Algorithmica
https://doi.org/10.1007/s00453-023-01192-2

# Complexity Issues on of Secondary Domination Number

**Joanna Raczek**[1]

Received: 30 April 2021 / Accepted: 5 November 2023


**Abstract**
In this paper we study the computational complexity issues of the problem of secondary domination (known also as (1, 2)-domination) in several graph classes. We also study the computational complexity of the problem of determining whether the domination and secondary domination numbers are equal. In particular, we study the influence of triangles and vertices of degree 1 on these numbers. Also, an optimal algorithm for finding a minimum secondary dominating set in trees is presented.

**Keywords** Computational complexity · Algorithms · Domination number · Graph classes · Applications

## 1 Introduction

In networks some resources (possibly limited in number) should be available immediately and directly. Thus some nodes can function as special nodes, for example servers, e.g. licence servers, data sets or Internet providers. Then these special nodes form a dominating set. The domination number is the number of the smallest possible number of these special nodes. The review of the literature shows that domination parameters can be used in many applications, ranging from sensors for environment, vehicular ad hoc communications, road safety, health, home, peer-to-peer messaging, disaster rescue operations, air/land/navy defence, weapons, and robots.

However in some networks a higher reliability is demanded, so in case of one server's failure, another one can take over necessary tasks. In this paper we focus on a situation when a spare special node does not need to be a direct neighbour of an ordinary node, it might be at distance 1 or 2. If the spare special node is at distance 2, then the communication might be performed at a lower speed or only the most demanding tasks may be assured by the spare server. An analogous situation takes place when the poor signal coverage makes it impossible to make standard connections via a mobile

✉ Joanna Raczek
joanna.raczek@pg.edu.pl

[1] Gdansk University of Technology, ul. Narutowicza 11/12, Gdańsk, Poland

Published online: 27 November 2023

phone network. However in most situations we can call emergency numbers. To fulfill these conditions the special nodes need to form a secondary dominating set, known also as a $(1, 2)$-dominating set.

The secondary domination can be applied in case of the covid pandemic. Suppose we want to ensure good medical care for the citizens. Let $V$ be a set of towns with at least $n$ thousand of habitants and let two elements of $V$ be adjacent if the distance between them is at most $k$ kilometers. Then the minimum $(1, 2)$-dominating set is the optimum location of medical care points such that each town has a basic medical point at distance at most $k$ and in case of overflow of this, a spare medical point at distance at most $2k$.

The secondary domination in graphs was defined in 2008 by Hedetniemi et al. [2]. The authors of this paper mention a forthcoming paper on the algorithmic complexity of this domination problem, however this paper has not been written [8]. Since then many papers on this topic have appeared (for example see [3, 5–7]). Therefore we decided to study the computational complexity of this domination problem.

The $(1, 2)$-domination in graphs is a special case of $(1, k)$-domination. In this paper we only deal with the case when $k = 2$, that is with a secondary domination.

For a graph $G$ a set $D \subseteq V(G)$ is a dominating set if each vertex $v$ of $V - D$ has a neighbour in $D$. The domination number, $\gamma(G)$, is the cardinality of a smallest dominating set of $G$.

A set $D \subseteq V(G)$ is a $(1, 2)$-dominating set if each vertex $v$ of $V - D$ has a neighbour in $D$ as well as another vertex of $D$ at a distance not greater than 2 from $v$. The $(1, 2)$-domination number, denoted by $\gamma_{1,2}(G)$, is the cardinality of a smallest $(1, 2)$-dominating set of $G$.

In what follows we assume $G$ to be a connected graph with at least two vertices. The case when $G$ is not connected comes to considering the $(1, 2)$-dominating sets in each connected component separately.

## 2 Complexity of (1, 2)-Domination Number

Similarly as in the case of other types of domination problems, the decision problem of $(1, 2)$-dominating set is also NP-complete. Let us define this decision problem as follows.

$(1, 2)$-DOMINATING SET

*Instance*: A graph $G$ and a positive integer $k$.

*Question*: Does $G$ have a $(1, 2)$-dominating set of size at most $k$?

A split graph is a graph in which the vertex set can be partitioned into a clique and an independent set.

**Theorem 1** $(1, 2)$-*DOMINATING SET is NP-complete, even for split graphs and even for bipartite graphs.*

***Proof*** (*Outline*). It is obvious that $(1, 2)$-DOMINATING SET is a member of NP, since we can, in polynomial time, guess a set $D$ and verify whether $D$ has cardinality at most $k$ and is a $(1, 2)$-dominating set.

The reduction is from EXACT COVER BY 3-SETS (X3C). We are given an instance $X = \{x_1, \ldots, x_{3q}\}$ and $C = \{C_1, \ldots, C_m\}$ of X3C, where $C_j$ are subsets of $X$ of size $|C_j| = 3$ for $1 \leq j \leq m$. Assume that $m \geq 2$, since otherwise the answer is trivial. For this instance we construct a split graph $G$ with vertices for each $x_i \in X$, for each $c_j \in C$ and with edges $x_i C_j$ for all $x_i \in C_j$ and edges so that the subgraph induced by $\{C_1, \ldots, C_m\}$ is a complete graph $K_m$. Let $k = q$.

It is not difficult to show that $C$ contains an exact cover if and only if $G$ has a $(1, 2)$-dominating set of cardinality at most $k$.

Similarly, we construct a bipartite graph in the same way, but instead of adding all the edges between vertices of $C$, we add two new vertices, $y_0$, $y_1$ and edges $y_0 y_1$, $y_0 C_j$ for all $j$. Set $k = q + 1$. □

## 3 Domination and (1, 2)-Domination Numbers in Bipartite Graphs

This section is devoted to complexity issues of the domination and the $(1, 2)$-domination numbers.

It is obvious from the definitions of the domination number and the $(1, 2)$-domination number that for any graph $G$,

$$\gamma(G) \leq \gamma_{1,2}(G).$$

**Theorem 2** *If $G$ is a graph with $\delta(G) > 1$ and without a triangle, then*

$$\gamma(G) = \gamma_{1,2}(G).$$

***Proof*** Let $G$ be a graph with $\delta(G) > 1$ and without a triangle. Suppose $\gamma(G) < \gamma_{1,2}(G)$. Then there exists a minimum dominating set of $G$ named $D$ such that $D$ is not a $(1, 2)$-dominating set. Then there exists a vertex $v \in V(G) - D$ such that $v$ is dominated by a vertex $w \in D$, but each vertex of $D - \{w\}$ is at distance at least 3 from $v$.

Since $\delta(G) > 1$, denote by $s$ a neighbour of $v$ in $N(v) - \{w\}$. Moreover, $G$ is without a triangle and each vertex of $D - \{w\}$ is at distance at least 3 from $v$, so $s$ is not dominated by $w$ nor by any other vertex of $D$. Hence $D$ is not dominating, it is a contradiction. □

An easy conclusion from this theorem is that if a graph $G$ is bipartite and without a leaf, then $\gamma(G) = \gamma_{1,2}(G)$. In the next section we show that if a bipartite graph has at least one leaf, then the problem of determining whether those two domination parameters are equal is a co-NP-hard problem. We also obtain a similar result when a graph has no leaf and only one triangle.

### 3.1 Complexity of Equality

In this section we consider the computation complexity of stating whether for a given graph $G$ there is $\gamma(G) = \gamma_{1,2}(G)$.

**Theorem 3** *It is co-NP-hard to determine if for a given graph $G$ there is $\gamma(G) = \gamma_{1,2}(G)$ even for bipartite graphs with only one leaf.*

***Proof*** We describe a reduction from 3SAT, which was proven to be *NP*-complete in [1], to the considered problem. The formula in 3SAT is given in conjunctive normal form, where each clause contains three literals. We assume that the formula contains the instance of each literal $u$ and its negation $u'$ (in the other case all clauses containing the literal $u$ are satisfied by the true assignment of $u$) and that the formula contains at least two clauses (otherwise the answer is trivial).

Given an instance $E$, the set of literals $U = \{u_1, u_2, \ldots, u_n\}$ and the set of clauses $C = \{C_1, C_2, \ldots, C_m\}$ of 3SAT, we construct a bipartite graph $G$ whose order is polynomially bounded in terms of $n$ and $m$, and such that the formula is satisfiable if and only if $\gamma(G) < \gamma_{1,2}(G)$.

For each literal $u_i$ construct a copy of the cycle $C_6$, that is $G(u_i) = (u_i, a_i^2, u_i', a_i^4, a_i^5, a_i^6)$ for $i = 1, 2, \ldots, n$. For each clause $C_j$, $j = 1, 2, \ldots, m$ there is a clause vertex $c_j$. Additionally, we add three vertices, $z_1, z_2$ and $z_3$. For every clause $C_j$ with literals $x$, $y$ and $z$, we create the three edges $c_jx$, $c_jy$ and $c_jz$. Moreover, add edges $z_1c_j$ for $j = 1, 2, \ldots, m$ and edges $z_1z_2, z_2z_3$. It is easy to verify that the obtained graph is bipartite.

Each copy of $C_6$ has four vertices of degree 2 and therefore at least two vertices of each $C_6$ belong to each $\gamma(G)$-set and to each $\gamma_{1,2}(G)$-set. Because of $z_1, z_2, z_3$, there is at least one more vertex in $\gamma(G)$-set and two more vertices in $\gamma_{1,2}(G)$-set. Therefore, $\gamma(G) \geq 2n+1$ and $\gamma_{1,2}(G) \geq 2n+2$. Since any set containing two opposite vertices from each $C_6$ together with $z_1, z_2$ is a $(1, 2)$-dominating set, regardless the satisfiability of $E$, $\gamma_{1,2}(G)$ is always equal to $2n + 2$.

Assume first that $E$ is satisfiable and consider a satisfying truth assignment. Then the set consisting of the vertices corresponding to the true literal vertices, vertices at distance 3 belonging to the same $C_6$ cycle and $z_2$ form a dominating set of size $2n+1$. Hence in this case $\gamma(G) = 2n + 1$ and therefore $\gamma(G) < \gamma_{1,2}(G)$.

If $E$ is not satisfiable, then $\gamma(G) > 2n + 1$ and therefore $\gamma(G) = \gamma_{1,2}(G)$. □

**Theorem 4** *It is co-NP-hard to determine if for a given graph $G$ is that $\gamma(G) = \gamma_{1,2}$ even for graphs with $\delta(G) > 1$ and with only one triangle.*

***Proof*** The proof is similar to the proof of Theorem 1, however we add a new vertex $x_4$ and edges $x_2x_4, x_3x_4$. The obtained graph $G$ has exactly one triangle and $\gamma(G) = \gamma_{1,2}(G)$ if and only if $E$ is satisfiable. □

### 3.2 Complexity in Graph Classes

In this part we consider computational complexity of domination and $(1, 2)$-domination problems on the same graph classes.

Let $G$ be any graph with vertex set $V = \{v_1, v_2, \ldots, v_n\}$ and $\gamma(G)$-set $D$. Add to each vertex $v_i$ of $G$ a copy of a path $P_2^i = (x_i, y_i)$ and edge $v_ix_i$, and denote the new graph by $H_1$. Then $X = \{x_i : 1 \leq i \leq n\}$ is a minimum dominating set of $H_1$. The set $Y = \{y_i : 1 \leq i \leq n\} \cup D$ is a minimum $(1, 2)$- dominating set of $H_1$. Hence

computing the minimum dominating set of $H_1$ can be done in polynomial time, but computing $\gamma_{1,2}(H_1)$ is NP-hard (because determining $\gamma(G)$ is NP-hard).

On the other hand, let $H_2$ be a graph obtained from $G$ by adding to each vertex a copy of $P_3^i = (x_i, y_i, z_i)$ and edge $v_i x_i$. Then $Y$ (defined as above) is a minimum dominating set of $H_2$, whereas $\{x_i, z_i : 1 \leq i \leq n\}$ is a minimum $(1, 2)$-dominating set of $H_2$. In this case computing $\gamma_{1,2}(H_2)$ is polynomial and computing $\gamma(H_2)$ is NP-hard.

Letting $G$ to be bipartite, both $H_1$ and $H_2$ are also bipartite. We state these considerations as a theorem.

**Theorem 5** *There is a class of bipartite graphs for which determining the domination number is NP-hard and determining the* $(1, 2)$*-domination number is polynomial and there is a class of bipartite graphs for which determining the domination number is polynomial and determining the* $(1, 2)$*-domination number is NP-hard.*

Since $\gamma_{1,2}(H_1) = \gamma(G)$, the operation of obtaining $H_1$ from $G$ is a polynomial reduction from an NP-complete problem of DOMINATING SET to the $(1, 2)$-DOMINATING SET. Therefore we state the following theorem.

**Theorem 6** $(1, 2)$*-DOMINATING SET is NP-complete for chordal bipartite graphs,* $C_4$*-free graphs, maximum degree 4 graphs, partial grid graphs and planar graphs.*

***Proof*** For all those graph classes the DOMINATING SET problem is NP-complete and is NP-complete for maximum degree 3 graphs, (see [4]). Therefore the result follows. □

## 4 Optimal Algorithm for Trees

In this section we introduce an algorithm for finding a minimum $(1, 2)$-dominating set in trees. In this algorithm each vertex $v$ has an integer status $sta(v)$. In the beginning, each status is equal to 0. At the last phase this number is used to determine the minimum $(1, 2)$-dominating set of $T$.

Let $T$ be a rooted tree and $v_0$ be the root vertex such that $v_0$ is a leaf. Let $S$ be a permutation of $V(T)$ such that if $v$ is a father of $u$, then $u$ is before $v$ in $S$. Note that the last element of $S$ is the root vertex.

If $v \in V(T)$, denote by $CL(v)$ the set of children of $v$ which are leaves, denote by $CNL(v)$ the set of children of $v$ which are not leaves and denote by $f(v)$ the father of $v$. If $v$ has only one child, denote this child by $c(v)$.

During the algorithm execution a current vertex $v$ is being assigned a code. The code $code(v)$ depends on the statuses of children of $v$ and is an sequence $ABCD$ where

- $A$ reflects the number of leaves among children of $v$. If $v$ does not have such children, then $|CL(v)| = 0$ and consequently, $A = 0$. If $v$ has exactly one leaf child, then $A = 1$. Otherwise $A = 2$.
- $B, C$ and $D$ stand for children of $v$ with status greater than 10, equal to 10, and smaller than 10, respectively. If $v$ does not have this kind of children, then $B = 0$, $C = 0$ or $D = 0$, otherwise $B = 1$, $C = 1$ or $D = 1$.

For example, if $v$ two children of $v$ have status 10 and three children have status greater than 10, then code of $v$ is 0110. Let $X$ denote any number in code, e.g. $211X$ denotes any code from among 2110 and 2111. During the execution of the algorithm the code of a vertex may change whenever the status of its child changes. Note that each leaf, except possibly for $v_0$, has the code 0000.

The algorithm is divided into three parts. The first part establishes statuses of $V(T) - \{v_0\}$. The second part establishes the status for the root vertex. The last part determines the minimum $(1, 2)$-dominating set based on the statuses of all the vertices.

**Algorithm 1:** Part I: Statuses of $V(T) - \{v_0\}$

**Data**: A rooted tree $T$ with vertex ordering $S$
**Result**: Statuses for vertices belonging to $V(T) - \{v_0\}$

Set $sta(v) = 0$ for each $v \in V(T)$;
**for** $v \in S$ **do**
  designate $code(v)$;
  **if** *$v$ is not a leaf* **nor** *a root vertex* **then**
    **if** $code(v) \in \{2X11, 1X11\}$ **then**
      Set $sta(u) = 21$ for each $u \in CNL(v)$ with $sta(u) = 10$;
    **if** $code(v) = 0X1X$ **and** $sta(v) > 10$ **then**
      Set $sta(u) = 21$ for each $u \in CNL(v)$ with $sta(u) = 10$;
    **if** $code(v) = 0X1X$ **and** $sta(v) < 10$ **then**
      Set $sta(c(u)) = 21$ for each $u \in CNL(v)$ and $sta(u) = 10$;
      Set $sta(u) = 0$ for each $u \in CNL(v)$ and $sta(u) = 10$;
    update $code(v)$;
    **if** $code(v) \in \{2101, 1101\}$ **then**
      Set $sta(v) = 22$;
    **if** $code(v) = 2001$ **then**
      Set $sta(v) = 21$;
      Set $sta(f(v)) = 22$;
    **if** $code(v) = 1001$ **and** $sta(v) < 10$ **then**
      Set $sta(v) = 10$;
    **if** $code(v) = 000X$ **and** $sta(v) < 10$ **then**
      Set $sta(f(v)) = 21$;

Before we present the algorithm that determines the status of the root vertex, we note that since the degree of $v_0$ is one, the code of $v_0$ has the form 1001 when $T$ is a tree on two vertices. Other possible codes for the root vertex are 0100, 0010 or 0001.

**Algorithm 2:** Part II: Status of the root vertex

**Data**: A rooted tree $T$ with statuses of each vertex of $V(T)$
**Result**: A status for the root vertex $v_0$

**if** $code(v_0) \in \{1001, 0010\}$ **then**
  Set $sta(v_0) = sta(c(v_0)) = 22$;
**if** $code(v_0) = 0100$ **then**
  **if** $sta(c(v_0)) = 21$ **then**
    Set $sta(v_0) = 22$;
**if** $code(v_0) = 0001$ **then**
  Set $sta(v_0) = 21$;

At this point note that after the first and second part all the vertices have statuses smaller than 10 or greater than 10. No vertex has status 10. Moreover, the algorithm distinguishes between statuses 21 and 22: a vertex has status 22 if it is in a minimum $(1, 2)$-dominating set and one of its children has status greater than 10, while a status 21 is for vertex that belongs to a minimum $(1, 2)$-dominating set but none of its children has status greater than 10. This distinction is used only by the algorithm for the root vertex.

**Algorithm 3:** Part III: A minimum $(1, 2)$-dominating set

**Data**: A rooted tree $T$ with statuses of each vertex of $V(T)$
**Result**: A minimum $(1, 2)$-dominating set of $T$
Set $D = \emptyset$;
**for** $v \in S$ **do**
  **if** $sta(v) > 10$ **then**
    Set $D = D \cup \{v\}$;

The following two examples show results of the algorithm implemented in language R using package `igraph`. See Figs. 1 and 2.

In what follows we prove that the algorithm finds the minimum $(1, 2)$-dominating set of a tree $T$. Denote by $S_i$, where $1 \leq i \leq |S|$, the subsequence of the first $i$ elements of $S$ and denote by $T_v$ the subgraph of $T$ induced by $v$ and all of its descendants.

**Theorem 7** *Let $T_v$ be a tree in the forest $T[S_i]$. Then*

1. *the vertices of status greater than 10 in $V(T_v) \cup \{f(v)\}$ $(1, 2)$-dominate vertices that are at distance 2 and more from the root $v$;*
2. *the number of vertices with status greater than 10 is not greater than $\gamma_{1,2}(T_v)$.*

***Proof*** We proceed by induction on the height of a tree $T_v$. Assume first that the height of a tree $T_v$ in the forest $T[S_i]$ is equal to 2. If there is a support vertex adjacent to at least two leaves, then both this support vertex and the root vertex have statuses greater than 10. If it is not the case, then either a leaf $x$ or the support vertex adjacent to $x$ and the root $v$ have statuses greater than 10. In both cases the number of vertices with status greater than 10 is not greater than $\gamma_{1,2}(T_v)$.

Assume now that the height $h(T_v)$ of a tree $T_v$ in the forest $T[S_i]$ is greater than 2 and the result is true for each tree $T'$ with $h(T') < h(T_v)$ and assume that vertex $v$ is processed by the Algorithm part I.

If $code(v) \in \{2X11, 1X11\}$, then each child of $v$ with status 10 gets status 21. Therefore each leaf adjacent to a vertex of status 10 has a neighbour with status greater than 10. The $code(v)$ is updated and $v$ is further proceeded.

If $code(v) \in \{2101, 1101\}$, then $v$ is a support vertex and has a child of status greater than 10. Thus, $v$ gets status 22. If $code(v)$ is 2001, both $v$ and its father get statuses greater than 10. In both cases after processing vertex $v$ by the algorithm, all vertices of $T_v$ are $(1, 2)$ dominated by vertices with status greater than 10 (together with the father of $v$). Since $v$ is a support vertex, and by the induction hypothesis, we conclude that the number of vertices of status greater than 10 in $T_v$ is not greater than $\gamma_{1,2}(T_v)$.

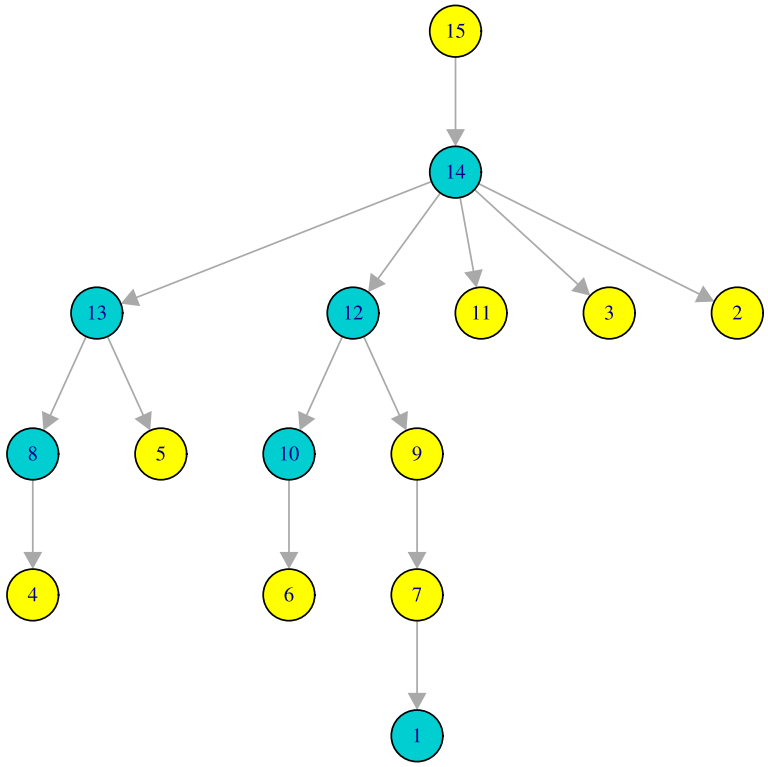


**Fig. 1** A tree. Dark vertices form a minimum (1, 2)-dominating set

If $code(v) = 2001$, then $v$ is a strong support vertex and every other child of $v$ (if exists) has status smaller than 10. For this reason both $v$ and its father get statuses greater than 10. After processing vertex $v$ by the algorithm, we obtain that the inductive step is true in this case.

If $code(v) = 1001$ and $sta(v) < 10$, then $v$ is a neighbour of exactly one leaf and every other child (if exists) has status smaller than 10. Each such child is adjacent to a vertex with status greater than 10, since otherwise $v$ would have status greater that 10. In this case $v$ gets status 10 and by the induction, the inductive step is true.

If $code(v) = 0X1X$ and $sta(v) > 10$, then $v$ got status 21 or 22 while processing one of its children. Then each child of $v$ with status 10 gets status 21 to (1, 2) dominate the leaf adjacent to it. The $code(v)$ is updated and the inductive step is true.

If $code(v) = 0X1X$ and $sta(v) < 10$, then every leaf adjacent to a vertex of status 10 gets status 21 and by the induction, the inductive step is true. The $code(v)$ is updated.

If $code(v) = 000X$ and $sta(v) < 10$, then the father of $v$ gets status 21 and again by the induction, the inductive step is true. □

**Theorem 8** *The algorithm for trees is valid and returns a minimum* (1, 2) *dominating set of a tree in time* $O(\Delta n)$.

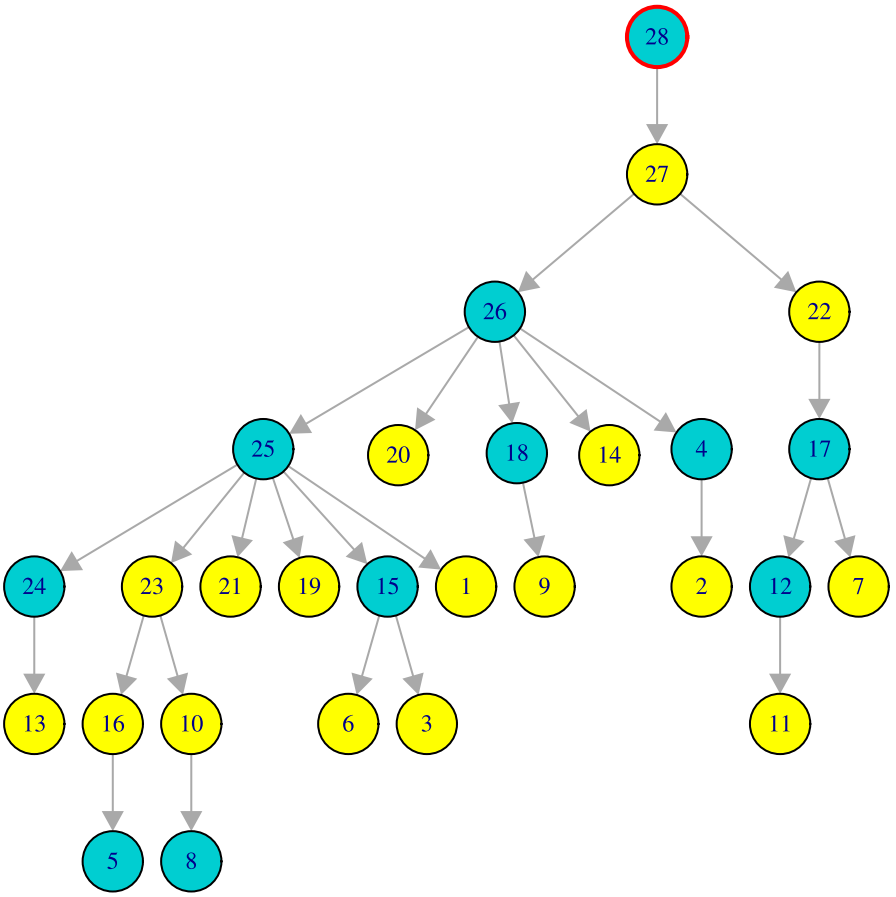


**Fig. 2** A tree. Dark vertices form a minimum (1, 2)-dominating set

***Proof*** At first we prove that after processing part II of the Algorithm, the vertices of status greater than 10 in $T$ form a $(1, 2)$-dominating set of $V(T)$ and the number of vertices with status greater than 10 is not greater than $\gamma_{1,2}(T)$.

By Theorem 7, all vertices at distance at least 3 from the root $v_0$ are $(1, 2)$ dominated by vertices of status greater than 10. Not all codes for $v_0$ are possible, however in all cases $v_0$ gets status greater than 10, except for the case when the child of $v_0$ has status 22. Therefore, after processing $v_0$ all of status greater than 10 in $T$ form a $(1, 2)$-dominating set of $V(T)$ and the number of vertices with status greater than 10 is not greater than $\gamma_{1,2}(T)$.

Part III of the Algorithm creates a set $D$, which consists of vertices having status at least 20.

To determine the status of a vertex $v$, the algorithm analyses its children and grandchildren. However, a grandchild is processed only if it is a leaf adjacent to a weak support vertex. Hence, the number of such grandchildren is smaller than $\delta(v)$ and therefore the algorithm works in time $O(\Delta n)$. □

**Funding** No funding was received to assist with the preparation of this manuscript.

## Declarations

**Conflict of interest** The author declares that she has no conflicts of interest.



## References

1. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. Freeman, San Francisco (1979)
2. Hedetniemi, S.M., Hedetniemi, S.T., Kinsley, J., Rall, D.F.: Secondary domination in graphs. AKCE J. Graph. Combin. **5**, 103–115 (2008)
3. Factor, K.A.S., Langley, L.J.: An introduction to (1, 2)-domination graphs. Congr. Numer. **199**, 33–38 (2009)
4. Information System on Graph Classes and their Inclusions. https://www.graphclasses.org/classes/problem_Domination.html
5. Kayathri, K., Vallirani, S.: (1, 2)-Domination in Graphs. In: Arumugam S., Bagga J., Beineke L., Panda B. (eds) Theoretical Computer Science and Discrete Mathematics. ICTCSDM 2016. Lecture Notes in Computer Science, vol 10398. Springer, Cham. https://doi.org/10.1007/978-3-319-64419-6_17 (2017)
6. Michalski, A., Włoch, I.: On the existence and the number of independent (1,2)-dominating sets in the $G$-join of graphs. Appl. Math. Comput. **377**, 125155 (2020). https://doi.org/10.1016/j.amc.2020.125155
7. Michalski, A., Włoch, I., Dettlaff, M., Lemańska, M.: On proper (1,2)-dominating sets in graphs. Math. Methods Appl. Sci. **45**, 7050–7057 (2022). https://doi.org/10.1002/mma.8223
8. Rall, D.F.: Personal communication, December 2000