) > 1$, and $T = \overline{G}$, then T' must be a path.*

Proof If G is a connected complement of a 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 leafs, 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 leafs in T, but as they are leafs of T', then they must have been adjacent to leafs 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 10 implies that there must be a vertex w such that $\{x, y, w, z\}$ is a total dominating set of T and by symmetry we may assume that xy and wz are edges of G; i.e., non-edges of T. But then, as z is nonadjacent in T to each of x, y, and w, the vertex z is not strongly dominated by $\{x, y, z, w\}$ in T, a contradiction. The contradiction came from the supposition that T' was not a path. □

Before presenting the characterization, we shall state an inequality that will be used in the proof of Lemma 12.

Theorem 16 (Chellali and Haynes 2006) *The following inequality holds for every tree T:*

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

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.

Lemma 12 *If G is a connected complement of a tree and $T = \overline{G}$, then $\alpha_n(G) > 1$ if and only if T' is either P_2, P_3, P_4, or T' is P_5 or P_6 with no central vertex of T' adjacent to a leaf of T.*

Proof Let us first prove that if G is a connected complement of a tree with $\alpha_n(G) > 1$, then T' is as described above. By Lemma 11, T' must be a path. Clearly T' cannot have only 1 vertex, because T would be a star and G would be disconnected. As we have already seen in Lemma 10, $\gamma_t(T) \leq 4$ if $T = \overline{G}$. Thus, Theorem 16 implies that $6 \geq n(T) - l(T)$, but $n(T) - l(T) = n(T')$. Therefore T' is P_i with $2 \leq i \leq 6$. 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 11, 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 $\gamma_t \leq 4$. 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 leafs of T, respectively, adjacent to different leafs 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 10, $\alpha_n(G) > 1$. 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 $\alpha_n(G) > 1$. □

Corollary 5 *If G is the complement of a tree, it can be decided in $\mathcal{O}(n + m)$ time whether $\alpha_n(G) > 1$ and if so find a neighborhood-independent set of G with size 2.*

Proof We use the characterization presented in Lemma 12. We can easily complement G and remove the vertices with degree 1. If the resulting tree is a P_2, P_3, P_4, or a P_5 or P_6 satisfying that no central vertex was adjacent to one of the removed vertices, then $\alpha_{\mathrm{n}}(G) > 1$ and, following the instructions of the proof of Lemma 12, we can obtain the two neighborhood-independent edges of G. As G is the complement of a tree, $m \in \mathcal{O}(n^2)$ meaning that we can complement G in time $\mathcal{O}(m)$. Deciding whether a tree becomes a path of bounded size by removing its leafs and, if so, also computing the corresponding path, can all be done in $\mathcal{O}(n)$. Finally obtaining the edges following the instructions of Lemma 12 can be easily done in time $\mathcal{O}(n)$. ◻

Algorithm 7: Computes $A_{\mathrm{n}}(h)$, $R_{\mathrm{n}}(h)$, $A_2(h)$, $D(h)$, for a given N-node h of the modular decomposition tree $T(G)$ of a tree-cograph G

Input: An N-node h of a modular decomposition tree of a tree-cograph G
Output: $A_{\mathrm{n}}(h)$, $R_{\mathrm{n}}(h)$, $A_2(h)$ and $D(h)$
1 **Step 1:**
2   **if** $\pi(h)$ *is a tree* **then**
3     $A_2(h) :=$ a maximum 2-independent set of $G[h]$
4     $D(h) :=$ a minimum dominating set of $G[h]$
5     $A_{\mathrm{n}}(h) :=$ a maximum matching of $G[h]$
6     $R_{\mathrm{n}}(h) :=$ a minimum vertex cover of $G[h]$
7   **else if** $\pi(h)$ *is the complement of a tree* **then**
8   **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
9   **else** $A_2(h) := \langle v_1 \rangle$ for any $v_1 \in G[h]$
10   **if** $\alpha_{\mathrm{n}}(\pi(h)) > 1$ **then** $A_{\mathrm{n}}(h) := \{e_1, e_2\}$ with e_1, e_2 neighborhood-independent edges of $G[h]$
11   **else** $A_{\mathrm{n}}(h) = \{e_1\}$ with e_1 any edge of $\pi(h)$
12     $D(h) := \langle v_l, v_n \rangle$, $R_{\mathrm{n}}(h) := \langle v_l, v_n \rangle$, with v_l a leaf of $\overline{G[h]}$ and v_n its only neighbor in $\overline{G[h]}$
13 **Step 2:**
14   Output $A_{\mathrm{n}}(h)$, $R_{\mathrm{n}}(h)$, $A_2(h)$, $D(h)$

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 13 *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. ◻

Theorem 17 *Algorithm 7 correctly finds $A_{\mathrm{n}}(h)$, $R_{\mathrm{n}}(h)$, $A_2(h)$ 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 complement of a 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 Lehel and Tuza (1986) 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 the complement of a tree, then as was seen in Lemma 13, 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 the complement of a tree cannot have an independent set of size three, thus $\alpha_{\mathrm{n}}(\pi(h)) \leq 2$. 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 5 that there is a linear-time algorithm to determine if $\alpha_{\mathrm{n}}(\pi(h)) > 1$ and if this is the case to find a neighborhood-independent set of size 2. Thus in line 10, we correctly obtain $A_{\mathrm{n}}(h)$. Note that $\alpha_{\mathrm{n}}(G[h]) \leq 2$, 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 18 *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 the complement of a tree because all N-nodes of a tree-cograph are trees or complements of trees. If $\pi(h)$ is a tree, linear-time algorithms for finding minimum vertex cover sets, minimum dominating sets and maximum matchings can be found in Mitchell et al. (1975, 1979) and Savage (1982). Obtaining a 2-independent maximum set of a tree can also be done efficiently with an algorithm very similar to the one mentioned in Mitchell et al. (1979) 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 Use(i), NUse(i), and NUseC(i) where Use(i) is a maximum 2-independent set using vertex i in the subtree rooted at i, NUse(i) is defined analogously but without using i, and NUseC(i) without using either i or the children of i. Thus, if i is not a leaf, $\mathrm{Use}(i) = i \cup \bigcup(\,\mathrm{NUseC}(j) : j$ is a child of $i)$ and $\mathrm{NUse}(i) = \max\{\bigcup(\,\mathrm{NUse}(j) : j \neq k) \cup \max\{\mathrm{Use}(k),\ \mathrm{NUse}(k)\} : k$ is child of $i\}$, where $\max\{A, B\}$ denotes a set with maximum number of vertices among A and B. Moreover, $\mathrm{NUseC}(i) = \bigcup(\,\mathrm{NUse}(j) : j$ is child of $i)$. If i is a leaf, then $\mathrm{Use}(i) = \{i\}$ and $\mathrm{NUse}(i) = \mathrm{NUseC}(i) = \emptyset$. This implies that, Use(i), NUse(i), and NUse(i) can be determined in overall linear-time for all vertices i. Finally, $\max\{\mathrm{Use}(i), \mathrm{NUse}(i)\}$, 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 the complement of a 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 Laskar et al. (1984), Chellali and Haynes (2006) and Henning and Yeo (2013), 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 5, we can find in time $\mathcal{O}(n_\pi(h) + m_\pi(h))$ 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 the complement of a tree, the algorithm can also be implemented to run in $\mathcal{O}(n_\pi(h) + m_\pi(h))$ time. □

5.3 Complexity results

Theorems 13 and 18 imply that the problem of finding $\alpha_{\mathrm{n}}(G)$ and $\rho_{\mathrm{n}}(G)$ can be solved in linear-time if G is the complement of a tree. Nevertheless, as was already stated, the problems of determining these two parameters for general graphs have been proven to be NP-hard in Chang et al. (1993). We prove here that even if G belong to the class of complement of bipartite graphs, that includes the class of complements of trees, these problems are NP-hard.

Theorem 19 *It is* NP*-hard to determine* $\alpha_{\mathrm{n}}(G)$ *and* $\rho_{\mathrm{n}}(G)$ *when* G *is the complement of a bipartite graph.*

We shall denote complement of bipartite graphs as co-bipartite graphs. The proofs of Lemmas 14 and 15 together constitute a proof of Theorem 19.

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.

Lemma 14 *It is* NP*-hard to determine the neighborhood independence number in co-bipartite graphs.*

Proof We shall prove the 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 $\alpha_{\mathrm{n}}(G) = \alpha(H)$.

Given any graph $H = (V, E)$, let $G = (X, Y, F)$ where $X = \{v' \colon v \in V\}$, $Y = V \cup E$ and $F = \{v'e \colon v \in V, e \in E \text{ and } v \text{ is incident to } e\} \cup \{v'v \colon v \in V\}$; 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 $S \in V$ of H, let I be the subset of F defined by $I = \{v'v \colon v \in S\}$. By definition, $|I| = |S|$. We can 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 $\alpha_{\mathrm{n}}(G) \geq \alpha(H)$.

Conversely, let I be a neighborhood-independent set of edges in G such that $I \subseteq F$. We shall see that $S = \{v \in V : v'y \in I\}$ is an independent set of H. Suppose, by contradiction, that there is an edge e of H joining two vertices v and w of S. By definition, there are $y_1, y_2 \in Y$ such that $v'y_1, w'y_2 \in F$ 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 $\alpha(H) \geq \alpha_{\mathrm{n}}(G)$. 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. □

To prove the NP-hardness of determining the neighborhood number of co-bipartite graphs, we will use the following result from Dinur and Safra (2005).

Theorem 20 *(Dinur and Safra 2005) Given a graph* G*, it is* NP*-hard to approximate the Minimum Vertex Cover to within any factor smaller than* $10\sqrt{5} - 21 = 1.3606\ldots$.

Lemma 15 *It is* NP*-hard to determine the neighborhood number in co-bipartite graphs.*

Proof To prove that the problem is NP-hard, we shall use Theorem 20, 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 $\beta(H) \leq \rho_{\mathrm{n}}(G) \leq \beta(H) + 1$, where $\beta(H)$ is the size of the minimum vertex cover of H. Namely, given any graph $H = (V, E)$, let $G = (X, Y, F)$ where $X = V$, $Y = E$ and $F = \{ve \in V \times E\colon v$ is incident to e 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 $\rho_{\mathrm{n}}(G) \leq \beta(H) + 1$.

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 $\rho_{\mathrm{n}}(G)$. 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 $\rho_{\mathrm{n}}(G)$, $\beta(H) \leq \rho_{\mathrm{n}}(G)$.

Now that we have proved that this co-bipartite graph G satisfies $\beta(H) \leq \rho_{\mathrm{n}}(G) \leq \beta(H) + 1$, 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 $\beta(H) \geq 3$ we construct in polynomial time G as described above. As proven before $\beta(H) \leq \rho_{\mathrm{n}}(G) \leq \beta(H) + 1$, which as $\beta(H) \geq 3$ means that $1 \leq \frac{\rho_{\mathrm{n}}(G)}{\beta(H)} \leq 1 + \frac{1}{3}$. 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. □

Remark 2 We observe that obtaining the other two optimal sets considered in this section, namely, a maximum 2-independent set and a minimum dominating set, can be done in linear time even for co-bipartite graphs. A maximum 2-independent set would be any pair of nonadjacent vertices if there is any such pair, or any vertex if not (that would necessarily be a universal vertex). While a minimum dominating set would be either a universal vertex (if there is one), or a pair of vertices, one from each bipartition of the complement.

6 Further remarks

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 in Courcelle et al. (2000). This approach allows for linear-time solutions of recognition and optimization problems that are expressible in a certain monadic second-order logic. Nevertheless, it is not clear whether the definition of neighborhood-perfectness can be expressed in the corresponding second-order logic. However, by virtue of our Theorem 6, neighborhood-perfectness restricted to P_4-tidy graphs is equivalent to the absence of a finite number of fixed induced subgraphs. As the absence of a fixed induced subgraph can be expressed in the logic and a clique-width expression of any P_4-tidy graph can be obtained in linear-time (Courcelle et al. 2000), Courcelle et al.'s methateorem together with our Theorem 6 gives an alternative proof of the fact that neighborhood-perfectness can be detected in linear time for P_4-tidy graphs. Similarly, as the class of tree-cographs also has bounded clique-width and the clique-width expression can be found in linear-time, Courcelle et al.'s metatheorem together with our Theorem 7 would give an alternative proof of the existence of a linear-time algorithm for the recognition problem of neighborhood-perfectness restricted to the class of tree-cographs.

Nevertheless, we stress the necessity of Theorems 6 and 7 for proving the existence of such linear-time recognition algorithms using the approach of Courcelle et al. (2000). Still, although Courcelle et al.'s metatheorem is of great theoretical importance, the algorithm obtained by it is far away from being practical; because it may have enormous hidden constants in the linear-time complexity [even if the input graph has small clique-width, see Courcelle (2008)]. 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 (Frick and Grohe 2004). 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 as those developed in Sect. 5. In addition, linear-time algorithms for solving the neighborhood-independence number problem for P_4-tidy graphs and tree-cographs, as those given in Sect. 5.2, do not seem to follow from the approach of Courcelle et al. (2000) as the problem is not directly expressible in the corresponding logic (as quantification over subsets of edges is not allowed).

Acknowledgements We would like to thank the anonymous reviewers for their suggestions and comments that helped improve the quality of this paper. This work was partially supported by UBACyT Grant 20020130100808BA (Argentina), CONICET PIP 112-201201-00450CO and PIO 14420140100027CO (Argentina), ANPCyT PICT 2015-2218 (Argentina), UNS PGI 24/ZL16 (Argentina), FONDECyT Grant 1140787 (Chile), and Institute for Complex Engineering Systems (ICM-FIC: P05-004-F, CONICYT: FB0816, Chile).

References

Baumann, S. (1996). A linear algorithm for the homogeneous decomposition of graphs. Report TUM M9615. Munich: Fakultt fr Mathematik, Technische Universitt Mnchen.

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

Bonomo, F., Durán, G., & Groshaus, M. (2007). Coordinated graphs and clique graphs of clique-helly perfect graphs. *Utilitas Mathematica*, *72*, 175–192.

Brandstädt, A., Chepoi, V. D., & Dragan, F. F. (1997). Clique r-domination and clique r-packing problems on dually chordal graphs. *SIAM Journal on Discrete Mathematics*, *10*(1), 109–127. https://doi.org/10.1137/S0895480194267853.

Buer, H., & Möhring, R. H. (1983). A fast algorithm for the decomposition of graphs and posets. *Mathematical Operations Research*, *8*(2), 170–184. https://doi.org/10.1287/moor.8.2.170.

Bulterman, R. W., van der Sommen, F. W., Zwaan, G., Verhoeff, T., van Gasteren, A. J. M., & Feijen, W. H. J. (2002). On computing a longest path in a tree. *Information Processing Letters*, *81*(2), 93–96. https://doi.org/10.1016/S0020-0190(01)00198-3.

Chang, G. J., Farber, M., & Tuza, Z. (1993). Algorithmic aspects of neighborhood numbers. *SIAM Journal on Discrete Mathematics*, *6*(1), 24–29.

Chellali, M., & Haynes, T. W. (2006). A note on the total domination number of a tree. *Journal of Combinatorial Mathematics and Combinatorial Computing*, *58*, 189–193.

Chudnovsky, M., Robertson, N., Seymour, P., & Thomas, R. (2006). The strong perfect graph theorem. *Annals of Mathematics*, *164*(1), 51–229. https://doi.org/10.4007/annals.2006.164.51.

Courcelle, B. (2008). A multivariate interlace polynomial and its computation for graphs of bounded clique-width. *The Electronic Journal of Combinatorics*, *15*(1), R69.

Courcelle, B., Makowsky, J. A., & Rotics, U. (2000). Linear time solvable optimization problems on graphs of bounded clique-width. *Theory of Computing Systems*, *33*(2), 125–150. https://doi.org/10.1007/s002249910009.

Cournier, A., & Habib, M. (1994). A new linear algorithm for modular decomposition. In *Trees in algebra and programming—CAAP '94 (Edinburgh, 1994), Lecture notes in computer science* (Vol. 787, pp. 68–84). Berlin: Springer. https://doi.org/10.1007/BFb0017474.

Dahlhaus, E., Gustedt, J., & McConnell, R. M. (2001). Efficient and practical algorithms for sequential modular decomposition. *Journal of Algorithms*, *41*(2), 360–387. https://doi.org/10.1006/jagm.2001.1185.

Dinur, I., & Safra, S. (2005). On the hardness of approximating minimum vertex cover. *Annals of Mathematics*, *162*(1), 439–485. https://doi.org/10.4007/annals.2005.162.439.

Fouquet, J. L., & Giakoumakis, V. (1997). On semi-P_4-sparse graphs. *Discrete Mathematics*, *165/166*, 277–300. https://doi.org/10.1016/S0012-365X(96)00177-X. Graphs and combinatorics (Marseille, 1995).

Frick, M., & Grohe, M. (2004). The complexity of first-order and monadic second-order logic revisited. *Annals of Pure and Applied Logic*, *130*(1–3), 3–31. https://doi.org/10.1016/j.apal.2004.01.007.

Fricke, G., & Laskar, R. (1992). Strong matchings on trees. In Proceedings of the Twenty-third Southeastern International Conference on Combinatorics, Graph Theory, and Computing (Boca Raton, FL, 1992), vol 89, pp 239–243.

Gallai, T. (1967). Transitiv orientierbare graphen. *Acta Mathematica Academiae Scientiarum Hungaricae*, *18*, 25–66.

Giakoumakis, V., Roussel, F., & Thuillier, H. (1997). On P_4-tidy graphs. *Discrete Mathematics and Theoretical Computer Science*, *1*(1), 17–41.

Golumbic, M. C., & Lewenstein, M. (2000). New results on induced matchings. *Discrete Applied Mathematics*, *101*(1–3), 157–165. https://doi.org/10.1016/S0166-218X(99)00194-8.

Guruswami, V., & Rangan, C. P. (2000). Algorithmic aspects of clique-transversal and clique-independent sets. *Discrete Applied Mathematics*, *100*(3), 183–202. https://doi.org/10.1016/S0166-218X(99)00159-6.

Gyárfás, A., Kratsch, D., Lehel, J., & Maffray, F. (1996). Minimal non-neighborhood-perfect graphs. *Journal of Graph Theory*, *21*(1), 55–66. https://doi.org/10.1002/(SICI)1097-0118(199601)21:1<55::AID-JGT8>3.0.CO;2-L.

Habib, M., & Paul, C. (2010). A survey of the algorithmic aspects of modular decomposition. *Computer Science Review*, *4*(1), 41–59. https://doi.org/10.1016/j.cosrev.2010.01.001.

Henning, M. A., & Yeo, A. (2013). *Total domination in graphs. Springer monographs in mathematics*. New York: Springer. https://doi.org/10.1007/978-1-4614-6525-6.

Laskar, R., Pfaff, J., Hedetniemi, S. M., & Hedetniemi, S. T. (1984). On the algorithmic complexity of total domination. *SIAM Journal on Algebraic Discrete Methods*, *5*(3), 420–425. https://doi.org/10.1137/0605040.

Lehel, J. (1994). Neighbourhood-perfect line graphs. *Graphs and Combinatorics*, *10*(4), 353–361. https://doi.org/10.1007/BF02986685.

Lehel, J., & Tuza, Z. (1986). Neighborhood perfect graphs. *Discrete Mathematics*, *61*(1), 93–101. https://doi.org/10.1016/0012-365X(86)90031-2.

McConnell, R. M., & Spinrad, J. P. (1999). Modular decomposition and transitive orientation. *Discrete Mathematics*, *201*(1–3), 189–241. https://doi.org/10.1016/S0012-365X(98)00319-7.

Mitchell, S., Hedetniemi, S., & Goodman, S. (1975). 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), Utilitas Math., Winnipeg, Man., pp 467–483. Congressus Numerantium, No. XIV.

Mitchell, S. L., Cockayne, E. J., & Hedetniemi, S. T. (1979). Linear algorithms on recursive representations of trees. *Journal of Computer and System Sciences*, *18*(1), 76–85. https://doi.org/10.1016/0022-0000(79)90053-9.

Sampathkumar, E., & Neeralagi, P. S. (1985). The neighbourhood number of a graph. *Indian Journal of Pure and Applied Mathematics*, *16*(2), 126–132.

Savage, C. (1982). Depth-first search and the vertex cover problem. *Information Processing Letters*, *14*(5), 233–235. https://doi.org/10.1016/0020-0190(82)90022-9.

Seinsche, D. (1974). On a property of the class of n-colorable graphs. *The Journal of Combinatorial Theory Series B*, *16*, 191–193.

Tedder, M., Corneil, D., Habib, M., & Paul, C. (2008). Simpler linear-time modular decomposition via recursive factorizing permutations. In *Automata, languages and programming. Part I, Lecture notes in computer science* (Vol. 5125, pp. 634–645). Berlin: Springer. https://doi.org/10.1007/978-3-540-70575-8_52.

Tinhofer, G. (1988/1989) Strong tree-cographs are Birkhoff graphs. *Discrete Applied Mathematics*, *22*(3), 275–288. https://doi.org/10.1016/0166-218X(88)90100-X.

West, D. B. (2001). *Introduction to graph theory*. Upper Saddle River, NJ: Prentice Hall.

Zito, M. (2000). Linear time maximum induced matching algorithm for trees. *Nordic Journal of Computing*, *7*(1), 58–63.