Watching Systems with Bounded Probes

Gennaro Cordasco[1], Luisa Gargano[1] and Adele A. Rescigno[1]

[1]Dipartimento di Informatica, Università degli Studi di Salerno, Via Giovanni Paolo II, 132 - 84084 Fisciano (SA)
{gcordasco,lgargano,arescigno}@unisa.it

Abstract

Graph identification problems have been widely studied for their applications, ranging from fault diagnosis to network monitoring. A central issue in this domain is the *identifying code*, which seeks a subset of vertices whose neighborhoods uniquely identify every vertex in the graph. In this paper, we explore a generalization of this concept through *watching systems*, a more flexible identification framework that allows each vertex (watcher) to probe arbitrary subsets of its closed neighborhoods. We focus on a constrained variant called the β-*Watching System* problem, where each vertex can probe at most β subsets. This restriction models practical limitations in monitoring capacity and introduces new computational challenges. We formally define the β-Watching System problem, establish its relationship to classical identifying codes, and investigate its structural and algorithmic properties. Our main contributions include: new computational complexity for finding optimal β-watching systems; a $(2\log n + 1)$-approximation algorithm for general graphs; an exact polynomial-time algorithm for computing optimal β-watching systems on trees.

Keywords

Identifying System, Identifying Code, Watching System, Approximation Algorithms

1. Introduction

In a wide range of network applications, including fault diagnosis, network verification, and surveillance, it is essential to uniquely determine the status or location of entities using limited resources. This need motivates the study of identifying codes, which aim to select a subset of vertices that can uniquely distinguish vertices in a graph based on their proximity to the probes.

The identifying code problem is a well-established topic within the broader domain of identification problems in graphs. The objective is to select a set of nodes I in a graph G such that the closed neighborhoods of all vertices have distinct and nonempty intersections with I. This ensures that every vertex in G can be uniquely identified based on its intersection with I. Such a set I, which is both dominating and identifying, is typically referred to as a code in the literature [13].

Let $G = (V, E)$ be an undirected graph. We denote by $n = |V|$ and $m = |E|$ the number of vertices and of edges of G, respectively. For each $v \in V$, we denote by $N_G(v) = \{u \in V \mid (u, v) \in E\}$ the open neighborhood and by $N_G[v] = N_G(v) \cup \{v\}$ the closed neighborhood of v. We omit the subscript G whenever the graph is clear from the context.

Definition 1 (Identifying System)**.** *Given a finite set X and a family $\mathcal{S}$ of subsets of X, the $\mathcal{S}$-identifying set (or $\mathcal{S}$-code) of an element $x \in X$ is defined as:*

$$C_{\mathcal{S}}(x) = \{S \in \mathcal{S} \mid x \in S\}.$$

The family $\mathcal{S}$ is called an identifying system of X if all $\mathcal{S}$-codes are non-empty and distinct.

First introduced by Karpovsky, Chakrabarty, and Levitin in 1998 [13], identifying codes are Identifying Systems in which X is the set of vertices of an undirected graph and $\mathcal{S}$ is the set of all the closed neighbourhoods of the vertices of the graph.

Definition 2 (Identifying Code)**.** *An* Identifying Code *(IC) of a graph $G = (V, E)$ is a subset $\mathcal{C} \subseteq V$ such that the family $\{N[v] \mid v \in \mathcal{C}\}$ forms an identifying system of V. Equivalently, $\mathcal{C}$ must satisfy:*

ICTCS 2025: Italian Conference on Theoretical Computer Science, September 10–12, 2025, Pescara, Italy
gcordasco@unisa.it (G. Cordasco); lgargano@unisa.it (L. Gargano); arescigno@unisa.it (A. A. Rescigno)

- Domination*:* $\forall v \in V$*,* $N[v] \cap \mathcal{C} \neq \emptyset$
- Separation*:* $\forall u, v \in V$ *with* $u \neq v$*,* $N[u] \cap \mathcal{C} \neq N[v] \cap \mathcal{C}$

Identifying codes have been extensively studied in a variety of settings. Depending on the type of domination considered, the identification criteria applied, and the structural properties of the underlying graphs, several variations of the identifying code problem have been proposed and investigated under different names [7, 8, 9, 12, 14, 16, 17]. These variations enhance the applicability of the concept across a wide range of graph classes and practical scenarios.

1.1. Watching systems

Watching systems have recently been introduced in [1] as a more flexible framework with respect to identifying codes. In a watching system, a subset of neighbor vertices (watching zone) can be selected instead of the whole closed neighborhood of a vertex. It is also possible to place several watchers at the same location, with distinct watching zones.

Definition 3 (Watching System)**.** *A* watching set *in* $G = (V, E)$ *is a collection* $W = \{(z_i, Z_i)\}_{i=1}^{k}$ *where each* $z_i \in V$ *is a location and* $Z_i \subseteq N[z_i]$*.* W *is a* watching system *if* $\{Z_1, \ldots, Z_k\}$ *forms an identifying system of* V*.*

A vertex $v \in V$ is *covered* by a watcher $w = (z, Z) \in W$ if $v \in Z$; the code $C_W(v)$ of v is the set of watchers covering v. The identifying condition requires that

$$C_W(v) \neq \emptyset \text{ and } C_W(v) \neq C_W(u) \text{ for all } u \neq v.$$

We notice that an *Identifying Code* $\mathcal{C} = \{v_1, v_2, \ldots, v_\ell\}$ is a watching system with ℓ watchers $(v_i, N[v_i])$ for each $v_i \in \mathcal{C}$. However, while it is well known that only twin-free graphs admit an *Identifying Code*[1], any graph G admits a watching system. Indeed, since $v_i \in N[v_i]$, one can always consider the trivial solution $W_t = \{(v_1, \{v_1\}), \ldots, (v_n, \{v_n\})\}$.
Moreover, as noticed in [1], the size of a watching system can be dramatically smaller than that of an optimal identifying code. In particular, watching systems give much better bounds than identifying codes in graphs with a large degree, the simplest example being a star graph, where any identifying code has size $n - 1$ while a set of $\lceil \log n \rceil$ watchers located in the center vertex of the star suffice.

The first detailed analysis of watching systems was presented in [1], where fundamental properties were established and the concept of the watching number, defined as the minimum number of watchers needed to construct a watching system for a given graph, was introduced. Subsequent research has investigated various properties of watching systems, including structural characteristics and bounds on the watching number across different graph classes [2, 3, 10, 11, 15].

1.2. The β-Watching System problem

We are interested in a constrained version of watching systems, where locations cannot be overloaded: a single vertex cannot be used as the location of a watcher for an indefinite number of times.

Definition 4 (β-Watching System)**.** *For a watching system* W*, define* $\sigma_W(v) = |\{w \in W \mid v \text{ is } w\text{'s location}\}|$*. A* β-Watching System *satisfies* $\sigma_W = \max_{v \in V} \sigma_W(v) \leq \beta$*.*

Of particular interest is the case $\beta = 1$, called a *single-location watching system* (1-Watching System), where each vertex serves as a watcher location at most once. Given a 1-Watching System W, to simplify the notation, the code of any vertex v will be $C_W(v) = \{z \mid v \in Z, (z, Z) \in W\}$.

We study the following computational problem:

β-Watching System (β-WS)

[1] A vertex v is a twin of another vertex u if $N[v] = N[u]$. A graph G is called twin-free if no vertex has a twin.

Input: A graph $G = (V, E)$ and an integer k.

Question: Does G admit a β-Watching System W of size $|W| \leq k$?

The corresponding optimization problem, Min β-WS, seeks to find a β-Watching System of minimum size.

1.3. Our Contributions

This paper provides the first systematic study of the β-Watching System problem. Our main contributions are threefold:

- Strong complexity bounds for 1-Watching System: NP-hardness on bounded-degree graphs, exclusion of subexponential-time algorithms, and parameterized intractability (Section 2).
- An $(2 \log n + 1)$-approximation algorithm for general β-Watching System (Section 3).
- An exact polynomial-time algorithm for β-Watching System on trees (Section 4).

2. Lower Bounds

Auger et al. have shown that the Watching System problem is NP-hard [1]. In this section, we present stronger hardness results for the β-Watching System problem.

2.1. The A-hierarchy

Parameterized complexity is a refinement to classical complexity theory in which one takes into account not only the input size, but also other aspects of the problem given by a parameter p. A problem Q with input size n and parameter p is called *fixed parameter tractable (FPT)* with respect to parameter p, if it can be solved in time $f(p) \cdot n^c$, where f is a computable function only depending on p and c is a constant. The inherent computational difficulty of solving many parameterized problems with even small parameter values has suggested that certain parameterized problems are not fixed-parameter tractable, which has motivated the theory of fixed-parameter intractability. The *W-hierarchy* $\bigcup_{t \geq 0} W[t]$ has been introduced to characterize the inherent level of intractability for parameterized problems $FPT = W[0] \subseteq W[1] \subseteq W[2] \subseteq \cdots$. However, a large number of parameterized problems have been proved to be hard or complete for various levels in the W-hierarchy. Hence, a W[1]-hard problem is not fixed-parameter tractable (unless FPT=W[1]) and one can prove W[1]-hardness by means of a parameterized reduction from a W[1]-hard problem.

A *fpt-reduction* from a parameterized problem Q to a parameterized problem Q' is an algorithm A that transforms each instance $\langle x, p \rangle$ of Q into an instance $\langle x', g(p) \rangle$ in time $f(p)|x|^{O(1)}$, where f and g are computable functions, such that $\langle x, p \rangle \in Q$ if and only if $\langle x', g(p) \rangle \in Q'$.

Flum, Grohe, and Weyer [6] introduced the *ept-reduction*. An fpt-reduction is an ept-reduction from a parameterized problem Q to a parameterized problem Q' if it transforms an instance $\langle x, p \rangle$ of Q to an instance $\langle x', p' \rangle$ of Q' in time $2^{O(p)}|x|^{O(1)}$ with $p' = O(p + \log |x|)$ such that $\langle x, p \rangle \in Q$ if and only if $\langle x', p' \rangle \in Q'$.

According to the ept-reduction, in [5] it was introduced the A-hierarchy $\bigcup_{t \geq 1} A[t]$, where $A[1] = W[1]$ (the class of problems FPT solved in time $2^{O(p)}n^c$) and $A[t] \subseteq W[t]$. Many completeness results for the W-hierarchy can be transferred to the A-hierarchy. For example, Hitting Set and Dominating Set are $A[2]$-complete under ept-reductions [6].

2.2. An EPT-reduction

We prove that 1-WS cannot be solved in time $2^{O(k)}n^c$, unless FPT$= A[2] \subseteq W[2]$.

Theorem 1. *1-WS is $A[2]$-complete under ept-reductions.*

Proof. The proof is based on an ept-reduction from Dominating Set (DS). We recall that, given graph $G = (V, E)$ and an integer k, DS asks if there exists $D \subseteq V$ such that $|D| \leq k$ and $D \cap N(v) \neq \emptyset$ for each $v \in V \setminus D$.

Starting from an instance $\langle G, k\rangle$ of DS, the ept-reduction described below gives an instance $\langle G', k'\rangle$ of 1-WS in linear time and $k' = k + \lfloor \log_2 \Delta \rfloor + 1$, where Δ is the maximum degree of any vertex in G. Since DS is $A[2]$-complete under ept-reductions, the theorem will follow.

We describe and analyze the desired ept-reduction. Let $\langle G = (V, E), k\rangle$ be an instance of DS. Let Δ be the maximum degree of any vertex of G and let $2^{a-1} \leq \Delta < 2^a$ (i.e., $a = \lfloor \log_2 \Delta \rfloor + 1$). In the following, we assume that $\Delta \geq 3$ since when G is a path or a cycle (i.e. $\Delta \leq 2$), the dominating set of G can be immediately obtained. We construct a graph $G' = (V', E')$ with $|V'| = |V| + a + 2^a - 1$. Namely, $V' = V \cup A \cup B$ where A is a set of a independent vertices and B contains $2^a - 1$ independent vertices. The graph $G' = (V', E')$ is then obtained starting from G and connecting each vertex in the independent set A with each vertex in the independent set B, and connecting each vertex in V to each vertex in A. Formally,

$$V' = V \cup A \cup B, \qquad E' = E \cup \{(u, v) \mid u \in V, v \in A\} \cup \{(v, w) \mid v \in A, w \in B\}.$$

We show now that for any k

$$\langle G, k\rangle \text{ is a YES-instance of DS iff } \langle G', k' = k + a\rangle \text{ is a YES-instance of 1-WS.} \tag{1}$$

Assume first that D is a dominating set of G of size at most k. W.l.o.g. we assume $|D| \geq 2$. For any vertex $v \in V \setminus D$ choose any neighbor of v in D and denote it with d_v. For each vertex $u \in D$, let $L(u) = \{v \in N_G(u) \setminus D \mid u = d_v\}$. Now, we define the desired watching system for G'.
For each vertex $u \in D$, let $S_u = A \cup \{u\} \cup L(u)$.
Let $A = \{v_1, v_2, \cdots, v_a\}$ and let $B = \{w_1, w_2, \cdots, w_{2^a-1}\}$.
For each $v_i \in A$,
– let $B(v_i)$ be the subset of B consisting of all the vertices $w_j \in B$ such that the value of bit i in the binary representation of j is 1;
– for each $u \in D$, let $L(u) = \{x_1, \cdots, x_{n_u}\}$. Define $L(v_i, u)$, the subset of $L(u)$ consisting of all the vertices x_j such that the value of bit i in the binary representation of j is 1 (recall $n_u \leq \Delta < 2^a$).
We set $S_{v_i} = \{v_i\} \cup B(v_i) \cup \bigcup_{u \in D} L(v_i, u)$. We claim that $W = \{(u, S_u) \mid u \in D\} \cup \{(v_i, S_{v_i}) \mid v_i \in A\}$ is a 1-Watching System for G. First of all, notice that $|W| = |D| + |A| \leq k + a$. By construction, each vertex v in G' has code $C_W(v) \neq \emptyset$. Indeed,

$$C_W(v) = \begin{cases} \{v\} & \text{if } v \in D, \\ \{v\} \cup D & \text{if } v \in A, \\ \{v_i \in A \mid \text{the value of bit } i \text{ in the binary} \\ \qquad \text{representation of } j \text{ is } 1\} & \text{if } v = w_j \in B, \\ \{d_v\} \cup \{v_i \in A \mid \text{if } v = x_j \in L(d_v), \text{ the value of bit } i \\ \qquad \text{in the binary representation of } j \text{ is } 1\} & \text{if } v \in V \setminus D. \end{cases} \tag{2}$$

Now we prove that for each pair of vertices v, v' in G', it holds $C_W(v) \neq C_W(v')$.
– $v, v' \in B$. Let $v = w_j$ and $v' = w_{j'}$. Since the binary representation of j and j' are different, we have $C_W(v) \neq C_W(v')$.
– $v \in A$ and $v' \in B$. Since $D \subseteq C_W(v)$ and $D \cap C_W(v') = \emptyset$, we have $C_W(v) \neq C_W(v')$.
– $v, v' \in A$. We have $C_W(v) \setminus D = \{v\} \neq \{v'\} = C_W(v') \setminus D$.
– $v \in A$ and $v' \in D$. We have $v \in C_W(v)$ and $v \notin C_W(v')$.
– $v \in A$ and $v' \in V \setminus D$. Since $|D| \geq 2$, $|C_W(v) \cap D| \geq 2$ and $|C_W(v') \cap D| = |\{d_{v'}\}| = 1$.
– $v, v' \in D$. $C_W(v) = \{v\} \neq \{v'\} = C_W(v')$.
– $v \in V$ and $v' \in B$. $C_W(v) \cap D \neq \emptyset$ and $C_W(v) \cap D = \emptyset$ (recall $C_W(v') \subset A$).
– $v \in D$ and $v' \in V \setminus D$. $C_W(v') \cap A \neq \emptyset$ and $C_W(v) \cap A = \emptyset$.

– $v, v' \in V \setminus D$. If $d_v \neq d_{v'}$, then the claim holds since $d_v \in C_W(v) \setminus C_W(v')$ and $d_{v'} \in C_W(v') \setminus C_W(v)$. Otherwise, let v and v' be dominated by the same vertex $d_v \in D$ (that is, $v, v' \in L(d_v)$). Let $v = x_j$ and $v' = x_{j'}$ with $j \neq j'$, and let i be a bit in which j and j' differ. W.l.o.g., assume the value of bit i in the binary representation of j is 1. Hence, by (2) $v_i \in C_W(v) \setminus C_W(v')$, and the claim holds.

Let W be a 1-Watching System for G with $|W| \leq k + a$; that is, W locates at most $k + a$ watchers each in a different vertex. We first claim that all the a vertices in A are locations of watchers to allow that the codes $C_W(w)$, for $w \in B$, are non-empty and distinct. Assume that at most $a - x$ vertices of A, for $x \geq 1$, are locations of watchers. Let X be the subset of vertices of A that are not the location of watchers. In this case, at most $2^{a-x} - 1$ vertices in B can have non-empty and distinct codes thanks to the watchers located in A. Since the vertices in A are the only neighbors of vertices in B, and the vertices in B are independent, we have that at least $(2^a - 1) - (2^{a-x} - 1) \geq x$ (recall $\Delta \geq 3$ and then $a \geq 2$) vertices in B need to be locations of watchers.

Hence, if x watchers located in the vertices of B are substituted with x watchers located at the x vertices in X so that all the remaining vertices in B have non-empty and distinct codes, we have that all the a vertices in A are locations of watchers. This implies that the $2^a - 1$ codes $C_W(w)$ for $w \in B$ match with all the $2^a - 1$ non-empty subsets of the locations of watchers in A.
By the above, we have that at most k locations of watchers are in V and that the codes $C_W(v), C_W(v')$ of any two vertices $v, v' \in V$, must contain at least a vertex in V that is a location of a watcher. Therefore, denoted by D the set of locations in V and considering that $C_W(v) \subseteq N[v]$ for each $v \in V$, we have $C_W(v) \cap D \neq \emptyset$ for each $v \in V$. Hence, D is a dominating set of G. □

Corollary 1. *1-WS cannot be solved in time $2^{O(k)}n^c$, unless FPT= $A[2] \subseteq W[2]$.*

2.3. No subexponential algorithms for 1-WS

In this section, we prove the following result.

Theorem 2. *Unless ETH fails, the 1-WS problem does not admit an algorithm working in time $O^*(2^{o(n+m)})$.*

We present a linear reduction from 3-SAT to 1-WS.

Let ϕ be a 3-CNF boolean formula with r variables $x_1, x_2, \cdots, x_r$ and s clauses $C_1, C_2, \cdots, C_s$ where each clause is of length exactly three. We construct an instance $\langle G_\phi, k = 3r + 2s \rangle$ of 1-WS. The graph G_ϕ is constructed as follows.
– For each variable x_i, with $i \in [r]$, we introduce a variable gadget $\mathcal{X}_i$ that is obtained by first considering a complete bipartite graph in which each of the two vertices in the independent set $A_i = \{a_{i1}, a_{i2}\}$ is connected to each of the three vertices in the independent set $B_i = \{b_{i1}, b_{i2}, b_{i3}\}$ and to the vertices corresponding to x_i and $\overline{x}_i$; and finally adding an edge connecting x_i and $\overline{x}_i$. See Figure 1 on the left.
– For each clause C_q, with $q \in [s]$, we introduce a clause gadget $\mathcal{C}_q$ that is a complete bipartite graph in which each of the two independent vertices in the set $A_q = \{a_{q1}, a_{q2}\}$ is connected to each of the three independent vertices in the set $B_q = \{b_{q1}, b_{q2}, b_{q3}\}$ and to the vertex x_i (resp. $\overline{x}_i$) if x_i (resp. $\overline{x}_i$) appears in the clause C_q. See Figure 1 on the right.

It is easy to see that the construction of G_ϕ can be done in linear time.

Claim 1. *ϕ is satisfiable iff $\langle G_\phi, k = 3r + 2s \rangle$ is a* YES*-instance of 1-WS.*

Proof. Let $\{f_1, f_2, \cdots, f_r\}$ be a satisfying assignment for ϕ, where f_i corresponds to the truth value of the variable x_i. We construct a 1-Watching System W for $V(G_\phi)$ as follows:
– For each $i \in [r]$, if f_i is true, then include the pair $(x_i, N_{G_\phi}[x_i])$ in W, otherwise include $(\overline{x}_i, N_{G_\phi}[\overline{x}_i])$.
– For each $i \in [r]$, include the pairs $(a_{i1}, \{a_{i1}, b_{i1}, b_{i2}, x_i, \overline{x}_i\})$,$(a_{i2}, \{a_{i2}, b_{i2}, b_{i3}\})$ in W.
– For each $q \in [s]$, include the pairs $(a_{q1}, \{a_{q1}, b_{q1}, b_{q2}\})$, $(a_{q2}, \{a_{q2}, b_{q2}, b_{q3}\})$ in W.

It is easy to see that the codes of the vertices of G_ϕ are the following and they are all distinct:
$C_W(b_{i1}) = \{a_{i1}\}, C_W(b_{i2}) = \{a_{i1}, a_{i2}\}, C_W(b_{i3}) = \{a_{i2}\}$,

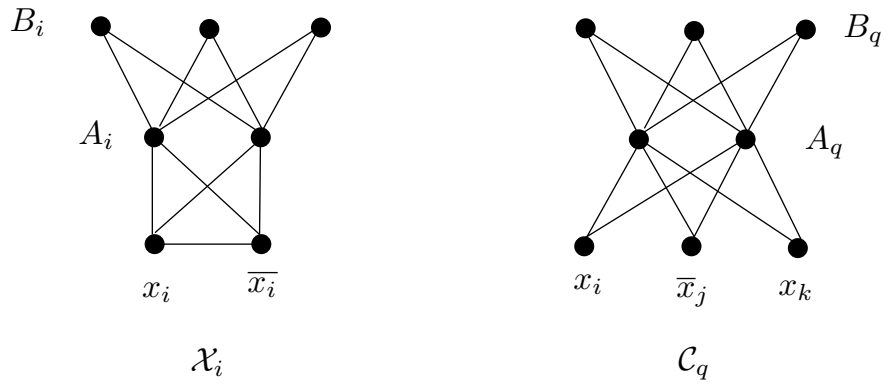

Figure 1: On the left, the variable gadget $\mathcal{X}_i$ for the variable x_i, $i \in [r]$. On the right, the clause gadget $\mathcal{C}_q$ for the clause $C_q = (x_i \vee \overline{x}_j \vee x_k)$, $q \in [s]$.

$$C_W(a_{i1}) = \begin{cases} \{a_{i1}, x_i\} & \text{if } f_i \text{ is true,} \\ \{a_{i1}, \overline{x}_i\} & \text{otherwise,} \end{cases} \qquad C_W(a_{i2}) = \begin{cases} \{a_{i2}, x_i\} & \text{if } f_i \text{ is true,} \\ \{a_{i2}, \overline{x}_i\} & \text{otherwise,} \end{cases}$$

$$C_W(x_i) = \begin{cases} \{x_i\} & \text{if } f_i \text{ is true,} \\ \{\overline{x}_i, a_{i1}\} & \text{otherwise} \end{cases} \qquad C_W(\overline{x}_i) = \begin{cases} \{\overline{x}_i\} & \text{if } f_i \text{ is false,} \\ \{x_i, a_{i1}\} & \text{otherwise,} \end{cases}$$

$C_W(b_{q1}) = \{a_{q1}\}, C_W(b_{q2}) = \{a_{q1}, a_{q2}\}, C_W(b_{q3}) = \{a_{q2}\}$,
$C_W(a_{q1}) = \{a_{q1}\} \cup \{x_i \mid x_i \in C_q \text{ and } f_i \text{ is true}\} \cup \{\overline{x}_i \mid \overline{x}_i \in C_q \text{ and } f_i \text{ is false}\}$,
$C_W(a_{q2}) = \{a_{q2}\} \cup \{x_i \mid x_i \in C_q \text{ and } f_i \text{ is true}\} \cup \{\overline{x}_i \mid \overline{x}_i \in C_q \text{ and } f_i \text{ is false}\}$.
Notice that the set $\{x_i \mid x_i \in C_q \text{ and } f_i \text{ is true}\} \cup \{\overline{x}_i \mid \overline{x}_i \in C_q \text{ and } f_i \text{ is false}\} \neq \emptyset$ since at least one of the literals in C_q is true for each $q \in [s]$.

Assume, now that W is a solution for the instance $\langle G_\phi, k = 3r + 2s \rangle$ of 1-WS.
If two watchers are used to have a different code for each vertex in B_i (resp. B_q), then they are located at the two vertices of A_i (resp. A_q). (Notice that if W is such that three or more watchers are used to cover the vertices in some B_i (resp. B_q), then we can rearrange such a solution by moving the location of two watchers to the vertices of A_i (resp. A_q) and one watcher to one vertex in $N(A_i) \setminus B_i$ (resp. $N(A_q) \setminus B_q$)). However, the two watchers in A_i (resp. the two watchers in A_q) are able to generate at most three non-empty codes - one for each vertex in B_i (resp. one for each vertex in B_q)). Hence, at least a vertex in $N(A_i) \setminus B_i$ and at least a vertex in $N(A_q) \setminus B_q$ must be a location of a watcher (to be sure that each of the vertices in A_i has a distinct code, and each of the vertices in A_q has a distinct code). Since the total number of watchers in W is $3r + 2s$, then there are r watchers located in $\left(\bigcup_{i \in [r]} N(A_i) \setminus B_i\right) \bigcup \left(\bigcup_{q \in [s]} N(A_q) \setminus B_q\right)$. If there are at least 2 watchers located in $N(A_i) \setminus B_i$, then there are at least an $i' \in [r]$ with $i' \neq i$ such that no watcher is in $N(A_{i'}) \setminus B_{i'}$, and this is not possible. Hence, there is exactly 1 watcher located in $N(A_i) \setminus B_i$ for each $i \in [r]$; moreover, among these r watchers at least one is also in $N(A_q) \setminus B_q$, for each $q \in [s]$. This means that there is at least a literal, w.l.o.g. let say x_i, in each clause C_q, locating a watcher, i.e. $x_i \in N(A_q) \setminus B_q$. Since $x_i \in N(A_i) \setminus B_i$ and at most one watcher is located in $N(A_i) \setminus B_i$, we have $\overline{x}_i$ does not locate a watcher. Hence, giving value true to the literals whose vertices locate a watcher, we have a satisfying assignment for ϕ. □

Since the reduction from 3-SAT is linear (that is, the output instance has size bounded by $O(r + s)$), if 1-WS admits an algorithm with running time $2^{o(|I|)}$, where I is the size of the input instance, then composing the reduction with such an algorithm would yield an algorithm for 3-SAT running in time $2^{o(r+s)}$, which contradicts ETH, so completing the proof of Theorem 2.

It is easy to see that the lower bound given in Theorem 2 is tight. Indeed, given an instance $\langle G, k \rangle$ of 1-WS, we could try each subset A of $V(G)$, with $|A| \leq k$, to locate watchers and then, removing all the edges in $E(G)$ that are not incident on A, so obtaining a set of edges E_A, we could try each subset B of edges in E_A to identify the subset of neighbors S_u for each vertex $u \in A$, i.e., $S_u = \{v \mid v \in N_G[u],\ (u, v) \in B\}$. Finally, we check if $W = \{(u, S_u) \mid u \in A\}$ is a watching system

of G.

2.4. Graphs of bounded degree

We prove now that 1-WS remains NP-hard even when the degree of the input graph is bounded. We establish the hardness by a reduction from the variant of 3-SAT in which each clause is of length exactly 3 and each variable occurs exactly twice unnegated and twice negated. This variant was shown to be NP-complete by Darmann and Döcker in [4].

Consider any 3-CNF boolean formula ϕ with r variables $x_1, x_2, \cdots, x_r$ and s clauses $C_1, C_2, \cdots, C_s$ such that each clause is of length exactly three, and each variable appears twice unnegated and twice negated, and use the same construction given in Theorem 3. The vertices in the sets A_i, B_i in gadget $\mathcal{X}_i$ for all $i \in [r]$ have degree at most five and the vertices in the sets A_q, B_q in gadget $\mathcal{C}_q$, for $q \in [s]$, have degree at most six. Furthermore, since each literal appears in exactly 2 clauses, it is easy to see that the maximum degree of G_ϕ is seven. Also, the construction of G_ϕ can be done in linear time.

Following the proof of Theorem 3, we can prove that ϕ is satisfiable iff $\langle G_\phi, k = 3r + 2s \rangle$ is a YES-instance of 1-WS. Hence, we have the theorem.

Theorem 3. *1-WS is* NP*-hard even on graphs with maximum degree seven.*

3. An $O(\log n)$-approximation algorithm the β-Watching System problem

Before describing the algorithm, we make the following observation: Given a watching set W for a graph G, if all vertex codes are distinct but a vertex v has the empty code ($C_W(v) = \emptyset$), we can easily construct a watching system by adding an extra watcher $w = (v, \{v\})$. For the sake of simplicity, in this section, we allow the presence of empty codes, as resolving them requires adding at most one additional watcher, thus increasing the size of the watching system by at most one.

For a graph $G = (V, E)$, let $\mathcal{W}$ the set of all the possible β-Watching System, we define a function $f : \mathcal{W} \to \mathbb{N}$, as follow:

Consider a watching set $W = \{w_1, w_2, \ldots, w_k\} \in \mathcal{W}$ where each w_i is a couple $w_i = (v_i, Z_i)$, v_i is a vertex and $Z_i \subseteq N[v_i]$. We denote by $M_v(W) = \{u \in V \mid C_W(v) = C_W(u)\}$ the set that comprises all the vertices having the same code of v according to the watching set W.

We denote by $m_v(W) = |M_v(W)|$ the cardinality of $M_v(W)$, that is, the number of times the code associated with vertex v is repeated across all vertices in the graph. Analogously, for each code $\ell \in 2^W$ we denote by $n_\ell(W) = |\{u \in V \mid C_W(u) = \ell\}|$ the number of times the code ℓ is repeated across all vertices in the graph.

For any possible set of watchers W let

$$f(W) = n^2 - \sum_{v \in V} m_v(W) \tag{3}$$

We notice that $\sum_{v \in V} m_v(W) = \sum_{\ell \in 2^W} n_\ell(W)^2$, that is, the sum of the squares of the multiplicities of each code.

Lemma 1. *The following properties of f hold:*
(i) f is integer valued;
(ii) $f(\emptyset) = 0$;
(iii) f is non-decreasing;
(iv) For every watching set $W \in \mathcal{W}$, $f(W) \leq n^2 - n$. Moreover, $f(W) = n^2 - n$ if and only if all codes are distinct.

Proof. *(i)* trivially hold by definition of f.

To prove *(ii)*, we notice that if $W = \emptyset$, then the same code (the empty one) is shared by all the n vertices and, consequently, $f(\emptyset) = n^2 - n^2 = 0$.

To prove *(iii)*, it suffices to show that for each $v \in V$ and for each watching set W and each watcher $w \notin W$, we have $M_v(W \cap \{w\}) \subseteq M_v(W)$ and consequently the function $m_v()$ is non-increasing. Indeed, each time we add a new watcher $w = (z, Z) \notin W$, we have that if $\emptyset \neq Z \cap M_v(W) \subset M_v(W)$ then $M_v(W \cup \{w\}) \subset M_v(W)$. Otherwise $M_v(W \cup \{w\}) = M_v(W)$.

Consider now *(iv)*. Let W be a watching set. Since every vertex receives a code, we have $\sum_{v \in V} m_v(W) \geq n$ and thus $n^2 - n$ is the maximum value of the function f.

The maximum value $f(W) = n^2 - n$ is achieved if and only if each vertex has a unique code—i.e., all codes are distinct. For example, the trivial watching set $W_t = \{(v_1, \{v_1\}), \ldots, (v_n, \{v_n\})\} \in \mathcal{W}$ assigns a distinct code to each vertex, achieving $f(W_t) = n^2 - n$.

Conversely, if at least one code is repeated, then there exist at least two vertices sharing the same code. In that case, $\sum_{v \in V} m_v(W) \geq (n-2) + 2 + 2$ and consequently $f(W) < n^2 - n$. □

Lemma 2. *The function f, given in (3), is submodular.*

Proof. Recall that a function $f : \mathcal{W} \to N$ is submodular if for all $W_1, W_2 \in \mathcal{W}$ with $W_1 \subseteq W_2$ and for all $w \notin W_2$, the inequality $f(W_2 \cup \{w\}) - f(W_2) \leq f(W_1 \cup \{w\}) - f(W_1)$ holds.

Recalling that the class of submodular functions is closed under non-negative linear combinations, it suffices to show that, for each $v \in V$, the function $-m_v(W)$ is submodular, that is. That is, we need to verify the inequality:

$$m_v(W_2) - m_v(W_2 \cup \{w\}) \leq m_v(W_1) - m_v(W_1 \cup \{w\}). \tag{4}$$

for all $W_1, W_2 \in \mathcal{W}$ with $W_1 \subseteq W_2$ and $w \notin W_2$.

To prove this, consider a watching set W, a watcher $w = (z, Z) \notin W$ and any vertex v. There are two cases to consider, depending on the relation between v and Z.

Assume first that $v \notin Z$. In this case, we have:

$$M_v(W \cup \{w\}) = M_v(W) \setminus Z.$$

Hence,

$$m_v(W \cup \{w\}) = m_v(W) - |Z \cap M_v(W)|,$$

or equivalently

$$m_v(W) - m_v(W \cup \{w\}) = |Z \cap M_v(W)|.$$

Using this relation on both sides of inequality (4), we obtain:

$$|Z \cap M_v(W_2)| \leq |Z \cap M_v(W_1)|.$$

Noting that $M_v(W_2) \subseteq M_v(W_1)$ (see proof of (iii) above), we have that the inequality (4) holds.

The case $v \in Z$ is proved with the same arguments, using $\overline{Z} = V \setminus Z$ in place of Z. □

Theorem 4. *MIN-β-WS can be approximated in polynomial time by a factor of* $2 \log n + 1$.

Proof. Let $\mathcal{A}$ denote the natural greedy strategy which starts with $W = \emptyset$ and iteratively adds to W the element $w \notin W$ that maximizes the marginal gain $f(W \cup \{w\}) - f(\{w\})$, until $f(W) = n^2 - n$ is achieved. By a classical result of Wolsey [18], this greedy algorithm $\mathcal{A}$ is a $(\ln(\max_w f(\{w\})) + 1)$-approximation algorithm for the Watching System problem. Noticing that using a single watcher w we can have only two codes ($\{\emptyset, \{w\}\}$). Hence,

$$f(\{w\}) = n^2 - \sum_{\ell \in 2^W} n_\ell(W)^2 = n^2 - \left(n_\emptyset(\{w\})^2 + n_{\{w\}}(\{w\})^2\right) = n^2 - \left(a^2 + (n-a)^2\right)$$

for some $1 \leq a \leq n-1$. The maximum of this function is obtained for $a = n/2$ and consequently, for each watcher w, we have

$$f(\{w\}) \leq n^2 - (n^2/4 + n^2/4) = n^2/2.$$

Hence the approximation provided by the algorithm $\mathcal{A}$ is

$$\ln(n^2/2) + 1 < \log(n^2/2) + 1 = 2\log n.$$

Additionally, since the final solution may still include a vertex associated with the empty code, we may need to add one extra watcher to obtain a watching system. This increases the approximation ratio by at most 1, yielding a final approximation factor of $2\log n + 1$. □

It is worth mentioning that, to identify the watcher w that maximizes the marginal gain at each step, the algorithm considers all the vertices $z \in V$ as the location of a new watcher w, excluding those that have already been used β times. For each candidate location z, the maximal marginal gain can be computed in polynomial time based on the following observations:

- Assigning the new code $\ell' = \ell \cup \{w\}$ to a vertex having code ℓ does not affect the value of the function f for all the vertices having a different code. Therefore, the vertices can be grouped by their current codes, and each group can be analyzed independently.
- Within a group of vertices sharing the same code ℓ, updating any subset of them to receive the new code $\ell' = \ell \cup \{w\}$, has the same effect on f, regardless of which specific vertices are chosen. As a result, it is sufficient to determine, for each group, how many vertices should be updated, rather than which ones.

Hence, once the location z of the watcher is fixed, the optimal subset $Z \subseteq N[z]$ that maximizes the marginal gain can be computed as follows. For each code ℓ appearing among the vertices in $N[Z]$, define:

- $c = |\{u \in N[Z] \mid C_W(u) = \ell\}|$: the number of vertices in $N[Z]$ with code ℓ,
- $C = |\{u \in V \mid C_W(u) = \ell\}|$: the total number of vertices in the graph with code ℓ.

The objective is to choose a value $1 \leq x \leq c$, representing how many vertices in the group should have their code updated from ℓ to $\ell' = \ell \cup \{w\}$, in a way that maximizes the increase in f. Since maximizing the increase in f corresponds to minimizing the function $g(x) = x^2 + (C - x)$, which is convex and minimized at $x = C/2$, the optimal number of updates is $x = \min\{c, \lfloor C/2 \rfloor\}$.

Thus, for a fixed location z, an optimal subset $Z \subseteq N[z]$ can be computed in $O(|N[z]|)$ time. Since this procedure is repeated for each candidate vertex z, the overall time complexity for identifying the locally optimal watcher w at each step is $O(m)$.

4. An Algorithm for Trees

In this section, given a watching set W, with a slight abuse of notation, we refer to a vertex v as a watcher if v is the location of at least one watcher in W.

We give a dynamic programming algorithm, which, exploiting the structure of the input graph (a tree), enables us to solve the MIN-β-WS problem.

Before proceeding with the description of the algorithm, the following observation should be noted.

Observation 1. *Given a graph $T = (V, E)$ and a watching set W. If a node v is covered by at least two watchers in W, then its code $C_W(v)$ is unique.*

Given a tree $T = (V, E)$, we root it at an arbitrary vertex r, and define $T(v)$ as the subtree of vertex v, for any $v \in V$. We denote by $V(T(v))$ the vertex set of $T(v)$. Moreover, for each $v \neq r$, we denote by $p(v)$ the parent of v and by $Ch(v)$ the set of children of v in T.

The algorithm processes the vertices in a postorder manner, so that all children's vertices are always visited before their parent.

Fix a vertex $v \in V$ and consider the tree $T(v)$. In order to be able to recursively reconstruct the solution, the algorithm calculates partial solutions W_v associated with $T(v)$, under different hypotheses, based on the following considerations.

The vertex v can be either a watcher (cases a, b, c) or not (cases d, e, f). If v is a watcher, then it is necessary to distinguish three cases: (a) The watching set W_v, associated with $T(v)$, is a watching system for the nodes in $V(T(v))$ plus the parent node $p(v)$ (that is, it enables to identify all the nodes in $V(T(v)) \cup \{p(v)\}$); (b) The watching set W_v, associated with $T(v)$, is a watching system for the nodes in $V(T(v))$; (c) The watching set W_v, associated with $T(v)$, is a watching system for the nodes in $V(T(v)) \setminus \{v\}$.

In the case v is not a watcher, it is necessary to distinguish three additional cases: (d) The watching set W_v, associated with $T(v)$, is a watching system for the nodes in $V(T(v))$; (e) The watching set W_v, associated with $T(v)$, is a watching system for the nodes in $V(T(v)) \setminus \{v\}$ and at least one child of v is a watcher; (f) The watching set W_v, associated with $T(v)$, is a watching system for the nodes in $V(T(v)) \setminus \{v\}$.

Hence, the following cases are considered:

Case: v is a watcher: We consider three sub-cases:

Case a: The watching set W_v, associated with $T(v)$, satisfies the inequality

- $\log(c'(v) + 3) \leq \sigma_W(v) \leq \beta$, if no children of v is a watcher
- $\log(c'(v) + 2) \leq \sigma_W(v) \leq \beta$, if at least one children of v is a watcher

where $c'(v)$ denotes the number of children of v that have only a single watcher in their neighborhood—namely, v itself.

According to Observation 1, only the children that have a single watcher require distinct codes, provided by v, since no other watcher helps distinguish them. The additional 3 codes account for the following: the empty code is not permitted, one code must be assigned to v itself (this is not needed if at least one child of v is a watcher), and one code may be reserved for the parent $p(v)$, in case it is needed. That is, the watching set W_v, associated with $T(v)$, is a watching system for the nodes in $V(T(v))$ plus the parent node $p(v)$.

Case b: The watching set W_v, associated with $T(v)$, satisfies the inequality

- $\log(c'(v) + 2) \leq \sigma_W(v) \leq \beta$, if no children of v are watcher
- $\log(c'(v) + 1) \leq \sigma_W(v) \leq \beta$, if at least one children of v is a watcher

meaning it is a watching system for the nodes in $V(T(v))$.

Case c: The watching set W_v, associated with $T(v)$, satisfies the inequality

- $\log(c'(v) + 1) \leq \sigma_W(v) \leq \beta$,

meaning it is a watching system for the nodes in $V(T(v)) \setminus \{v\}$.

We notice that if at least one child of v is a watcher, cases b and c coincide.

Case v is not a watcher: We consider three sub-cases:

Case d: The watching set W_v, associated with $T(v)$, is a watching system for the nodes in $V(T(v))$. This occurs when v has at least two watchers among its children, or one child can provide an additional distinct code (see Case a).

Case e: The watching set W_v, associated with $T(v)$, is not a watching system in $V(T(v))$, and there exists a child $u \in Ch(v)$ such that u is a watcher (i.e., v is covered by one watcher).

Case f: The watching set W_v, associated with $T(v)$, is not a watching system in $V(T(v))$.

We notice that for cases a, b, d we have that all the vertices in $T(v)$ have a distinct code, and consequently, they enable us to identify a β-Watching System. The other cases are auxiliary and will be used to reconstruct the solutions related to the tree, rooted in v, starting from the solutions of the trees rooted in each of its children.

Definition 5. *Let $T = (V, E)$ be a tree of n vertices. For each $v \in V$, we denote by $\mathcal{S}[v, x]$ the size of an optimal partial solution W_v^*, such that all the vertices in $V(T(v)) \setminus \{v\}$ have a distinct code and case $x \in \{a, b, c, d, e, f\}$ occurs.*

According to Definition 5, the size of an optimal β-Watching System for T is

$$\min\{\mathcal{S}[r, a], \mathcal{S}[r, b], \mathcal{S}[r, d]\}.$$

We compute all the values $\mathcal{S}[v, x]$, for $x \in \{a, b, c, d, e, f\}$ and for each $v \in V$, using a bottom up approach. In particular, for each leaf ℓ we have,

$$\mathcal{S}[\ell, a] = \begin{cases} 2 & \text{if } \beta \geq 2 \\ \infty & \text{Othewise,} \end{cases}$$
$$\mathcal{S}[\ell, b] = \mathcal{S}[\ell, c] = 1,$$
$$\mathcal{S}[\ell, d] = \mathcal{S}[\ell, e] = \infty,$$
$$\mathcal{S}[\ell, f] = 0.$$

For each internal vertex v, we consider first the cases where v is a watcher.

Let δ_v the number of children of v, we denote u_i the i^{th} children of v. We are going to build another auxiliary table to compute the optimal solution of internal nodes.

Definition 6. *Let $T = (V, E)$ be a tree of n vertices. For each $v \in V$, and $1 \leq q \leq p \leq \delta_v$, we denote by*

- *$F[v, p, q, 0]$ the size of an optimal partial solution W for the forest $T(u_1), T(u_2), \ldots, T(u_p)$ such that at most q vertices among $u_1, u_2, \ldots, u_p$ are not covered by any watcher while none of them is a watcher.*
- *$F[v, p, q, \geq 1]$ the size of an optimal partial solution W for the forest $T(u_1), T(u_2), \ldots, T(u_p)$ such that at most q vertices among $u_1, u_2, \ldots, u_p$ are not covered by any watcher and there is at least one watcher among the vertices $u_1, u_2, \ldots, u_p$.*

We denote by $F[v, p, q] = \min\{F[v, p, q, 0], F[v, p, q, \geq 1]\}$ the size of an optimal partial solution W for the forest $T(u_1), T(u_2), \ldots, T(u_p)$ such that at most q vertices among $u_1, u_2, \ldots, u_p$ are not covered by any watcher.

We have

$$F[v, 1, i, 0] = \begin{cases} \min_{x \in \{d,e\}}\{\mathcal{S}[u_1, x]\} & \text{if } i = 0 \\ \min_{x \in \{d,e,f\}}\{\mathcal{S}[u_1, x]\} & \text{if } i = 1 \end{cases}$$

For all $1 < j \leq \delta_v$,

$$F[v, j, i, 0] = \begin{cases} F[v, j-1, i-1, 0] + \min_{x \in \{d,e,f\}}\{\mathcal{S}[u_j, x]\} & \text{if } i = j \\ \min \begin{cases} F[v, j-1, i, 0] + \min_{x \in \{d,e\}}\{\mathcal{S}[u_j, x]\} \\ F[v, j-1, i-1, 0] + \mathcal{S}[u_j, f] \end{cases} & \text{otherwise (i.e., } i < j) \end{cases}$$

For the table $F[\cdot, \cdot, \cdot, \geq 1]$, we have

$$F[v, 1, i, \geq 1] = \min_{x \in \{a,b,c\}}\{\mathcal{S}[u_1, x]\} \qquad \text{for } i = 0, 1.$$

For all $1 < j \leq \delta_v$,

$$F[v,j,i,{\geq}1] = \begin{cases} \min \begin{cases} F[v,j-1,i-1] + \min_{x\in\{a,b,c\}}\{\mathcal{S}[u_j,x]\} \\ F[v,j-1,i-1,{\geq}1] + \min_{x\in\{d,e,f\}}\{\mathcal{S}[u_j,x]\} \end{cases} & \text{if } i = j \\ \min \begin{cases} F[v,j-1,i] + \min_{x\in\{a,b,c\}}\{\mathcal{S}[u_j,x]\} \\ F[v,j-1,i,{\geq}1] + \min_{x\in\{d,e\}}\{\mathcal{S}[u_j,x]\} \\ F[v,j-1,i-1,{\geq}1] + \mathcal{S}[u_j,f] \end{cases} & \text{otherwise (i.e., } i < j) \end{cases}$$

Exploiting the table F, we are able to compute the values for $\mathcal{S}$. We have,

$$\mathcal{S}[v,a] = \min \begin{cases} \min_{0\leq i\leq z}\{F[v,\delta_v,i,0] + \lceil\log(i+3)\rceil\},\ \text{where } z \leq \delta_v \text{ and } \lceil\log(z+3)\rceil \leq \beta \\ \min_{0\leq i\leq z}\{F[v,\delta_v,i,{\geq}1] + \lceil\log(i+2)\rceil\},\ \text{where } z \leq \delta_v \text{ and } \lceil\log(z+2)\rceil \leq \beta \end{cases} \tag{5}$$

$$\mathcal{S}[v,b] = \min \begin{cases} \min_{0\leq i\leq z}\{F[v,\delta_v,i,0] + \lceil\log(i+2)\rceil\},\ \text{where } z \leq \delta_v \text{ and } \lceil\log(z+2)\rceil \leq \beta \\ \min_{0\leq i\leq z}\{F[v,\delta_v,i,{\geq}1] + \lceil\log(i+1)\rceil\},\ \text{where } z \leq \delta_v \text{ and } \lceil\log(z+1)\rceil \leq \beta \end{cases} \tag{6}$$

$$\mathcal{S}[v,c] = \min_{0\leq i\leq z}\{F[v,\delta_v,i] + \lceil\log(i+1)\rceil\},\ \text{where } z \leq \delta_v \text{ and } \lceil\log(z+1)\rceil \leq \beta \tag{7}$$

We notice that $\mathcal{S}[v,a] \geq \mathcal{S}[v,b] \geq \mathcal{S}[v,c]$. From now on, we consider cases where v is not a watcher. We have,

$$\mathcal{S}[v,d] = \min \begin{cases} \min_{w_1,w_2\in Ch(v)}\{\mathcal{S}[w_1,b] + \mathcal{S}[w_2,b] + \sum_{u\in Ch(v)\setminus\{w_1,w_2\}} \min_{x\in\{b,d\}} \mathcal{S}[u,x]\} \\ \min_{w\in Ch(v)}\{\mathcal{S}[w,a] + \sum_{u\in Ch(v)\setminus\{w\}} \min_{x\in\{b,d\}} \mathcal{S}[u,x]\} \end{cases} \tag{8}$$

$$\mathcal{S}[v,e] = \min_{w\in Ch(v)} \left\{ \mathcal{S}[w,b] + \sum_{u\in Ch(v)\setminus\{w\}} \mathcal{S}[u,d] \right\} \tag{9}$$

$$\mathcal{S}[v,f] = \sum_{u\in Ch(v)} \mathcal{S}[u,d]. \tag{10}$$

Theorem 5. *Let $T = (V,E)$ be a tree of n vertices. We can compute a solution for the instance $\langle T,\beta\rangle$ of the MIN-β-WS problem in time $O(n\Delta^3)$, where Δ is the maximum degree of any vertex in T.*

Proof. By induction on the tree, we can prove that the recursive formula presented in (5)–(10) coincides with the definition of $\mathcal{S}[\cdot,\cdot]$; hence, the algorithm described above is correct.

For each $v \in V$ the table $F[v,\cdot,\cdot,\cdot]$ comprises $O(\Delta^2)$ values, which can be computed recursively in time $O(\Delta^2)$. Given the table $F[v,\cdot,\cdot,\cdot]$, the computation of $\mathcal{S}[v,\cdot]$ comprises $O(1)$ values, which can be computed in time $O(\Delta)$. Hence, the optimal value, which corresponds to

$$\min\{\mathcal{S}[r,a],\mathcal{S}[r,b],\mathcal{S}[r,d]\}$$

can be computed within $O(n\Delta^3)$ time. An optimal watching system W^* can be computed within the same time by a standard backtracking technique. □

Declaration on Generative AI

The authors have not employed any Generative AI tools.

References

[1] D. Auger, I. Charon, O. Hudry and A. Lobstein. Watching systems in graphs: An extension of identifying codes. Discrete Applied Mathematics, vol. 161(12), 1674–1685, (2013).

[2] D. Auger, I. Charon. Watching systems in the king grid. Graphs Combin. 29, 333–347, (2012).

[3] D. Auger, I. Charon, O. Hudry, and A. Lobstein. Maximum size of a minimum watching system and the graphs achieving the bound. Discrete Appl. Math. 164, 20–33, (2014).

[4] A. Darmann and J. Döcker. On simplified np-complete variants of monotone 3-sat. Discret. Appl. Math., 292, pp. 45–58, (2021).

[5] J. Flum and M. Grohe. Parameterized Complexity theory. Texts in Theoretical Computer Science. An EATCS Series. Springer-Verlag, Berlin, (2006).

[6] J. Flum, M. Grohe and M. Weyer. Bounded fixed-parameter tractability and $\log_2 n$ nondeterministic bits. Proc. 31st International Colloquium on Automata, Languages, and Programming (ICALP 2004), LNCS 31425, 55–567, (2004).

[7] F. Foucaud. Combinatorial and algorithmic aspects of identifying codes in graphs. Thesis, Université Sciences et Technologies - Bordeaux, (2012).

[8] F. Foucaud, M.A. Henning, C. Löwenstein, T. Sasse. Locating-dominating sets in twin-free graphs. Discrete Applied Mathematics, 200, 52–58, (2016). DOI:10.1016/j.dam.2015.06.019

[9] F. Foucaud, G. Perarnau. Bounds for identifying codes in terms of degree parameters. Discrete Applied Mathematics, 232, 99–114, (2017). DOI:10.1016/j.dam.2017.07.033

[10] M. Ghorbani, M. Dehmer, H. Maimani, S. Maddah, M. Roozbayani, F. Emmert-Streib. The watching system as a generalization of identifying code. Applied Mathematics and Computation, 380, (2020).

[11] M. Ghorbani, S. Maddah. On the watching number of graphs using discharging procedure. J. Appl. Math. Comput. 67, 507–518, (2021). https://doi.org/10.1007/s12190-020-01482-wS.

[12] T. W. Haynes, M. A. Henning, J. Howard. Locating and total-dominating sets in trees. Discrete Applied Mathematics, 154, 1293–1300, (2006).

[13] M.G. Karpovsky, K. Chakrabarty and L.B. Levitin. On a new class of codes for identifying vertices in graphs. IEEE T. Inform. Theory 44, 599–611, (1998).

[14] A. Lobstein. Watching systems, identifying, locating-dominating and discriminating codes in graphs: A bibliography. arXiv preprint arXiv:1004.2192, (2010).

[15] M. Roozbayani and H. R. Maimani. Identifying codes and watching systems in Kneser graphs. Discrete Mathematics, Algorithms and Applications 09(1), (2017).

[16] S. J. Seo, P. J. Slater. Open neighborhood locating dominating sets. Australasian Journal of Combinatorics, 46, 109–119, (2010).

[17] P. J. Slater. Dominating and reference sets in a graph. Journal of Mathematical and Physical Sciences, 22, 445–455, (1988).

[18] L. A. Wolsey. An analysis of the greedy algorithm for the submodular set covering problem. Combinatorica 2, 385–393, (1982). https://doi.org/10.1007/BF02579435