Metric Dimension Parameterized by Treewidth in Chordal Graphs

Nicolas Bousquet, Quentin Deschamps(✉), and Aline Parreau

Univ. Lyon, Université Lyon 1, CNRS, LIRIS UMR 5205, 69621 Lyon, France
{nicolas.bousquet,quentin.deschamps,aline.parreau}@univ-lyon1.fr

Abstract. The metric dimension has been introduced independently by Harary, Melter [11] and Slater [15] in 1975 to identify vertices of a graph G using its distances to a subset of vertices of G. A *resolving set* X of a graph G is a subset of vertices such that, for every pair (u, v) of vertices of G, there is a vertex x in X such that the distance between x and u and the distance between x and v are distinct. The metric dimension of the graph is the minimum size of a resolving set. Computing the metric dimension of a graph is NP-hard even on split graphs and interval graphs. Bonnet and Purohit [2] proved that the metric dimension problem is W[1]-hard parameterized by treewidth. Li and Pilipczuk strengthened this result by showing that it is NP-hard for graphs of treewidth 24 in [14]. In this article, we prove that metric dimension is FPT parameterized by treewidth in chordal graphs.

1 Introduction

Determining the position of an agent on a network is a central problem. One way to determine its position is to place sensors on nodes of the network and the agents try to determine their positions using their positions with respect to these sensors. More formally, assume that agents know the topology of the graph. Can they, by simply looking at their position with respect to the sensors determine for sure their position in the network? Conversely, where do sensors have to be placed to ensure that any agent at any possible position can easily determine for sure its position? These questions received a considerable attention in the last decades and have been studied in combinatorics under different names such as metric dimension, identifying codes, locating dominating sets...

Let $G = (V, E)$ be a graph and s, u, v be three vertices of G. We say that s *resolves* the pair (u, v) if the distance between s and u is different from the distance between s and v. A *resolving set* of a graph $G = (V, E)$ is a subset S of vertices of G such that any vertex of G is identified by its distances to the vertices of the resolving set. In other words, S is a resolving set if for every pair (u, v) of vertices of G, there is a vertex s of S such that s resolves (u, v). The *metric dimension* of G, denoted by $\dim(G)$, is the smallest size of a resolving set of G.

This notion has been introduced in 1975 by Slater [15] for trees and by Harary and Melter [11] for graphs to simulate the moves of a sonar. The associated

D. Paulusma and B. Ries (Eds.): WG 2023, LNCS 14093, pp. 130–142, 2023.
https://doi.org/10.1007/978-3-031-43380-1_10

decision problem, called the Metric Dimension problem, is defined as follows: given a graph G and an integer k, is the metric dimension of G is at most k?

The Metric Dimension problem is NP-complete [9] even for restricted classes of graphs like planar graphs [4]. Epstein et al. [6] proved that this problem is NP-complete on split graphs, bipartite and co-bipartite graphs. The problem also is NP-complete on interval graphs [8] or sub-cubic graphs [12]. On the positive side, computing the metric dimension is linear on trees [11,15] and polynomial in outer-planar graphs [4].

Parameterized Algorithms. In this paper, we consider the Metric Dimension problem from a parameterized point of view. We say a problem Π is *fixed parameter tractable* (FPT) for a parameter k if any instance of size n and parameter k can be decided in time $f(k) \cdot n^{O(1)}$. Two types of parameters received a considerable attention in the literature: the size of the solution and the "width" of the graph (for various widths, the most classical being the treewidth).

Hartung and Nichterlein proved in [12] that the Metric Dimension problem is W[2]-hard parameterized by the size of the solution. Foucaud et al. proved that it is FPT parameterized by the size of the solution in interval graphs in [8]. This result was extended by Belmonte et al. who proved in [1] that Metric Dimension is FPT parameterized by the size of the solution plus the tree-length of the graph. In particular, it implies that computing the metric dimension for chordal graph is FPT parameterized by the size of the solution.

Metric Dimension is FPT parameterized by the modular width [1]. Using Courcelle's theorem, one can also remark that it is FPT parameterized by the treedepth of the graph as observed in [10]. Metric dimension has been proven W[1]-hard parameterized by the treewidth by Bonnet and Purohit in [2]. Li and Pilipczuk strengthened this result by showing that it is NP-complete for graphs of treewidth, and even pathwidth, 24 in [14]. While Metric dimension is polynomial on graphs of treewidth 1 (forests), its complexity is unknown for graphs of treewidth 2 is open (even if it is known to be polynomial for outerplanar graphs). Our main result is the following:

Theorem 1. Metric Dimension *is FPT parameterized by treewidth on chordal graphs. That is,* Metric Dimension *can be decided in time* $O(n^3 + n^2 \cdot f(\omega))$ *on chordal graphs of clique number* ω.

Recall that, on chordal graphs, the treewidth is equal to the size of a maximum clique minus one. Our proof is based on a dynamic programming algorithm. One of the main difficulty to compute the metric dimension is that a pair of vertices might be resolved by a vertex far from them in the graph. This non-locality implies that it is not simple to use classical algorithmic strategies like divide-and-conquer, induction or dynamic programming since a single edge or vertex modification somewhere in the graph might change the whole solution[1].

[1] The addition of a single edge in a graph might modify the metric dimension by $\Omega(n)$, see e.g. [7].

The first ingredient of our algorithm consists in proving that, given a chordal graph, if we are using a clique tree of a desirable form and make some simple assumptions on the shape of an optimal solution, we can ensure that resolving a pair of vertices close to a separator implies that we resolve all the pairs of vertices in the graph. Using this lemma, we build a dynamic programming algorithm that computes the minimum size of a resolving set containing a given vertex in FPT-time parameterized by treewdith.

The special type of clique tree used in the paper, inspired from [13], is presented in Sect. 2.1. We then give some properties of resolving sets in chordal graphs in Sect. 2.2. These properties will be needed to prove the correctness and the running time of the algorithm. Then, we present the definition of the extended problem in Sect. 3.1 and the rules of the dynamic programming in Sect. 3.2 where we also prove the correction of the algorithm. We end by an analysis of the complexity of the algorithm in Sect. 4.

Further Work. The function of the treewidth in our algorithm is probably not optimal and we did not try to optimize it to keep the algorithm as simple as possible. A first natural question is the existence of an algorithm running in time $2^{\omega} \cdot Poly(n)$ for chordal graphs.

We know that Theorem 1 cannot be extended to bounded treewidth graphs since Metric Dimension is NP-hard on graphs of treewidth at most 24 [14]. One can nevertheless wonder if our proof technique can be adapted to design polynomial time algorithms for graphs of treewidth at most 2 on which the complexity status of Metric Dimension is still open.

Our proof crucially relies on the fact that a separator X of a chordal graph is a clique and then the way a vertex in a component of $G \setminus X$ interacting with vertices in another component of $G \setminus X$ is simple. One can wonder if there is a tree decomposition in G where all the bags have diameter at most C, is it true that Metric Dimension is FPT parameterized by the size of the bags plus C. Note that, since Metric Dimension is NP-complete on chordal graphs, the problem is indeed hard parameterized by the diameter of the bags only.

2 Preliminaries

2.1 Nice Clique Trees

Unless otherwise stated, all graphs considered in this paper are undirected, simple, finite and connected. For standard terminology and notations on graphs, we refer the reader to [3]. Let us first define some notations we use throughout the article.

Let $G = (V, E)$ be a graph where V is the set of vertices of G and E the set of edges; we let $n = |V|$. For two vertices x and y in G, we denote by $d(x, y)$ the length of a shortest path between x and y and call it *distance between x and y*. For every $x \in V$ and $U \subseteq V$, the *distance between x and U*, denoted by $d(x, U)$, is the minimum distance between x and a vertex of U. Two vertices x and y are *adjacent* if $xy \in E$. A *clique* is a graph where all the pairs of vertices

are adjacent. We denote by ω the size of a maximum clique. Let U be a set of vertices of G. We denote by $G \setminus U$ the subgraph of G induced by the set of vertices $V \setminus U$. We say that U is a *separator* of G if $G \setminus U$ is not connected. If two vertices x and y of $V \setminus U$ belong to two different connected components in $G \setminus U$, we say that U *separates* x and y. If a separator U induces a clique, we say that U is a *clique separator* of G.

Definition 1. *A* tree-decomposition *of a graph G is a pair (X, T) where T is a tree and $X = \{X_i | i \in V(T)\}$ is a collection of subsets (called bags) of $V(G)$ such that:*

- $\bigcup_{i \in V(T)} X_i = V(G)$.
- *For each edge $xy \in E(G), x, y \in X_i$ for some $i \in V(T)$.*
- *For each $x \in V(G)$, the set $\{i | x \in X_i\}$ induces a connected sub-tree of T.*

Let G be a graph and (X, T) a tree-decomposition of G. The *width* of the tree-decomposition (X, T) is the biggest size of a bag minus one. The *treewidth* of G is the smallest width of (X, T) amongst all the tree-decompositions (X, T) of G.

Chordal graphs are graphs with no induced cycle of length at least 4. A characterization given by Dirac in [5] ensures chordal graphs are graphs where minimal vertex separators are cliques. Chordal graphs admit tree-decompositions such that all the bags are cliques. We call such a tree-decomposition a *clique tree*.

Our dynamic programming algorithm is performed in a bottom-up way on a clique tree of the graph with more properties than the one given by Definition 1. These properties permit to simplify the analysis of the algorithm. We adapt the decomposition of [13, Lemma 13.1.2] to get this tree-decomposition.

Lemma 2. *Let $G = (V, E)$ be a chordal graph and r a vertex of G. There exists a clique tree (X, T) such that (i) T contains at most $7n$ nodes, (ii) T is rooted in a node that contains only the vertex r, (iii) T contains only four types of nodes, that are:*

- *Leaf nodes, $|X_i| = 1$ which have no child.*
- *Introduce nodes i which have exactly one child j, and that child satisfies $X_i = X_j \cup \{v\}$ for some vertex $v \in V(G) \setminus X_j$.*
- *Forget nodes i which have exactly one child j, and that child satisfies $X_i = X_j \setminus \{v\}$ for some vertex $v \in X_j$.*
- *Join node i which have exactly two children i_1 and i_2, and these children satisfy $X_i = X_{i_1} = X_{i_2}$.*

Moreover, such a clique tree can be found in linear time.

In the following, a clique tree with the properties of Lemma 2 will be called a *nice clique tree* and we will only consider nice clique trees (X, T) of chordal graphs G.

Given a rooted clique tree (T, X) of G, for any node i of T, we define the *subgraph of G rooted in X_i*, denoted by $T(X_i)$, as the subgraph induced by the subset of vertices of G contained in at least one of the bags of the sub-tree of T rooted in i (i.e. in the bag of i or one of its descendants).

2.2 Clique Separators and Resolving Sets

In this section, we give some technical lemmas that will permit to bound by $f(\omega)$ the amount of information we have to remember in the dynamic programming algorithm.

Lemma 3. *Let K be a clique separator of G and G_1 be a connected component of $G \setminus K$. Let G_{ext} be the subgraph of G induced by the vertices of $G_1 \cup K$ and $G_{int} = G \setminus G_{ext}$. Let $x_1, x_2 \in V(G_{int})$ be such that $|d(x_1, K) - d(x_2, K)| \geq 2$. Then, every vertex $s \in V(G_{ext})$ resolves the pair (x_1, x_2).*

Before proving Lemma 5, let us state a technical lemma.

Lemma 4. *Let G be a chordal and T be a nice clique tree of G. Let X, Y be two bags of T such that $X \cap Y = \emptyset$. Assume that there exist $x \in X, y \in Y$ such that $d(x, y) \geq 2$ and let z be a neighbour of x that appears in the bag the closest to Y in T amongst all the bags on the path between X and Y. Then z belongs to a shortest path between x and y.*

Lemma 5. *Let S be a subset of vertices of a chordal graph G. Let X, Y and Z be three bags of a nice tree-decomposition T of G such that Z is on the path P between X and Y in T. Denote by $P = X_1, \dots Z \dots X_p$ the bags of P with $X = X_1$ and $Y = X_p$. Let x be a vertex of X and y a vertex of Y with $d(x, Z) \geq 2$ and $d(y, Z) \geq 2$. Assume that any pair of vertices (u, v) with $u \in X_2 \cup \dots \cup Z$, $v \in Z \cup \dots \cup X_p$, $d(u, Z) < d(x, Z)$ and $d(v, Z) < d(y, Z)$ is resolved by S. Then the pair (x, y) is resolved by S.*

Proof. Let i_1 be such that $X_{i_1} \cap N[x] \neq \emptyset$ and for every $j > i_1$, $X_j \cap N[x] = \emptyset$ and i_2 be such that $X_{i_2} \cap N[y] \neq \emptyset$ and for $j < i_2$, $X_j \cap N[y] = \emptyset$. Let x' be the only neighbour of x in X_{i_1} and y' be the only neighbour of y in X_{i_2}. They are unique by definition of nice tree-decomposition. Note that $d(x, y) \geq 4$ since $d(x, Z) \geq 2$ and $d(y, Z) \geq 2$. So $N[x]$ is not adjacent to $N[y]$ and then $i_1 < i_2$. By Lemma 4, x' is on a shortest path between x and Z and y' is on a shortest path between y and Z. So $d(x', Z) < d(x, Z)$ and $d(y', Z) < d(y, Z)$. By hypothesis, there is a vertex $s \in S$ resolving the pair (x', y'). Let us prove that s resolves the pair (x, y).

If s belongs to $N[x]$ or to $N[y]$ then s resolves the pair (x, y) since $d(x, y) \geq 4$. So we can assume that $d(s, x) \geq 2$ and $d(s, y) \geq 2$. Let X_s be a bag of T containing s and X'_s be the closest bag to X_s on P between X and Y.

Case 1: $s \in X_{i_1}$ and $s \in X_{i_2}$. Then, $d(s, x') \leq 1$ and $d(s, y') \leq 1$. The vertex s resolves the pair (x', y') so $d(s, x') \neq d(s, y')$ so $s = x'$ or $s = y'$. Assume by symmetry that $s = x'$, then $d(s, x) = 1$ and $d(s, y) \geq 3$ because $d(x, y) \geq 4$. So s resolves the pair (x, y).

Case 2: s belongs to exactly one of X_{i_1} or X_{i_2}. By symmetry assume that $s \in X_{i_1}$. By Lemma 4, y' is on a shortest path between y and s. So $d(s, y) = d(s, y') + 1$. As s belongs to X_{i_1} then $d(x', s) \leq 1$ and $d(x, s) \leq 2$. As $d(y', s) \neq d(x', s)$ we have $d(y', s) \geq 2$, so $d(s, y) \geq 3$. Thus s resolves the pair (x, y).

Case 3: $s \notin X_{i_1}$ and $s \notin X_{i_2}$. First, we consider the case where X'_s is between X_{i_1} and X_{i_2}. Then, $d(s,x) = d(s,x') + 1$ and $d(s,y) = d(s,y') + 1$ by Lemma 4 as X_{i_1} separates y and s and X_{i_2} separates x and s. Thus, s resolves the pair (x,y).

By symmetry, we can now assume that X'_s is between X and X_{i_1}. Since $i_1 < i_2$, X_{i_2} separates s and y. So $d(s,y) = d(s,y')+1$ by Lemma 4. To conclude we prove that $d(s,x') < d(s,y')$. Let Q be a shortest path between s and y'. The bag X_{i_1} separates s and y' so $Q \cap X_{i_1} \neq \emptyset$. Let $y_1 \in Q \cap X_{i_1}$. By definition of Q, $d(s,y') = d(s,y_1) + d(y_1,y')$. Since $y_1, x' \in X_{i_1}$ and X_{i_1} is a clique, we have that $y_1 \in N[x']$ and so, $y_1 \neq y'$. So $d(y_1,y') \neq 0$. We also have $d(s,x') \leq d(s,y_1) + 1$ because y_1 is a neighbour of x'. As $d(s,x') \neq d(s,y')$, this ensures $d(s,x') < d(s,y')$. So s resolves the pair (x,y) because $d(s,x) \leq d(s,x') + 1 < d(s,y') + 1 = d(s,y)$. □

The following lemma is essentially rephrasing Lemma 5 to get the result on a set of vertices.

Lemma 6. *Let G be a chordal graph and S be a subset of vertices of G. Let T be a nice clique tree of G. Let X be a bag of T and let $T_1 = (X_1, E_1)$ and $T_2 = (X_2, E_2)$ be two connected components of $T \setminus X$. Assume that any pair of vertices (u,v) of $(X_1 \cup X) \times (X_2 \cup X)$ with $d(u,X) \leq 2$ and $d(v,X) \leq 2$ is resolved by S. Then any pair of vertices (u,v) of (X_1, X_2) with $|d(u,X) - d(v,X)| \leq 1$ is resolved by S.*

3 Algorithm Description

In this section, we fix a vertex v of a chordal graph G and consider a nice clique tree (T,X) rooted in v which exists by Lemma 2. We present an algorithm computing the smallest size of a resolving set of G containing v.

3.1 Extension of the Problem

Our dynamic programming algorithm computes the solution of a generalization of metric dimension which is easier to manipulate when we combine solutions. In this new problem, we will represent some vertices by vectors of distances. We define notations to edit vectors.

Definition 7. *Given a vector $\mathbf{r}$, the notation $\mathbf{r}_i$ refers to the i-th coordinate of $\mathbf{r}$.*

- *Let $\mathbf{r} = (r_1, \ldots, r_k) \in \mathbb{N}^k$ be a vector of size k and $m \in \mathbb{N}$. The vector $\mathbf{r}' = \mathbf{r}|\mathbf{m}$ is the vector of size $k+1$ with $r'_i = r_i$ for $1 \leq i \leq k$ and $r'_{k+1} = m$.*
- *Let $\mathbf{r} = (r_1, \ldots, r_k) \in \mathbb{N}^k$ be a vector of size k. The vector $\mathbf{r}^-$ is the vector of size $k-1$ with $r^-_i = r_i$ for $1 \leq i \leq k-1$.*

Definition 8. *Let i be a node of T and let $X_i = \{v_1, \ldots, v_k\}$ be the bag of i. For a vertex x of G, the* distance vector $\mathbf{d_{X_i}(x)}$ *of x to X_i is the vector of size k such that, for $1 \leq j \leq k$, $\mathbf{d_{X_i}(x)}_j = d(x, v_j)$. We define the set $d_{\leq 2}(X_i)$ as the set of distance vectors of the vertices of $T(X_i)$ at distance at most 2 of X_i in G (i.e. one of the coordinate is at most 2).*

Definition 9. *Let G be a graph and $K = \{v_1, \ldots, v_k\}$ be a clique of G. Let x be a vertex of G. The* trace *of x on K, denoted by $\mathbf{Tr_K}(x)$, is the vector $\mathbf{r}$ of $\{0,1\}^k \setminus \{1, \ldots, 1\}$ such that for every $1 \leq i \leq k$, $d(x, v_i) = a + \mathbf{r}_i$ where $a = d(x, K)$.*

Let S be a subset of vertices of G. The trace $Tr_K(S)$ of S in K is the set of vectors $\{\mathbf{Tr_K}(x), x \in S\}$.

The trace is well-defined because for a vertex x and a clique K, the distance between x and a vertex of K is either $d(x, K)$ or $d(x, K) + 1$.

Definition 10. *Let $\mathbf{r_1}, \mathbf{r_2}$ and $\mathbf{r_3}$ be three vectors of same size k. We say that $\mathbf{r_3}$* resolves *the pair $(\mathbf{r_1}, \mathbf{r_2})$ if*

$$\min_{1 \leq i \leq k} (\mathbf{r_1} + \mathbf{r_3})_i \neq \min_{1 \leq i \leq k} (\mathbf{r_2} + \mathbf{r_3})_i.$$

Lemma 11. *Let K be a clique separator of G and G_1 be a connected component of $G \setminus K$. Let (x, y) be a pair of vertices of $G \setminus G_1$ and let $\mathbf{r}$ be a vector of size $|K|$. If $\mathbf{r}$ resolves the pair $(\mathbf{d_K(x)}, \mathbf{d_K(y)})$, then any vertex $s \in V(G_1)$ with $\mathbf{Tr_K}(s) = \mathbf{r}$ resolves the pair (x, y).*

Proof. Let s be a vertex of G_1 such that $\mathbf{Tr_K}(s) = \mathbf{r}$. The clique K separates s and x (resp. y) so $d(x, s) = \min_{1 \leq i \leq |K|}(\mathbf{d_K(x)} + \mathbf{Tr_K}(s))_i + d(K, s)$ (resp. $d(y, s) = \min_{1 \leq i \leq |K|}(\mathbf{d_K(y)} + \mathbf{Tr_K}(s))_i + d(K, s)$). The vector $\mathbf{r}$ resolves the pair $(\mathbf{d_K(x)}, \mathbf{d_K(y)})$. So $d(x, s) \neq d(y, s)$ and s resolves the pair (x, y). □

Definition 12. *Let K be a clique separator of G and G_1, G_2 be two (non necessarily distinct) connected components of $G \setminus K$. Let M be a set of vectors and let $x \in V(G_1) \cup K$ and $y \in V(G_2) \cup K$. If a vector $\mathbf{r}$ resolves the pair $(\mathbf{d_K(x)}, \mathbf{d_K(y)})$, we say that $\mathbf{r}$* resolves *the pair (x, y). We say that the pair of vertices (x, y)* is resolved *by M if there exists a vector $\mathbf{r} \in M$ that resolves the pair (x, y).*

We can now define the generalised problem our dynamic programming algorithm actually solves. We call it the EXTENDED METRIC DIMENSION problem (EMD for short). We first define the instances of this problem.

Definition 13. *Let i be a node of T. An* instance for a node i *of the* EMD *problem is a 5-uplet $I = (X_i, S_I, D_{int}(I), D_{ext}(I), D_{pair}(I))$ composed of the bag X_i of i, a subset S_I of X_i and three sets of vectors satisfying*

- $D_{int}(I) \subseteq \{0,1\}^{|X_i|}$ *and* $D_{ext}(I) \subseteq \{0,1\}^{|X_i|}$,
- $D_{pair}(I) \subseteq \{0,1,2,3\}^{|X_i|} \times \{0,1,2,3\}^{|X_i|}$,

- $D_{ext}(I) \neq \emptyset$ *or* $S_I \neq \emptyset$,
- *For each pair of vectors* $(\mathbf{r_1}, \mathbf{r_2}) \in D_{pair}(I)$, *there exist two vertices* $x \in T(X_i)$ *with* $\mathbf{d_{X_i}}(\mathbf{x}) = \mathbf{r_1}$ *and* $d(x, X_i) \leq 2$ *and* $y \notin T(X_i)$ *with* $\mathbf{d_{X_i}}(\mathbf{y}) = \mathbf{r_2}$ *and* $d(y, X_i) \leq 2$.

Definition 14. *A set* $S \subseteq T(X_i)$ *is a solution for an instance* I *of the* EMD *problem if*

- ***(S1)*** *Every pair of vertices of* $T(X_i)$ *is either resolved by a vertex in* S *or resolved by a vector of* $D_{ext}(I)$.
- ***(S2)*** *For each vector* $\mathbf{r} \in D_{int}(I)$ *there exists a vertex* $s \in S$ *such that* $\mathbf{Tr_{X_i}}(s) = \mathbf{r}$.
- ***(S3)*** *For each pair of vector* $(\mathbf{r_1}, \mathbf{r_2}) \in D_{pair}(I)$, *for any vertex* $x \in T(X_i)$ *with* $\mathbf{d_{X_i}}(\mathbf{x}) = \mathbf{r_1}$ *and any vertex* $y \notin T(X_i)$ *with* $\mathbf{d_{X_i}}(\mathbf{y}) = \mathbf{r_2}$, *if* $d(x, X_i) \leq 2$ *and* $d(y, X_i) \leq 2$ *the pair* (x, y) *is resolved by* S.
- ***(S4)*** $S \cap X_i = S_I$.

In the rest of the paper, for shortness, we will refer to an instance of the EMD problem only by an instance.

Definition 15. *Let* I *be an instance. We denote by* $\dim(I)$ *the minimum size of a set* $S \subseteq T(X_i)$ *which is a solution of* I. *If such a set does not exist we define* $\dim(I) = +\infty$. *We call this value the* extended metric dimension *of* I.

We now explain the meaning of each element of I. Firstly, a solution S must resolve any pair in $T(X_i)$, possibly with a vector of $D_{ext}(I)$ which represents a vertex of $V \setminus T(X_i)$ in the resolving set. Secondly, for all $\mathbf{r}$ in $D_{int}(I)$, we are forced to select a vertex in $T(X_i)$ whose trace is $\mathbf{r}$. This will be useful to combine solutions since it will be a vector of D_{ext} in other instances. The elements in $D_{pair}(I)$ will also be useful for combinations. In some sense $D_{pair}(I)$ is the additional gain of S compared to the main goal to resolve $T(X_i)$. The set S_I constrains the intersection between S and X_i by forcing a precise subset of X_i to be in S.

The following lemma is a consequence of Definition 14. It connects the definition of the extended metric dimension with the metric dimension.

Lemma 16. *Let* G *be a graph,* T *be a nice tree-decomposition of* G *and* r *be the root of* T. *Let* I_0 *be the instance* $(\{r\}, \{r\}, \emptyset, \emptyset, \emptyset)$, *then* $\dim(I_0)$ *is the smallest size of a resolving set of* G *containing* r.

To ensure that our algorithm works well, we will need to use Lemma 3 in some subgraphs of G. This is possible only if we know that the solution is not included in the subgraph. This corresponds to the condition $D_{ext}(I) \neq \emptyset$ or $S_I \neq \emptyset$ and this is why the algorithm computes the size of a resolving set containing the root of T.

3.2 Dynamic Programming

We explain how we can compute the extended metric dimension of an instance I given the extended metric dimension of the instances on the children of X_i in T. The proof is divided according to the different type of nodes.

Leaf Node. Computing the extended metric dimension of an instance for a leaf node can be done easily with the following lemma:

Lemma 17. *Let I be an instance for a leaf node i and v be the unique vertex of X_i. Then,*

$$\dim(I) = \begin{cases} 0 & \text{if } S_I = \emptyset,\ D_{int}(I) = \emptyset \text{ and } D_{pair}(I) = \emptyset \\ 1 & \text{if } S_I = \{v\} \text{ and } D_{int}(I) \subseteq \{(\mathbf{0})\} \\ +\infty & \text{otherwise} \end{cases}$$

Proof. Let I be an instance for i. If $S_I = \emptyset$, only the set $S = \emptyset$ can be a solution for I. This set is a solution only if $D_{int}(I) = \emptyset$ and $D_{pair}(I) = \emptyset$. If $S_I = \{v\}$, only the set $S = \{v\}$ can be a solution for I. This is a solution only if $D_{int}(I)$ is empty or only contains the vector $\mathbf{Tr_{x_i}}(v)$. □

In the rest of the section, we treat the three other types of nodes. For each type of nodes we will proceed as follows: define some conditions on the instances on children to be compatible with I, and prove an equality between the extended metric dimension on compatible children instances and the extended metric dimension of the instance of the node.

Join Node. Let I be an instance for a join node i and let i_1 and i_2 be the children of i.

Definition 18. *A pair of instances (I_1, I_2) for (i_1, i_2) is* compatible *with I if*

- ***(J1)*** $S_{I_1} = S_{I_2} = S_I$,
- ***(J2)*** $D_{ext}(I_1) \subseteq D_{ext}(I) \cup D_{int}(I_2)$ *and* $D_{ext}(I_2) \subseteq D_{ext}(I) \cup D_{int}(I_1)$,
- ***(J3)*** $D_{int}(I) \subseteq D_{int}(I_1) \cup D_{int}(I_2)$,
- ***(J4)*** *Let* $C_1 = \{(\mathbf{r}, \mathbf{t}) \in D_{pair}(I_1)$ *such that* $\mathbf{r} \notin d_{\leq 2}(X_{i_1})\}$ *and* $C_2 = \{(\mathbf{r}, \mathbf{t}) \in D_{pair}(I_2)$ *such that* $\mathbf{r} \notin d_{\leq 2}(X_{i_2})\}$. *Let* $D_1 = \{(\mathbf{r}, \mathbf{t}) \in d_{\leq 2}(X_{i_1}) \times d_{\leq 2}(G \setminus X_{i_1})$ *such that there exists* $\mathbf{u} \in D_{int}(I_2)$ *resolving the pair* $(\mathbf{r}, \mathbf{t})\}$ *and* $D_2 = \{(\mathbf{r}, \mathbf{t}) \in d_{\leq 2}(X_{i_2}) \times d_{\leq 2}(G \setminus X_{i_2}))$ *such that there exists* $\mathbf{u} \in D_{int}(I_1)$ *resolving the pair* $(\mathbf{r}, \mathbf{t})\}$ *Then* $D_{pair}(I) \subseteq (C_1 \cup D_1 \cup D_{pair}(I_1)) \cap (C_2 \cup D_2 \cup D_{pair}(I_2))$,
- ***(J5)*** *For all* $\mathbf{r_1} \in d_{\leq 2}(X_{i_1})$, *for all* $\mathbf{r_2} \in d_{\leq 2}(X_{i_2})$, $(\mathbf{r_1}, \mathbf{r_2}) \in D_{pair}(I_1)$ *or* $(\mathbf{r_2}, \mathbf{r_1}) \in D_{pair}(I_2)$ *or there exists* $\mathbf{t} \in D_{ext}(I)$ *such that* $\mathbf{t}$ *resolves the pair* $(\mathbf{r_1}, \mathbf{r_2})$.

Condition **(J4)** represents how the pairs of vertices of $V(T(X_{i_1})) \times V(T(X_{i_2}))$ can be resolved. A pair $(\mathbf{r}, \mathbf{t})$ is in $(C_1 \cup D_1 \cup D_{pair}(I_1))$ if all the pairs of vertices (x, y) with $x \in V(T(X_{i_1}))$ and $y \in V(T(X_{i_2}))$ are resolved. If $(\mathbf{r}, \mathbf{t})$ is in C_1, no pair (x, y) with $x \in V(T(X_{i_1}))$ and $y \in V(T(X_{i_2}))$ exists, if $(\mathbf{r}, \mathbf{t})$ is in D_1 the pairs of vertices are resolved by a vertex outside of $V(T(X_{i_1}))$ and if $(\mathbf{r}, \mathbf{t})$ is in $D_{pair}(I_1)$ the pairs of vertices are resolved by a vertex of $V(T(X_{i_1}))$. So a pair $(\mathbf{r}, \mathbf{t})$ is resolved if the pair is in $(C_1 \cup D_1 \cup D_{pair}(I_1))$ and in $(C_2 \cup D_2 \cup D_{pair}(I_2))$.

Let $\mathcal{F}_J(I)$ be the set of pairs of instances compatible with I. We want to prove the following lemma:

Lemma 19. *Let I be an instance for a join node i. Then,*

$$\dim(I) = \min_{(I_1,I_2)\in\mathcal{F}_J(I)}(\dim(I_1)+\dim(I_2)-|S_I|).$$

We prove the equality by proving the two inequalities in the next lemmas.

Lemma 20. *Let (I_1, I_2) be a pair of instances for (i_1, i_2) compatible with I with finite values for $\dim(I_1)$ and $\dim(I_2)$. Let $S_1 \subseteq V(T(X_{i_1}))$ be a solution for I_1 and $S_2 \subseteq V(T(X_{i_2}))$ be a solution for I_2. Then $S = S_1 \cup S_2$ is a solution for I. In particular,*

$$\dim(I) \leq \min_{(I_1,I_2)\in\mathcal{F}_J(I)}(\dim(I_1)+\dim(I_2)-|S_I|).$$

Proof. Let us prove that the conditions of Definition 14 are satisfied.
(S1) Let (x, y) be a pair of vertices of $T(X_i)$. Assume first that $x \in V(T(X_{i_1}))$ and $y \in V(T(X_{i_1}))$. Either (x, y) is resolved by a vertex of S_1 and then by a vertex of S or (x, y) is resolved by a vector $\mathbf{r} \in D_{ext}(I_1)$. By condition **(J2)**, $\mathbf{r} \in D_{ext}(I)$ or $\mathbf{r} \in D_{int}(I_2)$. If $\mathbf{r} \in D_{ext}(I)$ then (x, y) is resolved by a vector of $D_{ext}(I_1)$. Otherwise, there exists a vertex $t \in S_2$ such that $\mathbf{Tr}_{\mathbf{X_{i_2}}}(t) = \mathbf{r}$. So $t \in S$ and t resolves the pair (x, y). The case $x \in V(T(X_{i_2}))$ and $y \in V(T(X_{i_2}))$ is symmetric. So we can assume that $x \in V(T(X_{i_1}))$ and $y \in V(T(X_{i_2}))$. If $d(x, X_i) \leq 2$ and $d(y, X_i) \leq 2$, the condition **(J5)** ensures that the pair (x, y) is resolved by S or by a vector of $D_{ext}(I)$. Otherwise, either $|d(x, X_i)-d(y, X_i)| \leq 1$ and (x, y) is resolved by Lemma 6 or $|d(x, X_i)-d(y, X_i)| \geq 2$ and (x, y) is resolved by Lemma 3 because $D_{ext}(I) \neq \emptyset$ or $S_I \neq \emptyset$.
(S2) Let $\mathbf{r} \in D_{int}(I)$. By compatibility, the condition **(J3)** ensures that $\mathbf{r} \in D_{int}(I_1)$ or $\mathbf{r} \in D_{int}(I_2)$. As $S = S_1 \cup S_2$, S contains a vertex s such that $\mathbf{Tr}_{\mathbf{X_i}}(s) = \mathbf{r}$.
(S3) Let $(\mathbf{r}, \mathbf{t}) \in D_{pair}(I)$ and (x, y) with $x \in V(T(X_i))$ such that $\mathbf{d}_{\mathbf{X_i}}(\mathbf{x}) = \mathbf{r}$ and $y \notin T(X_i)$ such that $\mathbf{d}_{\mathbf{X_i}}(\mathbf{y}) = \mathbf{t}$. Without loss of generality assume that $x \in V(T(X_{i_1}))$.

By compatibility, $(\mathbf{r}, \mathbf{t}) \in (C_1 \cup D_1 \cup D_{pair}(I_1)) \cap (C_2 \cup D_2 \cup D_{pair}(I_2))$ so in $C_1 \cup D_1 \cup D_{pair}(I_1)$. If $(\mathbf{r}, \mathbf{t}) \in D_{pair}(I)_1$, then there exists $s \in S_1$ that resolves the pair (x, y) so the pair is resolved by S. If $(\mathbf{r}, \mathbf{t}) \in D_1$, there exists $\mathbf{u} \in D_{int}(I_2)$ such that $\mathbf{u}$ resolves the pair $(\mathbf{r}, \mathbf{t})$. By compatibility, there exists $s \in S_2$ such that $\mathbf{Tr}_{\mathbf{X_i}}(s) = \mathbf{u}$. So s resolves the pair (x, y). And $(\mathbf{r}, \mathbf{t}) \notin C_1$ since x belongs to $T(X_{i_1})$ with vector distance $\mathbf{r}$.
(S4) is clear since $X_{i_1} = X_{i_2} = X_i$.

Thus, $\dim(I) \leq \dim(I_1) + \dim(I_2) - |S_I|$ is true for any pair of compatible instances (I_1, I_2) so $\dim(I) \leq \min_{(I_1,I_2)\in\mathcal{F}_J(I)}(\dim(I_1) + \dim(I_2) - |S_I|)$. □

Lemma 21. *Let I be an instance for a join node i and let i_1 and i_2 be the children of i. Then,*

$$\dim(I) \geq \min_{(I_1,I_2)\in\mathcal{F}_J(I)}(\dim(I_1)+\dim(I_2)-|S_I|).$$

Proof. If $\dim(I) = +\infty$ then the result indeed holds. So we can assume that $\dim(I)$ is finite. Let S be a solution for I of minimal size. Let $S_1 = S \cap T(X_{i_1})$ and $S_2 = S \cap T(X_{i_2})$. We define now two instances I_1 and I_2 for i_1 and i_2. Let $S_{I_1} = S_{I_2} = S_I$, $D_{int}(I_1) = Tr_{X_i}(S_1)$, $D_{int}(I_2) = Tr_{X_i}(S_2)$, $D_{ext}(I_1) = D_{ext}(I) \cup D_{int}(I_2)$ and $D_{ext}(I_2) = D_{ext}(I) \cup D_{int}(I_1)$. To build the sets $D_{pair}(I_1)$ and $D_{pair}(I_2)$ we make the following process that we explain for $D_{pair}(I_1)$. For all pairs of vectors $(\mathbf{r}, \mathbf{t})$ of $(d_{\leq 2}(X_{i_1}), d_{\leq 2}(G \setminus X_{i_1}))$, consider all the pairs of vertices (x, y) with $x \in V(T(X_{i_1}))$, $y \in V(G \setminus T(X_{i_1}))$, $\mathbf{r} \in d_{\leq 2}(X_i)$, $\mathbf{t} \in d_{\leq 2}(G \setminus X_{i_1}))$, $\mathbf{d_{X_i}}(\mathbf{x}) = \mathbf{r}$ and $\mathbf{d_{X_i}}(\mathbf{y}) = \mathbf{t}$. If all the pairs are resolved by vertices of S_1 (that is for each pair, there exists a vertex of S_1 that resolves the pair), then add $(\mathbf{r}, \mathbf{t})$ to $D_{pair}(I_1)$.

Checking that (I_1, I_2) is compatible with I, that S_1 is a solution of I_1, and that S_2 is a solution of I_2 is straightforward. It consists of checking conditions of respectively Definition 18 and Definition 14.

Finally we prove the announced inequality. Since S is a minimal solution for I, we have $\dim(I) = |S|$. The sets S_1 and S_2 are solutions for S_1 and S_2 so $\dim(I_1) \leq |S_1|$ and $\dim(I_2) \leq |S_2|$. Since $|S| = |S_1| + |S_2| - |S_I|$, $\dim(I) \geq \dim(I_1) + \dim(I_2) - |S_I|$, giving the result. □

Lemma 19 is a direct consequence of Lemma 20 and Lemma 21.

Introduce Node. We now consider an instance I for an introduce node i. Let j be the child of i and $v \in V$ be such that $X_i = X_j \cup \{v\}$. Let $X_i = \{v_1, \ldots, v_k\}$ with $v = v_k$. The tree $T(X_i)$ contains one more vertex than its child. The definition of the compatibility is slightly different if we consider the same set as a solution (type 1) or if we add this vertex to the resolving set (type 2).

Definition 22. *An instance I_1 is compatible with I of type 1 (resp. 2) if*

- ***(I1)*** *$S_I = S_{I_1}$ (resp. $= S_{I_1} \cup \{v\}$).*
- ***(I2)*** *For all $\mathbf{r} \in D_{ext}(I)$, $\mathbf{r}^- \in D_{ext}(I_1)$ (resp. or $\mathbf{r} = (0, \ldots, 0)$).*
- ***(I3)*** *For all $\mathbf{r} \in D_{int}(I), \mathbf{r}_k = 1$ and $\mathbf{r}^- \in D_{int}(I_1)$ (resp. or $\mathbf{r} = (1, \ldots, 1, 0)$).*
- ***(I4)*** *For all $(\mathbf{r}, \mathbf{t}) \in D_{pair}(I)$, $(\mathbf{r}^-, \mathbf{t}^-) \in D_{pair}(I_1)$.*
- ***(I5)*** *If I_1 is of type 1, for all $(\mathbf{r}, \mathbf{t})$ with $\mathbf{t} = (0, \ldots, 0)$, $(\mathbf{r}, \mathbf{t}) \in D_{pair}(I_1)$.*

Lemma 23. *Let I be an instance for an introduce node i. Let $\mathcal{F}_1(I)$ be the set of instances I_1 for i_1 compatible with I of type 1 and $\mathcal{F}_2(I)$ be the set of instances I_2 for i_1 compatible with I of type 2. Then,*

$$\dim(I) = \min\{\min_{I_1 \in \mathcal{F}_1(I)}\{\dim(I_1)\}; \min_{I_2 \in \mathcal{F}_2(I)}\{\dim(I_2) + 1\}\}.$$

The proof of Lemma 23 consists in proving both inequalities similarly to Lemma 19. One inequality comes from the fact that we can get a solution of I from any compatible instance. The other consists in building a solution for a compatible instance from a minimal solution for I.

Forget Node. The construction for the forget nodes is similar to the one for introduce nodes. The main difference is that a vertex is removed from the bag so we have to keep the information about this vertex. The full construction leads to a similar equality between the extended metric dimension of an instance and the extended metric dimension of the compatible instances of its child.

3.3 Algorithm

Given as input a nice clique tree, the algorithm computes the extended metric dimension bottom up from the leaves. The algorithm computes the extended metric dimension for leaves using Lemma 17, for join nodes using Lemma 19, for introduce nodes using Lemma 23 and forget nodes using a similar lemma. The correction of the algorithm is straightforward by these lemmas.

We denote this algorithm by IMD in the following which takes as input a nice clique tree T and outputs the minimal size of a resolving set of G containing the root of T.

4 Proof of Theorem 1

Let us finally explain how we can compute the metric dimension of G. The following lemma is a consequence of Lemma 16.

Lemma 24. *The metric dimension of G is $\min_{v \in V(G)}\{IMD(T(v))\}$ where $T(v)$ is a nice clique tree of G rooted in v.*

So, n executions of the IMD algorithm with different inputs are enough to compute the metric dimension. Lemma 2 ensures that we can find for any vertex v of G a nice clique tree in linear time, the last part is to compute the complexity of the IMD algorithm.

Lemma 25. *The algorithm for IMD runs in time $O(n(T)^2 + n(T) \cdot f(\omega))$ where $n(T)$ is the number of vertices of the input tree T and $f = O(\omega^2 \cdot 2^{O(4^{2^\omega})})$ is a function that only depends on the size of a maximum clique ω.*

We now have all the ingredients to prove Theorem 1:

Proof. For each vertex v of G, one can compute a nice clique tree of size at most $7n$ according to Lemma 2. Given this clique tree, the IMD algorithm outputs the size of a smallest resolving set containing v by Lemma 16 in time $O(n(T)^2 + n(T) \cdot f(\omega))$ for a computable function f according to Corollary 25. Repeat this for all vertices of G permits to compute the metric dimension of G by Lemma 24 in time $O(n^3 + n^2 \cdot f(\omega))$. □

References

1. Belmonte, R., Fomin, F.V., Golovach, P.A., Ramanujan, M.S.: Metric dimension of bounded tree-length graphs. CoRR abs/1602.02610 (2016)
2. Bonnet, É., Purohit, N.: Metric dimension parameterized by treewidth. Algorithmica **83**(8), 2606–2633 (2021)
3. Chartrand, G., Lesniak, L., Zhang, P.: Graphs and Digraphs, 6th edn. Chapman and Hall/CRC (2015)
4. Díaz, J., Pottonen, O., Serna, M., van Leeuwen, E.J.: On the complexity of metric dimension. In: Epstein, L., Ferragina, P. (eds.) Algorithms - ESA 2012 (2012)
5. Dirac, G.A.: On rigid circuit graphs. Abh. Math. Semin. Univ. Hambg. **25**, 71–76 (1961). https://doi.org/10.1007/BF02992776
6. Epstein, L., Levin, A., Woeginger, G.J.: The (weighted) metric dimension of graphs: hard and easy cases. Algorithmica **72**(4), 1130–1171 (2015)
7. Eroh, L., Feit, P., Kang, C.X., Yi, E.: The effect of vertex or edge deletion on the metric dimension of graphs. J. Comb **6**(4), 433–444 (2015)
8. Foucaud, F., Mertzios, G.B., Naserasr, R., Parreau, A., Valicov, P.: Identification, location-domination and metric dimension on interval and permutation graphs. II. Algorithms and complexity. Algorithmica **78**(3), 914–944 (2017)
9. Garey, J.: A guide to the theory of NP-completeness. J. Algorithms (1979)
10. Gima, T., Hanaka, T., Kiyomi, M., Kobayashi, Y., Otachi, Y.: Exploring the gap between treedepth and vertex cover through vertex integrity. Theor. Comput. Sci. **918**, 60–76 (2022)
11. Harary, F., Melter, R.A.: On the metric dimension of a graph. Ars Combinatoria **2**, 191–195 (1975)
12. Hartung, S., Nichterlein, A.: On the parameterized and approximation hardness of metric dimension. In: 2013 IEEE Conference on Computational Complexity, pp. 266–276. IEEE (2013)
13. Kloks, T.: Treewidth: Computations and Approximations. Springer, Heidelberg (1994)
14. Li, S., Pilipczuk, M.: Hardness of metric dimension in graphs of constant treewidth. Algorithmica **84**(11), 3110–3155 (2022)
15. Slater, P.J.: Leaves of trees. Congressus Numerantium **14** (1975)