Algorithms and Hardness for Metric Dimension on Digraphs

Antoine Dailly[1(✉)], Florent Foucaud[1], and Anni Hakanen[1,2]

[1] Université Clermont Auvergne, CNRS, Clermont Auvergne INP, Mines
Saint-Étienne, LIMOS, 63000 Clermont-Ferrand, France
{antoine.dailly,florent.foucaud,anni.hakanen}@uca.fr
[2] Department of Mathematics and Statistics, University of Turku,
20014 Turku, Finland

Abstract. In the METRIC DIMENSION problem, one asks for a minimum-size set R of vertices such that for any pair of vertices of the graph, there is a vertex from R whose two distances to the vertices of the pair are distinct. This problem has mainly been studied on undirected graphs and has gained a lot of attention in the recent years. We focus on directed graphs, and show how to solve the problem in linear-time on digraphs whose underlying undirected graph (ignoring multiple edges) is a tree. This (nontrivially) extends a previous algorithm for oriented trees. We then extend the method to unicyclic digraphs (understood as the digraphs whose underlying undirected multigraph has a unique cycle). We also give a fixed-parameter-tractable algorithm for digraphs when parameterized by the directed modular-width, extending a known result for undirected graphs. Finally, we show that METRIC DIMENSION is NP-hard even on planar triangle-free acyclic digraphs of maximum degree 6.

1 Introduction

The metric dimension of a (di)graph G is the smallest size of a set of vertices that distinguishes all vertices of G by their vectors of distances from the vertices of the set. This concept was introduced in the 1970s by Harary and Melter [14] and by Slater [30] independently. Due to its interesting nature and numerous applications (such as robot navigation [18], detection in sensor networks [30] or image processing [22], to name a few), it has enjoyed a lot of attention. It also has been studied in the more general setting of metric spaces [3], and is generally part of the rich area of identification problems of graphs and other discrete structures [20].

More formally, let us denote by $\mathrm{dist}(x, y)$ the distance from x to y in a digraph. Here, the distance $\mathrm{dist}(x, y)$ is taken as the length of a shortest directed

path from x to y; if no such path exists, $\text{dist}(x,y)$ is infinite, and we say that y is not *reachable* from x. We say that a set S is a *resolving set* of a digraph G if for any pair of distinct vertices v,w from G, there is a vertex x in S with $\text{dist}(x,v) \neq \text{dist}(x,w)$. Furthermore, we require that every vertex of G is reachable from at least one vertex of S. The *metric dimension* of G is the smallest size of a resolving set of G, and a minimum-size resolving set of G is called a *metric basis* of G.[1]

We denote by METRIC DIMENSION the computational version of the problem: given a (di)graph G, determine its metric dimension.

For undirected graphs, METRIC DIMENSION has been extensively studied, and its non-local nature makes it highly nontrivial from an algorithmic point of view. On the hardness side, METRIC DIMENSION was shown to be NP-hard for planar graphs of bounded degree [7], split, bipartite and line graphs [9], unit disk graphs [17], interval and permutation graphs of diameter 2 [11], and graphs of pathwidth 24 [19]. On the positive side, it can easily be solved in linear time on trees [4,14,18,30]. More involved polynomial-time algorithms exist for unicyclic graphs [29] and, more generally, graphs of bounded cyclomatic number [9]; outerplanar graphs [7]; cographs [9]; chain graphs [10]; cactus-block graphs [16]; bipartite distance-hereditary graphs [24]. There are fixed parameter tractable (FPT) algorithms for the undirected graph parameters max leaf number [8], tree-depth [13], modular-width [2] and distance to cluster [12], but FPT algorithms are highly unlikely to exist for parameters solution size [15] and feedback vertex set [12].

Due to the interest for METRIC DIMENSION on undirected graphs, it is natural to ask what can be said in the context of digraphs. The metric dimension of digraphs was first studied in [5] under a somewhat restrictive definition; for our definitions, we follow the recent paper [1], in which the algorithmic aspects of METRIC DIMENSION on digraphs have been addressed. We call *oriented graph* a digraph without directed 2-cycles. A *directed acyclic digraph* (DAG for short) has no directed cycles at all. The *underlying multigraph* of a digraph is the one obtained by ignoring the arc orientations; its *underlying graph* is obtained from it by ignoring multiple edges. In a digraph, a *strongly connected component* is a subgraph where every vertex is reachable from all other vertices. Note that for the METRIC DIMENSION problem, undirected graphs can be seen as a special type of digraphs where each arc has a symmetric arc.

The NP-hardness of METRIC DIMENSION was proven for oriented graphs in [27] and, more recently, for bipartite DAGs of maximum degree 8 and maximum distance 4 [1] (the *maximum distance* being the length of a longest directed path without shortcuts). A linear-time algorithm for METRIC DIMENSION on oriented trees was given in [1].

---

[1] The definition that we use has been called *strong metric dimension* in [1], as opposed to *weak metric dimension*, where one single vertex may be unreachable from any resolving set vertex. The former definition seems more natural to us. However, the term *strong metric dimension* is already used for a different concept, see [25]. Thus, to prevent confusion, we avoid the prefix *strong* in this paper.

Our Results. We generalize the linear-time algorithm for METRIC DIMENSION on oriented trees from [1] to all digraphs whose underlying graph is a tree. In other words, here we allow 2-cycles. This makes a significant difference with oriented trees, and as a result our algorithm is nontrivial. We then extend the used methods to solve METRIC DIMENSION in linear time for unicyclic digraphs (digraphs with a unique cycle). Then, we prove that METRIC DIMENSION can be solved in time $f(t)n^{O(1)}$ for digraphs of order n and modular-width t (a parameter recently introduced for digraphs in [31]). This extends the same result for undirected graphs from [2], and is the first FPT algorithm for METRIC DIMENSION on digraphs. Finally, we complement the hardness result from [1] by showing that METRIC DIMENSION is NP-hard even for planar triangle-free DAGs of maximum degree 6 and maximum distance 4. For results marked with ($*$), we omit the full proof due to space constraints; those can be found in [6].

2   Digraphs Whose Underlying Graph is a Tree

For the sake of convenience, we call *di-tree* a digraph whose underlying graph is a tree. Trees are often the first non-trivial class to study for a graph problem. METRIC DIMENSION is no exception to this, having been studied in the first papers for the undirected [4,14,18,30] and the oriented [1] cases. In the undirected case, a minimum-size resolving set can be found by taking, for each vertex of degree at least 3 spanning k legs, the endpoint of $k-1$ of its legs (a *leg* is an induced path spanning from a vertex of degree at least 3, having its inner vertices of degree 2, and ending in a leaf). In the case of oriented trees, taking all the sources (a *source* is a vertex with no in-neighbour) and $k-1$ vertices in each set of k in-twins yields a metric basis (two vertices are *in-twins* if they have the same in-neighbourhood). Our algorithm, being on di-trees (which include both undirected trees and oriented trees), will reuse those strategies, but we will need to refine them in order to obtain a metric basis. The first refinement is of the notion of in-twins:

Definition 1. *A strongly connected component E of a di-tree is an* escalator *if it satisfies the following conditions:*

1. *its underlying graph is a path with vertices $e_1, \ldots, e_k$ ($k \geq 2$);*
2. *there is a unique vertex $y \notin E$ such that the arc $\overrightarrow{ye_1}$ (resp. $\overrightarrow{ye_k}$) exists;*
3. *there can be any number (possibly, zero) of vertices $z \notin E$ such that the arc $\overrightarrow{e_k z}$ (resp. $\overrightarrow{e_1 z}$) exists; for every $i \in \{1, \ldots, k-1\}$ (resp. $i \in \{2, \ldots, k\}$), no arc $\overrightarrow{e_i z}$ with $z \notin E$ exists.*

Definition 2. *In a di-tree, a set of vertices $A = \{a_1, \ldots, a_k\}$ is a set of* almost-in-twins *if there is a vertex x such that:*

1. *for every $i \in \{1, \ldots, k\}$, the arc $\overrightarrow{xa_i}$ exists and the arc $\overrightarrow{a_i x}$ does not exist;*
2. *for every $i \in \{1, \ldots, k\}$, either a_i is a trivial strongly connected component and $N^-(a_i) = \{x\}$, or a_i is the endpoint of an escalator and $N^-(a_i) = \{x, y\}$ where y is its neighbour in the escalator.*

Note that regular in-twins are also almost-in-twins. The second refinement is the following (for a given vertex x in a strongly connected component with C as an underlying graph, we call $d_C(x)$ the degree of x in C):

Definition 3. *Given the underlying graph C of a strongly connected component of a di-tree and a set D of vertices, we call a set S of vertices inducing a path of order at least 2 in C a* special leg *if it verifies the four following properties:*

1. *S has a unique vertex v such that $v \in D$ or $d_C(v) \geq 3$;*
2. *S has a unique vertex w such that $d_C(w) = 1$, furthermore $w \notin D$: w is called the* endpoint *of S;*
3. *all of the other vertices x of S verify $d_C(x) = 2$ and $x \notin D$;*
4. *at least one of the vertices $y \in S \setminus \{w\}$ has an out-arc $\overrightarrow{yz}$ with $z \notin C$.*

Note that several special legs can span from the same vertex, from which regular legs can also span. Algorithm 1, illustrated in Fig. 1, computes a metric basis of a di-tree.

Explanation of Algorithm 1. The algorithm will compute a metric basis $\mathcal{B}$ of a di-tree T in linear-time. The first thing we do is to add every source in T to $\mathcal{B}$ (line 1). Then, for every set of almost-in-twins, we add all of them but one to $\mathcal{B}$ (lines 2–3). Those two first steps, depicted in Fig. 1a, are the ones used to compute the metric basis of an orientation of a tree [1], and as such they are still necessary for managing the non-strongly connected components of the di-tree. Note that we are specifically managing sets of *almost-in-twins*, which include sets of in-twins, since it is necessary to resolve the specific case of escalators. The rest of the algorithm consists in managing the strongly connected components.

For each strongly connected component having C as an underlying graph, we first identify each vertex x of C that has an in-arc coming from **outside** C. Indeed, since x is the "last" vertex of a path coming from outside C, there are vertices of $\mathcal{B}$ "behind" this in-arc (or they can themselves be a vertex in $\mathcal{B}$), which we will call $\mathcal{B}_x$. However, the vertices in $\mathcal{B}_x$ can be "projected" on x since, T being a di-tree, x is on every shortest path from the vertices of $\mathcal{B}$ "behind" the in-arc to the vertices of C. Hence, we will mark x as a **dummy vertex** (lines 5–7, depicted in Figure 1b): we will consider that it is in $\mathcal{B}$ for the rest of this step, and acts as a representative of the set $\mathcal{B}_x$ with respect to C.

We then have to manage some specific cases whenever C is a path (lines 8–17). Indeed, the last two steps of the algorithm do not always work under some conditions. Those specific conditions are highlighted in the proof.

The last two steps are then applied. First, we have to consider the **special legs** defined in Definition 3. The idea behind those special legs is the following: for every out-arc $\overrightarrow{yz}$ with y in the special leg and z outside of C, any vertex in the metric basis "before" the start of the special leg will not distinguish z and the next neighbour of y in the special leg. Hence, we have to add at least one vertex to $\mathcal{B}$ for each special leg, and we choose the endpoint of the special leg (lines 18–19, depicted in Fig. 1c). Finally, we apply the well-known algorithm for

---

Algorithm 1: An algorithm computing the metric basis of a di-tree.

Input : A di-tree T.
Output: A metric basis $\mathcal{B}$ of T.

1 $\mathcal{B} \leftarrow$ Every source of T
2 foreach *set I of almost-in-twins* **do**
3   Add $|I| - 1$ vertices of I to $\mathcal{B}$

4 foreach *strongly connected component with C as an underlying graph* **do**
5 $D \leftarrow \emptyset$
6   **foreach** *arc $\overrightarrow{uv}$ with $v \in C$ and $u \notin C$* **do**
7   Add v to D

8   **if** *C is a path with endpoints x and y* **then**
9   **if** *there is no vertex in $C \cap D$* **then**
10   **if** *there is no out-arc from C to outside of C* **then**
11   Add x to $\mathcal{B}$
12   **else if** *there is an out-arc from x (resp. y) to outside of C and no other out-arc from C to outside of C* **then**
13   Add y (resp. x) to $\mathcal{B}$
14   **else**
15   Add x and y to $\mathcal{B}$
16   **else if** *there is exactly one vertex w in $C \cap D$, w is neither x nor y, and there is no out-arc from w to outside of C* **then**
17   Add x to $\mathcal{B}$

18   **foreach** *special leg L of C* **do**
19   Add the endpoint of L to $\mathcal{B}$
20   **foreach** *vertex of degree ≥ 3 in C from which span $k \geq 2$ legs of C that do not have a vertex in $\mathcal{B}$ or in D* **do**
21   Add the endpoint of $k - 1$ such legs to $\mathcal{B}$

22 return $\mathcal{B}$

---

computing the metric basis of a tree to the remaining parts of C (lines 20–21, depicted in Fig. 1d). The special legs and the legs containing a dummy vertex, being already resolved, are not considered in this part.

Theorem 4 (∗)**.** *Algorithm 1 computes a metric basis of a di-tree in linear time.*

3   Orientations of Unicyclic Graphs

A unicyclic graph U is constituted of a cycle C with vertices $c_1, \ldots, c_n$, and each vertex c_i is the root of a tree T_i (we can have T_i be simply the isolated c_i itself). The metric dimension of an undirected unicyclic graph has been studied in [26, 28, 29]. In [26], Poisson and Zhang proved bounds for the metric dimension
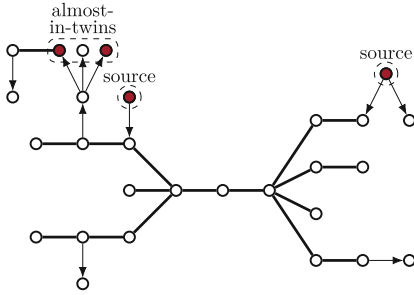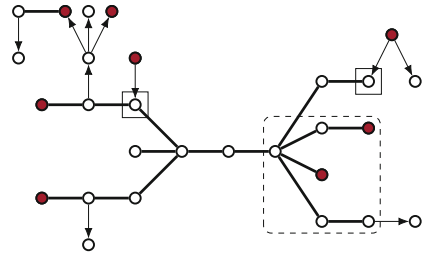
(a) The first step (lines 1-3) is to add every source and manage sets of almost-in-twins.
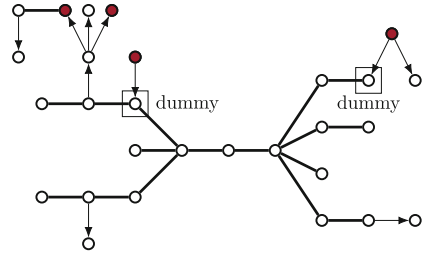
(b) The second step (lines 5-7) is to mark the dummy vertices of the strongly connected component.
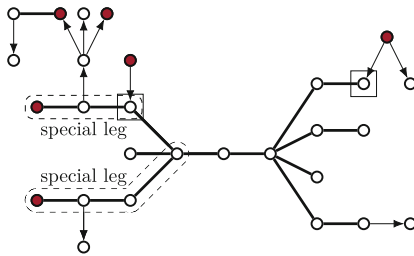
(c) The third step (lines 18-19) is to manage all the special legs.

(d) The fourth and final step (lines 20-21) is to manage the remaining legs with a common ancestor.

Fig. 1. Illustration of Algorithm 1. For the sake of simplicity, there are only two strongly connected components, for which we only represent the underlying graph with bolded edges, so every bolded edge is a 2-cycle. One of the two strongly connected components is a simple path that does not require any action. Vertices in the metric basis are colored in red. (Color figure online)

of a unicyclic graph in terms of the metric dimension of a tree we obtain by removing one edge from the cycle. Sedlar and Škrekovski showed more recently that the metric dimension of a unicyclic graph is one of two values in [28], and then the exact value of the metric dimension based on the structure of the graph in [29]. In this section, we will show that one can compute a metric basis of an orientation of a unicyclic graph in linear time. The algorithm mostly consists in using sources and in-twins, with a few specific edge cases to consider.

In this section, an *induced directed path* $\overrightarrow{P}$ is the orientation of an induced path with only one source and one sink which are its two endpoints. It is said to be *spanning from u* if u is its source endpoint, and its *length* is its number of edges. We also need the following definition:

Definition 5. *Let $\overrightarrow{U}$ be the orientation of a unicyclic graph. Given an orientation of a cycle $\overrightarrow{C}$ of even length $n = 2k$ with two sources, if its sources are c_i and c_{i+2}, its sinks are c_{i+1} and c_{i+1+k}, and there are, in $\overrightarrow{C} \setminus \{c_i, c_{i+2}\}$, neither in-twins nor in-arcs coming from outside of $\overrightarrow{C}$, we call an induced directed path $\overrightarrow{P}$ an* concerning path *if it verifies the three following properties:*

1. *$\overrightarrow{P}$ spans from c_{i+1};*
2. *$\overrightarrow{P}$ has length $k - 2$;*
3. *$\overrightarrow{P}$ has no in-arc coming from outside of $\overrightarrow{P} \cup \overrightarrow{C}$;*

Furthermore, if, for every vertex in $\overrightarrow{P}$ belonging to a nonempty set I of in-twins, every vertex in I belongs to a concerning path, then, we call $\overrightarrow{P}$ an unfixable path.

A path that is a concerning path, but not an unfixable path, will be called a fixable path.

Finally, a vertex might belong both to an unfixable path and to a fixable path; in this case, the fixable path takes precedence (i.e., we will consider that the vertex belongs to the fixable path).

Algorithm 2: An algorithm computing the metric basis of an orientation of a unicyclic graph.

Input   : An orientation $\overrightarrow{U}$ of a unicyclic graph U.
Output: A metric basis $\mathcal{B}$ of $\overrightarrow{U}$.

1 Add to $\mathcal{B}$ every source of $\overrightarrow{U}$
2 Apply the **special cases** in Algorithm 3
3 **foreach** *set I of in-twins in $\overrightarrow{U}$ that are not already in $\mathcal{B}$* **do**
4   **if** *all the vertices of I are in concerning paths* **then**
5   Add $|I| - 1$ vertices of I to $\mathcal{B}$, prioritizing vertices in unfixable paths
6   **else**
7   Add $|I| - 1$ vertices of I to $\mathcal{B}$, prioritizing vertices in the cycle $\overrightarrow{C}$ or in concerning paths, if there are any

8 **return** $\mathcal{B}$

Explanation of Algorithm 2. The algorithm will compute a metric basis $\mathcal{B}$ of an orientation $\overrightarrow{U}$ of a unicyclic graph U in linear-time. The first thing we do is to add every source in $\overrightarrow{U}$ to $\mathcal{B}$ (line 1). We will also manage the sets of in-twins in $\overrightarrow{U}$ (lines 3–7), which we need to do after taking care of some special cases that might influence the choice of in-twins. When we have the choice, we prioritize taking in-twins that are in the cycle to guarantee reachability of vertices in the cycle. Note that those two sets (along with the right priority) are enough in most cases.

Algorithm 3: Special cases of Algorithm 2.

---

1 **if** *the cycle $\overrightarrow{C}$ has no sink, there is no in-arc coming from outside of C, and no vertex of C is in a set of in-twin* **then**

2 $\quad$ Add c_1 to $\mathcal{B}$

3 **if** *the cycle $\overrightarrow{C}$ has no sink, there is exactly one in-arc $\overrightarrow{uc_i}$ with $u \notin \overrightarrow{C}$, no vertex c_j with $j \neq i$ is an in-twin or has in-arc coming from outside of $\overrightarrow{C}$, and u has an out-neighbour v with $N^-(v) = \{u\}$* **then**

4 $\quad$ Add c_i to $\mathcal{B}$

5 **if** *the cycle $\overrightarrow{C}$ has exactly one source c_i* **then**

6 $\quad$ **if** *the one sink is either c_{i-1} or c_{i+1}, and no vertex c_j with $j \neq i$ is an in-twin or has an in-arc coming from outside of $\overrightarrow{C}$* **then**

7 $\quad\quad$ Add c_{i-1} to $\mathcal{B}$

8 $\quad$ **else if** *the one sink is c_{i+k} with $k > 1$, $|\overrightarrow{C}| \geq 2k$, c_{i+k-1} (resp. c_{i+k+1}) has an out-neighbour v such that $N^-(v) = \{c_{i+k-1}\}$ (resp. $N^-(v) = \{c_{i+k+1}\}$), no vertex in $\{c_{i-1}, c_{i-2}, \ldots, c_{i+k}\}$ (resp. $\{c_{i+1}, c_{i+2}, \ldots, c_{i+k}\}$) has an in-arc, and no vertex in $\{c_{i-2}, c_{i-3}, \ldots, c_{i+k+1}\}$ (resp. $\{c_{i+2}, c_{i+3}, \ldots, c_{i+k-1}\}$) is an in-twin* **then**

9 $\quad\quad$ Add c_{i-1} (resp. c_{i+1}) to $\mathcal{B}$

10 $\quad$ **else if** *the one sink is c_{i+k} with $k > 1$, $|\overrightarrow{C}| = 2k$, c_{i+k-1} has an out-neighbour v_- such that $N^-(v_-) = \{c_{i+k-1}\}$, c_{i+k+1} has an out-neighbour v_+ such that $N^-(v_+) = \{c_{i+k+1}\}$, no vertex in $\overrightarrow{C}$ except c_i has an in-arc, no vertex in $\overrightarrow{C} \setminus \{c_i, c_{i-1}, c_{i+1}\}$ is an in-twin, and c_{i-1} and c_{i+1} are not in a set I of in-twins verifying $|I| \geq 3$* **then**

11 $\quad\quad$ Add c_{i+k} to $\mathcal{B}$

12 **if** *the cycle $\overrightarrow{C}$ has exactly two sources c_i and c_{i+2}, $|\overrightarrow{C}| = 2k$ with $k > 2$, the two sinks are c_{i+1} and c_{i+1+k}, no vertex from $\overrightarrow{C}$ except c_i and c_{i+2} is an in-twin or has an in-arc coming from outside of $\overrightarrow{C}$, there is at least one unfixable path, and there is no fixable path* **then**

13 $\quad$ Add c_{i+1} to $\mathcal{B}$

We then have to manage six specific cases (line 2). **Those special cases are handled in Algorithm 3**. The first two special cases occur when the cycle has no sink. First, if the cycle has no sink, no in-twin, and no arc coming from outside, then, we have to add one vertex of the cycle to $\mathcal{B}$ in order to maintain reachability (lines 1–2). Then, if the cycle has no sink, only one in-arc $\overrightarrow{uc_i}$ is coming from outside of it, and there is a vertex v with $N^-(v) = \{u\}$, then, we have to add either c_i or v to $\mathcal{B}$ in order to resolve them (lines 3–4).

The next three special cases occur when the cycle has one sink. First, if there is only one sink in the cycle, it is an out-neighbour of the source, and no vertex from the cycle apart from the source is an in-twin or has an in-arc coming from outside of the cycle, then we need to add one of the out-neighbours of the source in the cycle to $\mathcal{B}$ in order to resolve them (lines 6–7).

Then, there are two specific cases when the cycle has one sink, both based on the same principle. Both happen when the source is c_i, the sink is c_{i+k}, it has no in-arc, and the cycle contains at least $2k$ vertices. In the fourth special case (lines 8–9), the vertex c_{i+k-1} has an out-neighbour v verifying $N^-(v) = \{c_{i+k-1}\}$. We can see that, if no vertex in the other path from c_i to c_{i+k} (the path going through $c_{i-1}, c_{i-2}, \ldots, c_{i+k+1}$) is in $\mathcal{B}$, then, v and c_{i+k} will not be resolved. Those vertices can be added to $\mathcal{B}$ if they have an in-arc or if they are an in-twin (they will have priority). However, note that c_{i-1} might be an in-twin of c_{i+1}, in which case it should be added to $\mathcal{B}$, resolving the conflict. Hence, if none of $c_{i-1}, c_{i-2}, \ldots, c_{i+k+1}$ has an in-arc or is an in-twin, then, we can add c_{i-1} to $\mathcal{B}$ in order to resolve v and c_{i+k}. Note that, in this case, in comparison to just the sources and the resolution of sets of in-twins, we add one more vertex to $\mathcal{B}$ if c_{i-1} is the only in-twin of c_{i+1}. The same reasoning can be made with the symmetric case.

The fifth special case (lines 10–11) occurs when the cycle contains exactly $2k$ vertices and both c_{i+k-1} and c_{i+k+1} have an out-neighbour (respectively v_- and v_+) with in-degree 1: the pairs of vertices (v_-, c_{i+k}) and (v_+, c_{i+k}) might not be resolved. We can see that any in-arc or in-twin along a path from c_i to c_{i+k} will resolve c_{i+k} and the v pendant on the other path (thus either fully resolving those two pairs, or bringing us back to the previous special case), except if c_{i-1} and c_{i+1} are the only in-twins in the cycle and if they do not have another in-twin. Hence, if no vertex from the cycle except c_i has an in-arc, no vertex from the cycle except c_i, c_{i-1} and c_{i+1} is an in-twin, and c_{i-1} and c_{i+1} do not have another in-twin, then, we need to add at least one more vertex to $\mathcal{B}$ in order to resolve the two pairs of vertices, and adding c_{i+k} does exactly that.

Finally, the sixth special case is more complex (lines 12–13 and consideration in the choice of in-twins). Assume that the cycle $\overrightarrow{C}$ is of even length n, has neither in-twin nor in-arc coming from outside (except the sources), and that there are two sinks in the $\overrightarrow{C}$: one at distance 1 from the sources, and the other at the opposite end of $\overrightarrow{C}$. Now, if the first sink has spanning concerning paths, then, the second cycle and the endpoints of those concerning paths might not be resolved, since they are at the same distance ($\frac{n}{2} - 1$) of both sources of $\overrightarrow{C}$. Thus, we need to apply a strategy in order to resolve those vertices while trying to not add a supplementary vertex to $\mathcal{B}$. This is done by considering the two kinds of concerning paths, and having a priority in the selection of in-twins.

All the other cases of the cycle are already resolved through the sources and in-twins steps.

Theorem 6 (∗)**.** *Algorithm 2 computes a metric basis of an orientation of a unicyclic graph in linear time.*

4 Modular Width

In a digraph G, a set $X \subseteq V(G)$ is a *module* if every vertex not in X 'sees' all vertices of X in the same way. More precisely, for each $v \in V(G) \setminus X$ one of the

following holds: (i) $(v, x), (x, v) \in E(G)$ for all $x \in X$, (ii) $(v, x), (x, v) \notin E(G)$ for all $x \in X$, (iii) $(v, x) \in E(G)$ and $(x, v) \notin E(G)$ for all $x \in X$, (iv) $(v, x) \notin E(G)$ and $(x, v) \in E(G)$ for all $x \in X$. The singleton sets, $\emptyset$, and $V(G)$ are trivially modules of G. We call the singleton sets the *trivial modules* of G.

The graph $G[X]$ where X is a module of G is called a *factor* of G. A family $\mathcal{X} = \{X_1, \ldots, X_s\}$ is a *factorization* of G if $\mathcal{X}$ is a partition of $V(G)$, and each X_i is a module of G. If X and Y are two non-intersecting modules, then the relationship between $x \in X$ and $y \in Y$ is one of (i)-(iv) and always the same no matter which vertices x and y are exactly. Thus, given a factorization $\mathcal{X}$, we can identify each module with a vertex, and connect them to each other according to the arcs between the modules. More formally, we define the *quotient* $G/\mathcal{X}$ with respect to the factorization $\mathcal{X}$ as the graph with the vertex set $\mathcal{X} = \{X_1, \ldots, X_s\}$ and $(X_i, X_j) \in E(G/\mathcal{X})$ if and only if $(x_i, x_j) \in E(G)$ where $x_i \in X_i$ and $x_j \in X_j$. A quotient depicts the connections of the different modules of a factorization to each other while omitting the internal structure of the factors. Each factor itself can be factorized further (as long as it is nontrivial, i.e. not a single vertex). By factorizing the graph G and its factors until no further factorization can be done, we obtain a *modular decomposition* of G. The *width* of a decomposition is the maximum number of sets in a factorization (or equivalently, the maximum number of vertices in a quotient) in the decomposition. The *modular width* of G is defined as the minimum width over all possible modular decompositions of G, and we denote it by $\mathrm{mw}(G)$. An optimal modular decomposition of a digraph can be computed in linear time [21]. METRIC DIMENSION for undirected graphs was shown to be fixed parameter tractable when parameterized by modular width by Belmonte et al. [2]. We will generalize their algorithm to directed graphs.

The following result lists several useful observations.

Proposition 7 (∗). *Let $\mathcal{X} = \{X_1, \ldots, X_s\}$ be a factorization of G, and let $W \subseteq V(G)$ be a resolving set of G.*

(i) *For all $x, y \in X_i$ and $z \in X_j$, $i \neq j$, we have $\mathrm{dist}_G(x, z) = \mathrm{dist}_G(y, z)$ and $\mathrm{dist}_G(z, x) = \mathrm{dist}_G(z, y)$.*

(ii) *For all $x \in X_i$ and $y \in X_j$, $i \neq j$, we have $\mathrm{dist}_G(x, y) = \mathrm{dist}_{G/\mathcal{X}}(X_i, X_j)$.*

(iii) *For all $x, y \in V(G)$ we have either $\mathrm{dist}_G(x, y) \leq \mathrm{mw}(G)$ or $\mathrm{dist}_G(x, y) = \infty$.*

(iv) *The set $\{X_i \in \mathcal{X} \mid W \cap X_i \neq \emptyset\}$ is a resolving set of the quotient $G/\mathcal{X}$.*

(v) *For all distinct $x, y \in X_i$, where $X_i \in \mathcal{X}$ is nontrivial, we have $\mathrm{dist}_G(w, x) \neq \mathrm{dist}_G(w, y)$ for some $w \in W \cap X_i$.*

(vi) *Let $w_1, w_2 \in X_i$. If $\mathrm{dist}_G(w_1, x) \neq \mathrm{dist}_G(w_2, x)$, then $x \in X_i$ and $\mathrm{dist}_G(w_1, x) \neq \mathrm{dist}_G(w_1, y)$ or $\mathrm{dist}_G(w_2, x) \neq \mathrm{dist}_G(w_2, y)$ for each $y \notin X_i$.*

The basic idea of our algorithm (and that of [2]) is to compute metric bases that satisfy certain conditions for the factors and combine these local solutions into a global solution. We know that nontrivial modules must contain elements of a resolving set, as modules must be resolved locally (Proposition 7 (i)). While combining the local solutions of nontrivial modules, we need to make sure that a vertex $x \in X_i$, where X_i is nontrivial, is resolved from all $y \notin X_i$. If x and y are resolved as described in Proposition 7 (vi), then we need to do nothing special.

However, if $x \in X_i$ is such that $\text{dist}_G(w, x) = d$ for all $w \in W_i$ and a fixed $d \in \{1, \ldots, \text{mw}(G), \infty\}$, there might exist a vertex $y \notin X_i$ such that W_i does not resolve x and y. We call such a vertex x d-*constant* (with respect to W_i). We need to keep track of d-constant vertices and make sure they are resolved when we combine the local solutions. There are at most $\text{mw}(G) + 1$ d-constant vertices in each factor due to Proposition 7 (iii). We need to also make sure vertices in different modules that contain no elements of the solution set are resolved. To do this, we might need to include some vertices from the trivial modules in addition to the vertices we have included from the nontrivial modules.

In the algorithm presented in [2], the problems described above are dealt with by computing values $w(H, p, q)$ for every factor H, where $w(H, p, q)$ is the minimum cardinality of a resolving set of H (with respect to the distance in G) where some vertex is 1-constant iff $p = true$ and some vertex is 2-constant iff $q = true$ (for undirected graphs these are the only two relevant cases). The same values are then computed for the larger graph by combining different solutions of the factors and taking their minimum. Our generalization of this algorithm is along the same lines as the original, however, we have more boolean values to keep track of. One difference to the techniques of the original algorithm is that we do not use the auxiliary graphs Belmonte et al. use. These auxiliary graphs were needed to simulate the distances of the vertices of a factor in G as opposed to only within the factor. In our approach, we simply use the distances in G.

Theorem 8 (∗). *The metric dimension of a digraph G with $\text{mw}(G) \le t$ can be computed in time $\mathcal{O}(t^5 2^{t^2} n + n^3 + m)$ where $n = |V(G)|$ and $m = |E(G)|$.*

Proof (sketch). Let us consider one level of an optimal modular decomposition of G. Let H be a factor somewhere in the decomposition, and let $\mathcal{X} = \{X_1, \ldots, X_s\}$ be the factorization of H according to the modular decomposition. For the graph H (and its nontrivial factors $H[X_i]$) we denote by $w(H, \mathbf{p})$ the minimum cardinality of a set $W \subseteq V(H)$ such that

(i) W resolves $V(H)$ in G,
(ii) $\mathbf{p} = (p_1, \ldots, p_{\text{mw}(G)}, p_\infty)$ where $p_d = true$ if and only if H contains a d-constant vertex with respect to W.

If such a set does not exist, then $w(H, \mathbf{p}) = \infty$. In order to compute the values $w(H, \mathbf{p})$, we next introduce the auxiliary values $\omega(\mathbf{p}, I, P)$. The values $w(H[X_i], \mathbf{p})$ are assumed to be known for all $\mathbf{p}$ and nontrivial modules X_i. Let the factorization $\mathcal{X}$ be labeled so that the modules X_i are trivial for $i \in \{1, \ldots, h\}$ and nontrivial for $i \in \{h+1, \ldots, s\}$. Let $I \subseteq \{1, \ldots, h\}$ and $P = (\mathbf{p}^{h+1}, \ldots \mathbf{p}^s)^T$.

We define $\omega(\mathbf{p}, I, P) = |I| + \sum_{i=h+1}^{s} w(H[X_i], \mathbf{p}^i)$ if the conditions (a)–(d) hold. In what follows, a representative of a module X_i is denoted by x_i.

(a) The set $Z = \{X_i \in \mathcal{X} \mid i \in I \cup \{h+1, \ldots, s\}\}$ resolves the quotient $H/\mathcal{X}$ with respect to the distances in G.
(b) For $d \in \{1, \ldots, \text{mw}(G), \infty\}$ and $i \in \{h+1, \ldots, s\}$, if $p_d^i = true$, then for each trivial module $X_j = \{x_j\}$ where $j \notin I$ we have $\text{dist}_G(x_i, x_j) \ne d$ or there exists $X_k \in Z \setminus \{X_i\}$ such that $\text{dist}_G(x_k, x_i) \ne \text{dist}_G(x_k, x_j)$.

(c) For $d_1, d_2 \in \{1, \ldots, \mathrm{mw}(G), \infty\}$ and distinct $i, j \in \{h+1, \ldots, s\}$, if $p_{d_1}^i = p_{d_2}^j = true$, then $\mathrm{dist}_G(x_i, x_j) \neq d_1$, or $\mathrm{dist}_G(x_j, x_i) \neq d_2$, or there exists $X_k \in Z \setminus \{X_i, X_j\}$ such that $\mathrm{dist}_G(x_k, x_i) \neq \mathrm{dist}_G(x_k, x_j)$.

(d) For all $d \in \{1, \ldots, \mathrm{mw}(G), \infty\}$, we have $p_d = true$ (in $\mathbf{p}$) if and only if for some $i \in \{1, \ldots, h\} \setminus I$ we have $\mathrm{dist}_G(x_j, x_i) = d$ for all $X_j \in Z$, or for some $i \in \{h+1, \ldots, s\}$ we have $p_d^i = true$ and $\mathrm{dist}_G(x_j, x_i) = d$ for all $X_j \in Z \setminus \{X_i\}$.

If these conditions cannot be met, then we set $\omega(\mathbf{p}, I, P) = \infty$.

When we know all the values $\omega(\mathbf{p}, I, P)$, we can easily calculate $w(H, \mathbf{p})$ since $w(H, \mathbf{p}) = \min_{I,P} \omega(\mathbf{p}, I, P)$ (proof omitted due to lack of space).

To conclude the proof, we note that $w(G, \mathbf{p})$ is the minimum cardinality of a resolving set that gives some vertices the specific distance combinations according to $\mathbf{p}$. Thus, the metric dimension of G is $\min w(G, \mathbf{p})$ where the minimum is taken over all $\mathbf{p}$ such that $p_\infty = false$. □

5   NP-Hardness for Restricted DAGs

We now complement the hardness result from [1], which was for bipartite DAGs of maximum degree 8 and maximum distance 4.

Theorem 9 ($*$). METRIC DIMENSION *is NP-complete, even on planar triangle-free DAGs of maximum degree 6 and maximum distance 4.*

Proof (sketch). We reduce from VERTEX COVER on 2-connected planar cubic graphs, which is known to be NP-complete [23, Theorem 4.1].

Given a 2-connected planar cubic graph G, we construct a DAG G' as follows. First of all, note that by Petersen's theorem, G contains a perfect matching $M \subset E(G)$, that can be constructed in polynomial time. A planar embedding of G can also be constructed in polynomial time, so we fix one. We let $V(G') = V(G) \bigcup_{e=uv \in E(G)} \{a_e, b_e, c_e, d_e^u, d_e^v\} \bigcup_{e=uv \in M} \{f_e, g_e, h_e\}$. For every edge $e = uv$ of G, we add the arcs $\{\overrightarrow{a_e b_e}, \overrightarrow{b_e c_e}, \overrightarrow{c_e d_e^u}, \overrightarrow{c_e d_e^v}, \overrightarrow{u d_e^u}, \overrightarrow{v d_e^v}\}$. For every edge $e = uv$ of the perfect matching M of G, assuming the neighbours of u (in the clockwise cyclic order with respect to the planar embedding of G) are v, x, y and those of v are u, s, t, we arbitrarily fix one side of the edge uv to place the vertices f_e, g_e and h_e (say, on the side that is close to the edges ux and vt). We add the arcs $\{\overrightarrow{f_e g_e}, \overrightarrow{g_e c_e}, \overrightarrow{g_e h_e}, \overrightarrow{h_e u}, \overrightarrow{h_e v}, \overrightarrow{c_e c_{uy}}, \overrightarrow{c_e c_{vs}}, \overrightarrow{h_e c_{ux}}, \overrightarrow{h_e c_{vt}}\}$.

Using the embedding of G, G' can also be drawn in a planar way, it has maximum degree 6 (the vertices of type c_e are of degree 6), has no triangles, and no shortest directed path of length 4.

We claim that G has a vertex cover of size at most k if and only if G' has metric dimension at most $k + |E(G)| + |M|$ (proof omitted due to lack of space). □

6   Conclusion

METRIC DIMENSION can be solved in polynomial time on outerplanar graphs, using an involved algorithm [7]. Can one generalize our algorithms for trees and unicyclic graphs to solve METRIC DIMENSION for directed (or at least, oriented) outerplanar graphs in polynomial time? Extending our algorithm to cactus graphs already seems nontrivial.

Is METRIC DIMENSION NP-hard on planar bipartite subcubic DAGs?

Also, it would be interesting to see which hardness results known for METRIC DIMENSION of undirected graphs also hold for DAGs, or for oriented graphs.

References

1. Araujo, J., et al.: On finding the best and worst orientations for the metric dimension. Algorithmica 1–41 (2023)
2. Belmonte, R., Fomin, F.V., Golovach, P.A., Ramanujan, M.S.: Metric dimension of bounded tree-length graphs. SIAM J. Discret. Math. **31**(2), 1217–1243 (2017)
3. Blumenthal, L.M.: Theory and Applications of Distance Geometry. Oxford University Press, Oxford (1953)
4. Chartrand, G., Eroh, L., Johnson, M.A., Oellermann, O.R.: Resolvability in graphs and the metric dimension of a graph. Discret. Appl. Math. **105**(1), 99–113 (2000)
5. Chartrand, G., Raines, M., Zhang, P.: The directed distance dimension of oriented graphs. Math. Bohem. **125**, 155–168 (2000)
6. Dailly, A., Foucaud, F., Hakanen, A.: Algorithms and hardness for metric dimension on digraphs. arXiv preprint arXiv:2307.09389 (2023)
7. Díaz, J., Pottonen, O., Serna, M.J., van Leeuwen, E.J.: Complexity of metric dimension on planar graphs. J. Comput. Syst. Sci. **83**(1), 132–158 (2017)
8. Eppstein, D.: Metric dimension parameterized by max leaf number. J. Graph Algorithms Appl. **19**(1), 313–323 (2015)
9. Epstein, L., Levin, A., Woeginger, G.J.: The (weighted) metric dimension of graphs: hard and easy cases. Algorithmica **72**(4), 1130–1171 (2015)
10. Fernau, H., Heggernes, P., van't Hof, P., Meister, D., Saei, R.: Computing the metric dimension for chain graphs. Inf. Process. Lett. **115**(9), 671–676 (2015)
11. 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)
12. Galby, E., Khazaliya, L., Inerney, F.M., Sharma, R., Tale, P.: Metric dimension parameterized by feedback vertex set and other structural parameters. In: 47th International Symposium on Mathematical Foundations of Computer Science, MFCS 2022, 22–26 August 2022, Vienna, Austria. LIPIcs, vol. 241, pp. 51:1–51:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2022)
13. Gima, T., Hanaka, T., Kiyomi, M., Kobayashi, Y., Otachi, Y.: Exploring the gap between treedepth and vertex cover through vertex integrity. Theoret. Comput. Sci. **918**, 60–76 (2022)
14. Harary, F., Melter, R.A.: On the metric dimension of a graph. Ars Combin. **2**, 191–195 (1976)
15. Hartung, S., Nichterlein, A.: On the parameterized and approximation hardness of metric dimension. In: Proceedings of the 28th Conference on Computational Complexity, CCC 2013, K.lo Alto, California, USA, 5–7 June 2013, pp. 266–276. IEEE Computer Society (2013)

16. Hoffmann, S., Elterman, A., Wanke, E.: A linear time algorithm for metric dimension of cactus block graphs. Theoret. Comput. Sci. **630**, 43–62 (2016)
17. Hoffmann, S., Wanke, E.: METRIC DIMENSION for gabriel unit disk graphs Is NP-complete. In: Bar-Noy, A., Halldórsson, M.M. (eds.) ALGOSENSORS 2012. LNCS, vol. 7718, pp. 90–92. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-36092-3_10
18. Khuller, S., Raghavachari, B., Rosenfeld, A.: Landmarks in graphs. Discret. Appl. Math. **70**(3), 217–229 (1996)
19. Li, S., Pilipczuk, M.: Hardness of metric dimension in graphs of constant treewidth. Algorithmica **84**(11), 3110–3155 (2022)
20. Lobstein, A.: Watching systems, identifying, locating-dominating and discriminating codes in graphs: a bibliography (2022). https://www.lri.fr/~lobstein/debutBIBidetlocdom.pdf
21. McConnell, R.M., de Montgolfier, F.: Linear-time modular decomposition of directed graphs. Discret. Appl. Math. **145**(2), 198–209 (2005)
22. Melter, R.A., Tomescu, I.: Metric bases in digital geometry. Comput. Vision Graph. Image Process. **25**(1), 113–121 (1984)
23. Mohar, B.: Face covers and the genus problem for apex graphs. J. Combin. Theory Ser. B **82**(1), 102–117 (2001)
24. Moscarini, M.: Computing a metric basis of a bipartite distance-hereditary graph. Theoret. Comput. Sci. **900**, 20–24 (2022)
25. Oellermann, O.R., Peters-Fransen, J.: The strong metric dimension of graphs and digraphs. Discret. Appl. Math. **155**(3), 356–364 (2007)
26. Poisson, C., Zhang, P.: The metric dimension of unicyclic graphs. J. Combin. Math. Combin. Comput. **40**, 17–32 (2002)
27. Rajan, B., Rajasingh, I., Cynthia, J.A., Manuel, P.: Metric dimension of directed graphs. Int. J. Comput. Math. **91**(7), 1397–1406 (2014)
28. Sedlar, J., Škrekovski, R.: Bounds on metric dimensions of graphs with edge disjoint cycles. Appl. Math. Comput. **396**, 125908 (2021)
29. Sedlar, J., Škrekovski, R.: Vertex and edge metric dimensions of unicyclic graphs. Discret. Appl. Math. **314**, 81–92 (2022)
30. Slater, P.J.: Leaves of trees. Congressius Numer. **14**, 549–559 (1975)
31. Steiner, R., Wiederrecht, S.: Parameterized algorithms for directed modular width. In: Changat, M., Das, S. (eds.) CALDAM 2020. LNCS, vol. 12016, pp. 415–426. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-39219-2_33