Journal of Computer and System Sciences 158 (2026) 103776

Contents lists available at ScienceDirect

Journal of Computer and System Sciences

journal homepage: www.elsevier.com/locate/jcss

Algorithms and hardness for Metric Dimension on digraphs [☆]

Antoine Dailly [a,*], Florent Foucaud [a], Anni Hakanen [b,c,1]

[a] Université Clermont Auvergne, CNRS, Clermont Auvergne INP, Mines Saint-Étienne, LIMOS, 63000 Clermont-Ferrand, France
[b] Department of Mathematics and Statistics, University of Turku, FI-20014, Finland
[c] Turku Collegium for Science, Medicine and Technology TCSMT, Finland

ARTICLE INFO

Article history:
Received 25 July 2023
Received in revised form 12 March 2025
Accepted 19 January 2026
Available online 3 February 2026

Keywords:
Graphs
Digraphs
Directed graphs
Metric dimension
Algorithms
Fixed-parameter tractability
Parameterized complexity
Complexity
Trees
Modular-width

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 (non-trivially) extends a previous algorithm for oriented trees. We then extend the method to orientations of unicyclic graphs. 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 [19], 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 [18].

More formally, let us denote by $\text{dist}(x, y)$ the distance from x to y in a digraph. Here, the distance $\text{dist}(x, y)$ 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

☆ Work financed by the French government IDEX-ISITE initiative CAP 20-25 (ANR-16-IDEX-0001), the International Research Center "Innovation Transportation and Production Systems" of the I-SITE CAP 20-25, and by the ANR project GRALMECO (ANR-21-CE48-0004).

* Corresponding author.
E-mail address: antoine.dailly@inrae.fr (A. Dailly).

[1] Research supported by the Jenny and Antti Wihuri Foundation and partially by Research Council of Finland grant number 338797.

https://doi.org/10.1016/j.jcss.2026.103776

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.[2]

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 non-trivial 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 [20]. On the positive side, it can easily be solved in linear time on trees [4,14,19,30]. 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], and 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 the 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 (*i.e.*, replace every edge of the undirected graph by a directed 2-cycle).

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].

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 non-trivial. 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.

A short preliminary version of this paper has appeared in the proceedings of the WG 2023 conference [6].

2. Digraphs whose underlying graph is a tree (di-trees)

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 [4,14,19,30] 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-neighbor) 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-neighborhood). 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, for which we need the following notion:

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 $e_1, \dots, e_k$ ($k \geq 2$);
2. there is a unique vertex $y \notin E$ such that the arc $\overrightarrow{ye_1}$ exists and for all $i \in \{2, \dots, k\}$, e_i has no in-neighbors from outside E;
3. for every $i \in \{1, \dots, k-1\}$, no arc $\overrightarrow{e_iz}$ with $z \notin E$ exists.

An example of an escalator is depicted on Fig. 1b. Note that there can be any number (possibly, zero) of vertices $z \notin E$ such that the arc $\overrightarrow{e_kz}$ exists.

[2] 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.

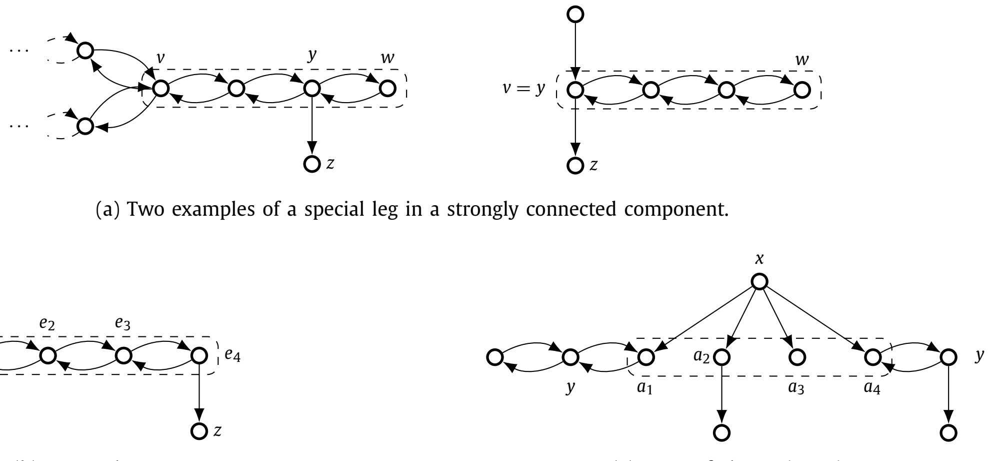

v
y
w
z
v = y
w
z
(a) Two examples of a special leg in a strongly connected component.
y
e_2
e_3
e_1
e_4
z
(b) An escalator.
x
a_2
y
y
a_1
a_3
a_4
(c) A set of almost-in-twins.

Fig. 1. Illustrations of Definitions 1 to 3. The vertex names are taken from those definitions.

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

1. for every $i \in \{1, \dots, k\}$, the arc $\overrightarrow{xa_i}$ exists and the arc $\overrightarrow{a_ix}$ does not exist;
2. for every $i \in \{1, \dots, k\}$, 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 neighbor in the escalator.

An example of a set of almost-in-twins is depicted on Fig. 1c. 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 $z \notin C$ and $N^-(z) = \{y\}$.

An example of a special leg is depicted on Fig. 1a. Note that several special legs can span from the same vertex, from which regular legs can also span. Algorithm 1, illustrated in Figs. 2 and 3, 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. 2a, 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.

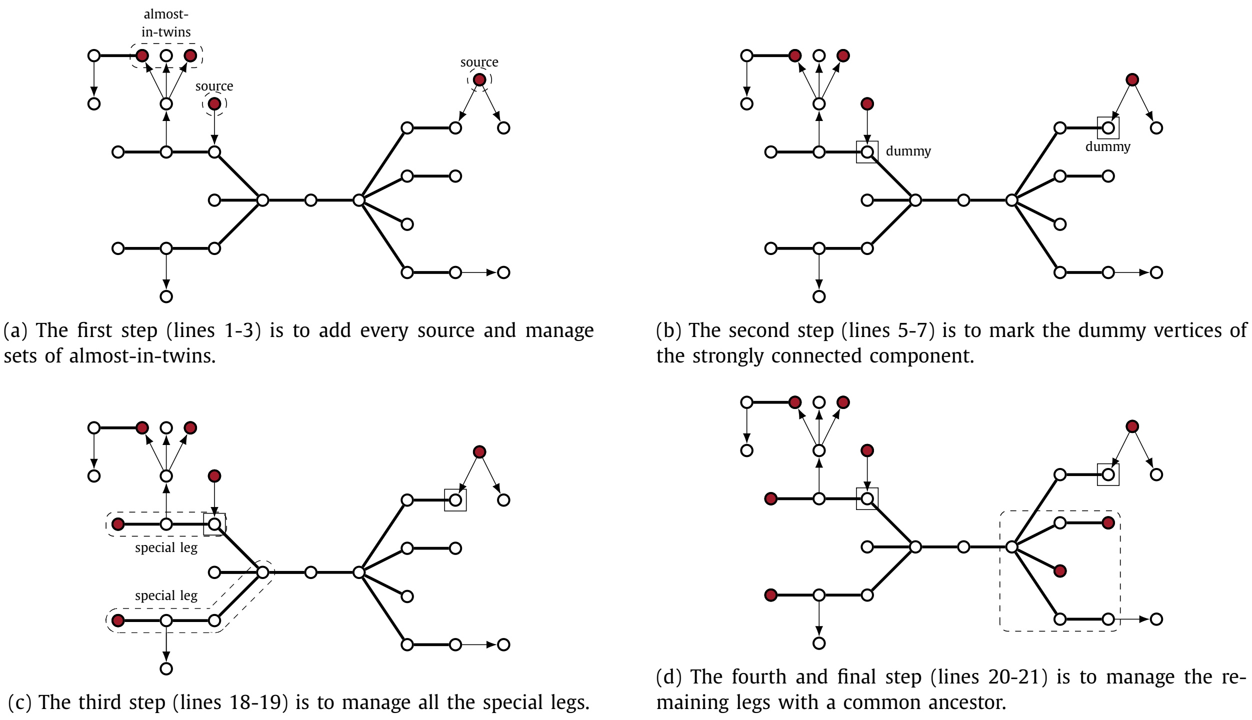
For each strongly connected component having C as an underlying graph, we first identify each vertex x of C that has an in-arc coming from **outside** C. Indeed, since x is the "last" vertex of a path coming from outside C, there are vertices of $\mathcal{B}$ "behind" this in-arc (or they can themselves 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 Fig. 2b): 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 will be highlighted in the proof, and are depicted in Fig. 3.

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 neighbor 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

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 arc $\overrightarrow{uv}$ such that $u \in C$, $v \notin C$ and $N^-(v) = \{u\}$ then
11                 Add $x$ to $\mathcal{B}$
12             else if the only arcs $\overrightarrow{uv}$ such that $u \in C$, $v \notin C$ and $N^-(v) = \{u\}$ are such that $u = x$ (resp. $u = y$) 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 arc $\overrightarrow{wz}$ such that $z \notin C$ and $N^-(z) = \{w\}$ 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 $\geq 3$ 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 $\mathcal{B}$
```



(a) The first step (lines 1-3) is to add every source and manage sets of almost-in-twins.

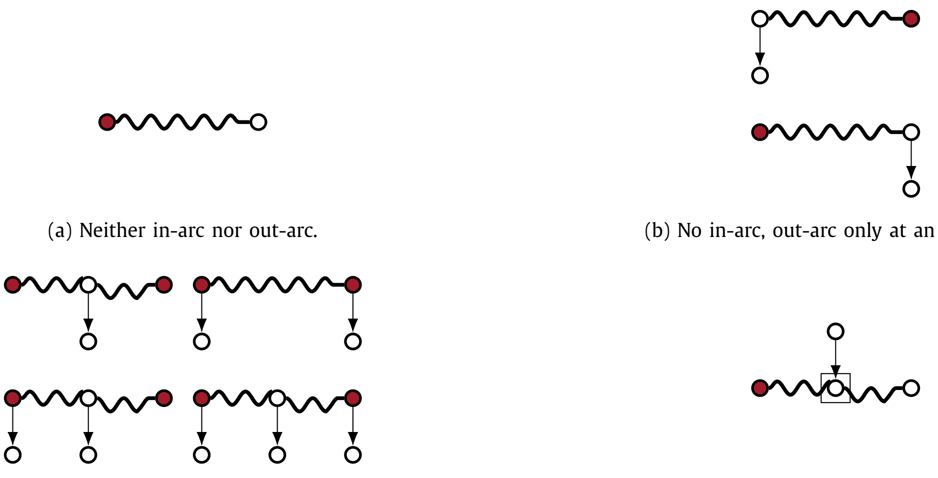
(b) The second step (lines 5-7) is to mark the dummy vertices of the strongly connected component.

(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. 2. 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 filled in red.

Fig. 2c). Finally, we apply the well-known algorithm for computing the metric basis of a tree to the remaining parts of C (lines 20-21, depicted in Fig. 2d). The special legs and the legs containing a dummy vertex, being already resolved, are not considered in this part.


(a) Neither in-arc nor out-arc.

(b) No in-arc, out-arc only at an endpoint.

(c) No in-arc, other cases.

(d) One in-arc, in the middle.

Fig. 3. The cases of Algorithm 1 where a strongly connected component is a path and we have to add specific vertices (filled in red) to the metric basis. The path is depicted as wavy bolded edges. Note that the considered out-arcs have to be towards a vertex with no other in-neighbor than the one in the path (out-arcs to vertices with other in-neighbors can still exist).

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

Proof. Let T be a di-tree, and $\mathcal{B}$ be the set of vertices returned by Algorithm 1. We need to prove that $\mathcal{B}$ resolves every pair of vertices of T, that $\mathcal{B}$ is of minimum size, and that Algorithm 1 runs in linear time.

First, note that, if T is either an orientation of a tree or a strongly connected graph (and thus seen as an undirected graph), then, Algorithm 1 does compute a metric basis. In the first case, $\mathcal{B}$ will contain only the vertices added in lines 1-3 (which correspond to a so-called *adequate* set of vertices in [1], see Lemma 2.10 and Theorem 2.11); and in the second case, $\mathcal{B}$ will contain either only a vertex added in lines 10-11 (if T is a path) or only the vertices added in lines 20-21 (which correspond to the well-known resolving set of trees, see for example Theorem 2.4 in [19]); those two cases do form a metric basis of T.

Hence, we will consider that T contains at least one strongly connected component and at least one non-strongly connected component. We will first show that the vertices we select in $\mathcal{B}$ are necessary to resolve at least one pair of vertices, and then that they do indeed form a resolving set, and thus that $\mathcal{B}$ is a metric basis.

First, let us consider the sources. It is easy to see that each source has to be in $\mathcal{B}$, since otherwise they would not be reachable from any other vertex in $\mathcal{B}$. Now, let us consider the in-twins. Again, it is easy to see that the only way to resolve two in-twins will be to have at least one of them in $\mathcal{B}$. However, Algorithm 1 considers *almost-in-twins*, which are more general than regular in-twins. This is because of the *escalators*: let u be the endpoint of an escalator having an in-arc coming from outside of it. If u has an almost-in-twin v, then, by definition, u and v cannot be resolved without taking in $\mathcal{B}$ either at least one of these or another vertex from the escalator. Thus, we choose to take either u or v, which will resolve vertices in the escalator as well as the almost-in-twins.

Note that sources and in-twins can only exist in a non-strongly connected component. Hence, in the rest of this part of the proof, we will consider a strongly connected component with C as an underlying graph. Note that, by our construction, if a vertex $x \in C$ is a dummy vertex, then, there will be a vertex $b \in \mathcal{B}$ such that there is a path from b to x (even if it is not necessarily the case yet).

We first consider the case of the special legs of C. Let L be a special leg with vertices $x_1, \dots, x_\ell$, starting from a vertex x_1 such that $d_C(x_1) \geq 3$ or x_1 is a dummy vertex, and ending at a vertex x_ℓ verifying $d_C(x_\ell) = 1$. Let x_i ($i \in \{1, \dots, \ell\}$) be a vertex such that $d_C(x_i) \geq 2$ and there is an arc $\overrightarrow{x_i y}$ with $y \notin C$ and $N^-(y) = \{x_i\}$. Note that, at this point, no vertex in the special leg can be in $\mathcal{B}$, and there is no arc from outside of C to a vertex of L apart from x_1. This means that, for vertices of L, x_1 can be seen as a representative from the set $\mathcal{B}$: every path from a vertex in $\mathcal{B}$ to a vertex in L has to go through x_1. Furthermore, y, having no other in-neighbor, is only reached by vertices of $\mathcal{B}$ through x_1. In practice, this means that x_{i+1} and y are not resolved: since they are at the same distance from x_1, they are at the same distance from any vertex in $\mathcal{B}$. Hence, it is necessary to add either a vertex from the set $\{x_{i+1}, \dots, x_\ell\}$ or y to $\mathcal{B}$ in order to resolve this pair. Since other such situations might occur in L, the easiest solution to resolve this pair is to add x_ℓ to $\mathcal{B}$, since doing so will resolve all such pairs.

We then consider the rest of C. Since T is a di-tree, C is a tree. Let L_1 and L_2 be two (non-special) legs spanning from a vertex x, and assume that neither of those legs have an in-arc coming from outside of C. If no vertex from either L_1 or L_2 is in $\mathcal{B}$, then, the vertices from L_1 and L_2 cannot be resolved. Here, we add the endpoint of either L_1 or L_2 to $\mathcal{B}$. Note that, since C is not a path (which either is a special case that we will consider below, or has been considered in the special leg case), if one of L_1 or L_2 has a dummy vertex or a vertex in $\mathcal{B}$, then, the vertices from L_1 and L_2 will be resolved without having to add another vertex in $\mathcal{B}$.

However, note that there are cases where the above method does not create a resolving set. Indeed, when C is a path with vertices $x_1, \dots, x_k$, it is possible to fall into some patterns where we need to add specific vertices to $\mathcal{B}$. The patterns are the following:

1. There is no arc $\overrightarrow{yx_i}$ with $y \notin C$, in which case we have the following subpatterns depending on the presence of out-arcs $\overrightarrow{x_iy}$ with $y \notin C$ and $N^-(y) = \{x_i\}$:

   (a) there is no such arc $\overrightarrow{x_iy}$, in which case we have to add either x_1 or x_k to $\mathcal{B}$ (the vertices of C have to be reached from $\mathcal{B}$ and separated);
   (b) there is such an arc and we have $u = x_1$ (resp. $u = x_k$), in which case we have to add a vertex from C to $\mathcal{B}$ in order to reach the vertices of C; we add x_k (resp. x_1) to $\mathcal{B}$ since, otherwise, either y and x_2 (resp. x_{k-1}) will not be resolved or some pair of vertices in the path will not be resolved;
   (c) in every other case (that is, there are such arcs leaving C from different vertices), we have to add both x_1 and x_k to $\mathcal{B}$ (since, otherwise, one vertex from C and one vertex reached by an out-arc from C will not be resolved).

   Note that there can be arcs $\overrightarrow{x_iy}$ with $y \notin C$ and where y has an in-neighbor not in C; those do not matter for this specific analysis, since such y are already resolved from the vertices in C by vertices in $\mathcal{B}$ that are ancestors of their other in-neighbors.
2. There is exactly one x_i, $i \notin \{1, k\}$, such that there exists an arc $\overrightarrow{yx_i}$ with $y \notin C$ and there is no arc $\overrightarrow{x_iz}$ with $z \notin C$ and $N^-(z) = \{x_i\}$, in which case we add either endpoint to $\mathcal{B}$ (since, otherwise, x_{i-1} and x_{i+1} will not be resolved). Note that if such an arc $\overrightarrow{x_iz}$ exists, then we have two special legs spanning from x_i and thus no problem arises.

In every other case where C is a path, the cases considered above (in particular, the special leg and the sources and almost-in-twins) will have us add to $\mathcal{B}$ the vertices necessary to resolve the vertices of C and its direct out-neighborhood.

At this point, every vertex that we added to $\mathcal{B}$ was necessary to resolve at least one pair of vertices which could not be resolved any other way. At each step, when we had the choice between several vertices, we chose as few as possible to resolve everything. Hence, we only need to check that every pair of vertices is resolved by $\mathcal{B}$, which will prove that $\mathcal{B}$ is a metric basis.

Assume by contradiction that two vertices u and v are not resolved by $\mathcal{B}$. This means that, for every vertex $b \in \mathcal{B}$, either both u and v are not reachable from b, or there are two unique shortest paths P_u^b between b and u and P_v^b between b and v such that $|P_u^b| = |P_v^b|$. Note that P_u^b and P_v^b are unique because the underlying graph of T is a tree. Given a vertex $b \in \mathcal{B}$, let x be the last common vertex of P_u^b and P_v^b (note that we may have $x = b$). Note that we can consider u and v to be the out-neighbors of x on P_u^b and P_v^b, since if those out-neighbors were resolved by any vertex $b' \in \mathcal{B}$ then u and v would be resolved by b' too, a contradiction. Now, u and v cannot be in-twins, since, otherwise, one of them would be in $\mathcal{B}$, a contradiction. Thus, one vertex, say without loss of generality u, has an in-neighbor w that is not an in-neighbor of v. There are now several cases to consider.

If there is no arc $\overrightarrow{uw}$, then, w has to be reachable from a vertex $b' \in \mathcal{B}$. However, since the underlying graph of T is a tree, the only path from b' to v (which necessarily exists, since otherwise b' resolves u and v since u is reachable from b', a contradiction) goes through u, and thus $\text{dist}(b', v) = \text{dist}(b', u) + 2$, and thus b' resolves u and v, a contradiction.

Hence, w and u are in a common strongly connected component C. First, assume that the arc $\overrightarrow{ux}$ does not exist, that is, v is not reachable from u. Note that if any vertex from C is in $\mathcal{B}$, then, u and v will be resolved, a contradiction. Now, there are only a limited number of cases where no vertex from C has been added to $\mathcal{B}$. In all those cases, the underlying graph of C is a path, and they are the cases that were considered neither in lines 8-17 of Algorithm 1 nor in the special legs case. The first case is if C is an escalator. The only possibility for this, by definition, is that u and v are almost-in-twins (since, otherwise, there would be another in-arc than the one at one endpoint), in which case, either u or v are added in $\mathcal{B}$, and thus they are resolved, a contradiction. The other case is if either both endpoints of C have in-arcs coming from outside of C, or if the in-arcs coming from outside of C the closest to the endpoints are not followed by any out-arc from the same vertex. However, note that, in this case, there is at least one vertex $b' \in \mathcal{B}$ such that u is reachable from b' but v is not (there is necessarily at least one vertex in $\mathcal{B}$ “behind” every in-arc of a strongly connected component), a contradiction.

This implies that the arc $\overrightarrow{ux}$ exists, and thus x also belongs to C. First, if any vertex $y \in C$ such that $\text{dist}(y, u) \neq \text{dist}(y, v)$ is in $\mathcal{B}$, then, u and v are resolved, a contradiction. Hence, there are only two possibilities: either no vertex from C is in $\mathcal{B}$, or vertices in $C \cap \mathcal{B}$ are either x or in a part of C that can only reach u and v through x. As in the previous case, if no vertex from C is in $\mathcal{B}$, then, we reach a contradiction. Indeed, C cannot be an escalator: either x is an endpoint of C and then $\overrightarrow{xv}$ is an out-arc that prevents C from being an escalator, or it is not an endpoint and then the in-arc arriving at x from P_u^b ($x \neq b$ since no vertex from C is in $\mathcal{B}$) prevents C from being an escalator. In the other cases, the underlying graph of C is a path with specific properties (either both endpoints have an in-arc coming from outside, or the two in-arcs coming from the outside the closest to the endpoints do not have out-arcs leaving C from the same vertex), and the in-arcs coming from outside of C will allow u and v to be resolved, a contradiction.

Thus, there is at least one vertex $b' \in C \cap \mathcal{B}$, and it can only reach u and v through x. This implies that u and w are in a leg of C with no in-arc from outside of C (since, otherwise, we would have added a vertex in $\mathcal{B}$ on this side of C, which would resolve u and v, a contradiction). There are three cases. First, if there is an in-arc from outside of C to x and $v \notin C$, then u and w are in a special leg, and thus its endpoint would be in $\mathcal{B}$ and would resolve u and v, a contradiction. Now, if there is no in-arc from outside of C to x and $v \notin C$, then either u and w are in a special leg (a contradiction, like before), or the underlying graph of C is a path, and our construction would have added the endpoint of C on the side of u to $\mathcal{B}$, which would resolve u and v, a contradiction. Finally, if $v \in C$, then, v has to be in a regular leg (since, otherwise, we would have added a vertex in $\mathcal{B}$ on this side of C, which would resolve u and v, a contradiction), but now either the underlying graph of C is a path and it would have an in-arc or we would have added one of its endpoints to $\mathcal{B}$, resolving u and v, a contradiction; or u and v are the first vertices in two regular legs of C spanning from the same vertex x and we would have added one of the endpoints of the legs to $\mathcal{B}$, resolving u and v, a contradiction.

Thus, by our construction, every pair of vertices is resolved, and thus $\mathcal{B}$ is a resolving set. This proves that $\mathcal{B}$ is a metric basis, and thus that Algorithm 1 is correct.

Finally, it is easy to see that Algorithm 1 computes $\mathcal{B}$ in linear time, which proves the statement of Theorem 4. □

In [1], the authors used the notion of *removable source* to characterize orientations of trees with a weak metric dimension[3] lower than the metric dimension. In the case of di-trees, however, the definition of removable source is not so clear-cut. Indeed, there are several cases where a source meets (resp. does not meet) the conditions of the removable source as defined in [1] and yet cannot (resp. can) be removed from a metric basis in order to obtain a weak metric basis. While we do not have a proper definition of a removable source in the context of di-trees, we get the following result:

Proposition 5. *There is a polynomial-time algorithm computing a weak metric basis of a di-tree.*

Proof. The result comes from two facts. The first is that every vertex we added to the metric basis in Algorithm 1 was necessary to either guarantee that every vertex is reached from the basis (sources) or "locally" resolve some pairs of vertices (almost-in-twins, special legs...), which still need to be resolved, and thus we cannot avoid adding those second ones to the weak metric basis either. The second is that the only possible *infinite-vertex* (that is, a vertex that is not reachable from any vertex in the basis) in a directed graph is a source (**Proposition 2.2** in [1]). Indeed, if the infinite-vertex s is not a source, then, any vertex u such that there is a path from u to s cannot be reached from any vertex in the basis, and thus we would have several infinite-vertices, a contradiction.

Hence, in a di-tree, the only possible way to have a weak metric basis is to remove a source from a metric basis without creating a pair of non-resolved vertices. This is possible in polynomial time, since the actualization of distance vectors in a di-tree will take 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 of a unicyclic graph in terms of the metric dimension of a tree obtained 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 6. 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 $\overrightarrow{P}$ a *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 $\overrightarrow{P} \cup \overrightarrow{C}$.

[3] In which one vertex may be unreachable from any resolving set vertex.

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 $\mathcal{B}$
```

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     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-neighbor $v$ with $N^-(v) = \{u\}$ then
4     Add $c_i$ to $\mathcal{B}$
5 if the cycle $\overrightarrow{C}$ has exactly one source $c_i$ then
6     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         Add $c_{i-1}$ to $\mathcal{B}$
8     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-neighbor $v$ 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         Add $c_{i-1}$ (resp. $c_{i+1}$) to $\mathcal{B}$
10    else if the one sink is $c_{i+k}$ with $k > 1$, $|\overrightarrow{C}| = 2k$, $c_{i+k-1}$ has an out-neighbor $v_-$ such that $N^-(v_-) = \{c_{i+k-1}\}$, $c_{i+k+1}$ has an out-neighbor $v_+$ 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        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    Add $c_{i+1}$ to $\mathcal{B}$
```

Furthermore, if, for every vertex in $\overrightarrow{P}$ belonging to a non-empty 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*.

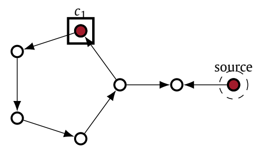
Finally, a vertex might belong both to an unfixable path and to a fixable path; in this case, the fixable path takes precedence (*i.e.*, we will consider that the vertex belongs to the fixable path).

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 result on several cases is depicted in Figs. 4 and 5. 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, as depicted in Fig. 5.

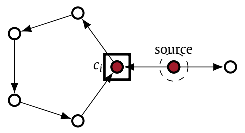
We then have to manage six specific cases (line 2). **Those special cases are handled in Algorithm 3, to which the line numbers in the next five paragraphs will refer.** The first two special cases occur when the cycle has no sink. First, if the cycle has no sink, no in-twin, and no arc coming from outside, then, we have to add one vertex of the cycle to $\mathcal{B}$ in order to maintain reachability (lines 1-2, depicted in Fig. 4a). 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 add c_i or v to $\mathcal{B}$ in order to resolve them (lines 3-4, depicted in Fig. 4b).

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-neighbor 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-neighbors of the source in the cycle to $\mathcal{B}$ in order to resolve them (lines 6-7), depicted in Fig. 4c).

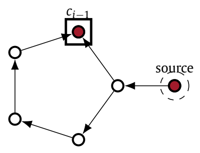
Then, there are two specific cases when the cycle has one sink, both based on the same principle. Both happen when the source is c_i, the sink is c_{i+k}, it has no in-arc, and the cycle contains at least $2k$ vertices. In the fourth special case



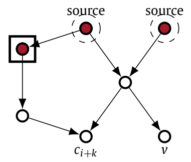
(a) First special case: there is neither source, nor in-twin or in-arc coming from outside in $\overrightarrow{C}$.



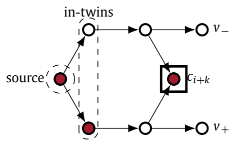
(b) Second special case: there is no source or in-twin in the cycle, and there is a unique in-arc $\overrightarrow{uc_i}$ and a vertex v with $N^-(v) = \{u\}$.



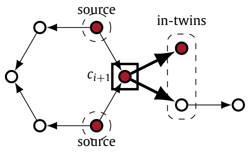
(c) Third special case: there is one source c_i in $\overrightarrow{C}$, the sink is either c_{i-1} or c_{i+1}, and there is neither in-twin nor in-arc in $\overrightarrow{C} \setminus \{c_i\}$.



(d) Fourth special case: $|\overrightarrow{C}| \geq 2k$, there is one source c_i in $\overrightarrow{C}$, the sink is c_{i+k}, c_{i+k-1} (w.l.o.g.) has an out-neighbor v with in-degree 1, no vertex in $\{c_{i-1}, c_{i-2}, \ldots, c_{i+k}\}$ has an in-arc, and no vertex in $\{c_{i-2}, c_{i-3}, \ldots, c_{i+k+1}\}$ is an in-twin.



(e) Fifth special case: $|\overrightarrow{C}| = 2k$, there is one source c_i in $\overrightarrow{C}$, the sink is c_{i+k}, c_{i+k-1} and c_{i+k+1} both have out-neighbors v_- and v_+ with in-degree 1, no vertex in $\overrightarrow{C} \setminus \{c_i\}$ 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 of in-twins of size at least 3.



(f) Sixth special case: there are two sources c_i and c_{i+2} in $\overrightarrow{C}$, the sinks are c_{i+1} and $c_{i+1+\frac{n}{2}}$, there is neither in-twin nor in-arc in $\overrightarrow{C} \setminus \{c_i, c_{i+2}\}$, and there is at least one unfixable path (in bolded arcs) and no fixable path.

Fig. 4. The special cases of Algorithm 2, managed in Algorithm 3, where we have to add to $\mathcal{B}$ one more vertex other than the sources and the resolution of sets of in-twins. Vertices in $\mathcal{B}$ are filled in red, and the supplementary vertex added to $\mathcal{B}$ is identified by a square around it.

(lines 8-9), depicted in Fig. 4d), the vertex c_{i+k-1} has an out-neighbor 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.

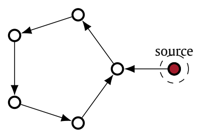
The fifth special case (lines 10-11), depicted in Fig. 4e), occurs when the cycle contains exactly $2k$ vertices and both c_{i+k-1} and c_{i+k+1} have an out-neighbor (respectively v_- and v_+) with in-degree 1: the pairs of vertices (v_-, c_{i+k}) and (v_+, c_{i+k}) might not be resolved. We can see that any in-arc or in-twin along a path from c_i to c_{i+k} will resolve c_{i+k} and the pendent v 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, depicted in Fig. 4f, and consideration in the choice of in-twins, depicted in Fig. 5g). 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 $\overrightarrow{C}$: 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 sink and the endpoints of those concerning paths might not be resolved, since they are at the same distance $(\frac{n}{2} - 1)$ from 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. The details will be in the proof.

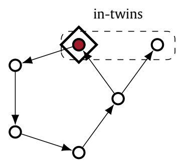
All the other cases of the cycle are already resolved through the sources and in-twins steps.

Theorem 7. *Algorithm 2 computes a metric basis of an orientation of a unicyclic graph in linear time.*

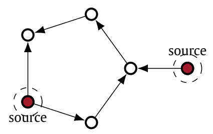
Proof. Let $\overrightarrow{U}$ be an orientation of a unicyclic graph with cycle $\overrightarrow{C}$. First, note that sources and $|I| - 1|$ of each set I of in-twins must be in the metric basis $\mathcal{B}$. Furthermore, when a vertex of a set I of in-twins is in $\overrightarrow{C}$, then, it should be prioritized for reachability reasons. However, this is not sufficient to obtain a metric basis, since either some vertices may be unreachable or some pairs of vertices may be non-resolved, which is why we will need the six special cases. In those cases, we will either have to give stronger priority to some in-twins, or have to add one more vertex to $\mathcal{B}$.



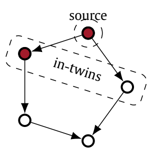
(a) No source in the cycle, but there is an in-arc and we are not in the second special case.



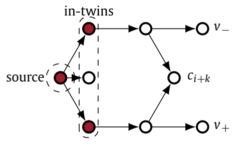
(b) No source in the cycle, but there is an in-twin, which has priority.



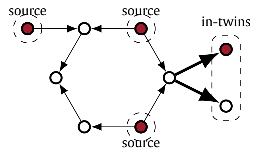
(c) A source c_i, a sink either c_{i-1} or c_{i+1}, but an in-arc coming from outside.



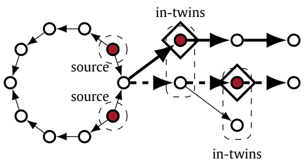
(d) A source c_i, and a sink that is neither c_{i-1} nor c_{i+1}.



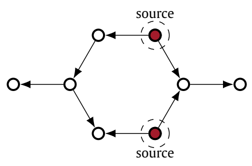
(e) $|\overrightarrow{C}| = 2k$, there is one source c_i in $\overrightarrow{C}$, the sink is c_{i+k}, c_{i+k-1} and c_{i+k+1} both have out-neighbors v_- and v_+ with in-degree 1, no vertex in $\overrightarrow{C} \setminus \{c_i\}$ has an in-arc, no vertex in $\overrightarrow{C} \setminus \{c_i, c_{i-1}, c_{i+1}\}$ is an in-twin, but c_{i-1} and c_{i+1} are in a set of in-twins of size at least 3.



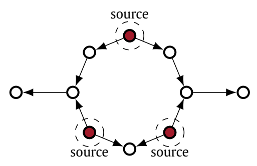
(f) There are two sources c_i and c_{i+2} in $\overrightarrow{C}$, the sinks are c_{i+1} and $c_{i+1+\frac{n}{2}}$, and there is at least one unfixable path (in bolded arcs) and no fixable path, but there is an in-arc coming from outside of $\overrightarrow{C}$.



(g) There are two sources c_i and c_{i+2} in $\overrightarrow{C}$, the sinks are c_{i+1} and $c_{i+1+\frac{n}{2}}$, there is neither in-twin nor in-arc in $\overrightarrow{C} \setminus \{c_i, c_{i+2}\}$, and there is at least one unfixable path (in bolded arcs), but there is at least one fixable path (in bolded and dashed arcs).



(h) Two sources, not in the special case configuration.



(i) More than two sources.

Fig. 5. The standard cases of Algorithm 2, when a metric basis $\mathcal{B}$ of an orientation of a unicyclic graph contains every source and resolves every set of in-twins (with some priority, identified with a diamond). Vertices in $\mathcal{B}$ are filled in red.

The first case is if vertices in $\overrightarrow{C}$ are not reachable from any source or in-twin. This is only possible if $\overrightarrow{C}$ contains no sink, no in-arc is coming from outside of $\overrightarrow{C}$, and no vertex of $\overrightarrow{C}$ is an in-twin. In this case, we need to add any vertex from $\overrightarrow{C}$ to $\mathcal{B}$.

The second case is if there is exactly one in-arc $\overrightarrow{uc_i}$ with $u \notin \overrightarrow{C}$, the cycle has no sink, no c_j with $j \neq i$ is an in-twin or has an in-arc coming from outside of $\overrightarrow{C}$, and there is a vertex v verifying $N^-(v) = \{u\}$. In this case, v and c_i are not resolved, and thus we need to add at least one of them to $\mathcal{B}$.

The third case is if there is only one source c_i in $\overrightarrow{C}$, the 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 any in-arc coming from outside of $\overrightarrow{C}$. In this case, the out-neighbors of c_i are not in-twins but cannot be resolved without taking at least one of them into $\mathcal{B}$.

The fourth and fifth case are linked. In both cases, the cycle $\overrightarrow{C}$ contains at least $2k$ vertices, one source c_i, and one sink c_{i+k} with no in-arc. The problem will be when an in-neighbor of the sink has an out-neighbor v with in-degree 1: v and c_{i+k} might not be resolved. Let us see when this can happen.

In the fourth case, either c_{i+k-1} or c_{i+k+1} has one out-neighbor v with in-degree 1. Without loss of generality, we will consider that it is c_{i+k-1}. Assume furthermore that no vertex from $\{c_{i-1}, c_{i-2}, \ldots, c_{i+k+1}\}$ has an in-arc, and no vertex from $\{c_{i-2}, c_{i-3}, \ldots, c_{i+k+1}\}$ is an in-twin (since, otherwise, v and c_{i+k} would be resolved). Now, whether c_{i-1} is an in-twin of c_{i+1} or not, we add c_{i-1} to $\mathcal{B}$, resolving v and c_{i+k}. Note that c_{i-1} could have been added to $\mathcal{B}$ at the in-twin step, but we ensure that it is added in order to resolve the two conflicting vertices.

In the fifth case, $\overrightarrow{C}$ contains exactly $2k$ vertices, and both c_{i+k-1} and c_{i+k+1} have out-neighbors v_- and v_+, respectively, with in-degree 1. Assume furthermore that no vertex in $\overrightarrow{C} \setminus \{c_i\}$ has an in-arc, and that no vertex in $\overrightarrow{C} \setminus \{c_i, c_{i-1}, c_{i+1}\}$ is an in-twin. Now, if c_{i-1} and c_{i+1} are in-twins and have another in-twin, then, they will both be added to $\mathcal{B}$, and the pairs

(v_-, c_{i+k}) and (v_+, c_{i+k}) are resolved. Otherwise, at least one of v_- and v_+ will remain non-resolved with c_{i+k}, and thus we add c_{i+k} to $\mathcal{B}$ in order to resolve the two pairs with the addition of just one vertex.

Finally, for the sixth case, assume that $\overrightarrow{C}$ is of even length and contains two sources c_i and c_{i+2}, that the two sinks are at equal distance of the sources (so they are c_{i+1} and $c_{i+1+\frac{n}{2}}$), and that there is neither in-twin nor in-arc coming from outside of $\overrightarrow{C}$ (except the sources themselves). Now, if there are concerning paths, then, their endpoints and $c_{i+1+\frac{n}{2}}$ might not be resolved, since they are all at distance $\frac{n}{2}-1$ from the sources. Hence, we have to pick carefully among the potential sets of in-twins, and we may need to add another vertex to $\mathcal{B}$. There are three cases to cover. First, if all the concerning paths are fixable paths, then, by prioritizing the in-twins that are in the fixable paths, the endpoints and $c_{i+1+\frac{n}{2}}$ will be resolved without having to add another vertex in $\mathcal{B}$. Now, if all the concerning paths are unfixable paths, then, every in-twin along the concerning paths belongs to a concerning path, and thus we will need to add a supplementary vertex to $\mathcal{B}$ in order to resolve the endpoints and the sink $c_{i+1+\frac{n}{2}}$; here, we choose the sink c_{i+1}. Finally, if there are both unfixable paths and fixable paths, then, we can resolve the endpoints and the second sink by having the following priority on in-twins: those on unfixable paths followed by those on fixable paths followed by those on non-concerning paths. Doing this will guarantee that the endpoints and $c_{i+1+\frac{n}{2}}$ are resolved. This settles the sixth and last special case.

We will now prove that the vertices we added to $\mathcal{B}$ (sources, in-twins, and the six special cases) do form a resolving set. Since they were necessary to add, this will prove that $\mathcal{B}$ is indeed a metric basis.

First, note that every vertex in $\overrightarrow{U}$ is reachable from some vertex in $\mathcal{B}$. Furthermore, recall that Algorithm 2 gives priority to vertices in the cycle when resolving a set of in-twins, which will be important in some parts of the proof: if we know that the cycle contains an in-twin, we know that it will be in $\mathcal{B}$.

Now, assume by contradiction that two vertices u and v are not resolved by $\mathcal{B}$. Since they are reachable, there is a vertex $b \in \mathcal{B}$ such that there are paths P_u^b and P_v^b from b to u and v, respectively. Let x be the last common vertex of P_u^b and P_v^b (we can have $x = b$), and we can assume that every pair of predecessors of u and v is resolved (since, otherwise, we can just take the first unresolved pair of vertices on both paths). There are two cases to consider:

1. $u, v \in N^+(x)$. Since u and v are not resolved, they cannot be in-twins (since, otherwise, one of them would be in $\mathcal{B}$, a contradiction), and hence there is a vertex w such that, without loss of generality, the arc $\overrightarrow{wu}$ exists and the arc $\overrightarrow{wv}$ does not exist. Now, w has to be reachable from a vertex in $\mathcal{B}$, so there are two more possibilities.
   First, assume that there is a path from x to w. There are two subcases here. In the first subcase, u, w and x are in the cycle $\overrightarrow{C}$ (of which x is the only source and u is the only sink). Then, whether v is also in the cycle or not, either there is an in-twin or an in-arc coming from outside of $\overrightarrow{C}$ along the cycle, which would resolve u and v, a contradiction; or we are in the third special case considered in Algorithm 2, and thus u and v are resolved, a contradiction. In the second subcase, the path from x to w goes through u, and thus u and w are both in the cycle $\overrightarrow{C}$ (which has no sink). Then, either there is an in-arc reaching v or a vertex in $\overrightarrow{C}$, or there is an in-twin in $\overrightarrow{C}$, or we are in the second special case considered in Algorithm 2, and thus u and v are resolved, a contradiction.
   Now, assume that there is no such path, and thus there exists a vertex $b' \in \mathcal{B}$ such that there is a path of length k from b' to w. Since u is an out-neighbor of w, this implies that there is a path of length $k+1$ from b' to v. There are two subcases here. First, x and b' (or a representative along the path from b' to w) are the two sources of the cycle $\overrightarrow{C}$, u and v are its two sinks, and $\overrightarrow{C}$ contains at least 6 vertices. In this subcase, it is necessary that, in $\overrightarrow{C}$, a vertex outside of the two sources is an in-twin or has an in-arc coming from outside of $\overrightarrow{C}$, which resolves u and v, a contradiction. The second subcase is if the path from b' to v goes through x (whether $b = b'$ or not). But then, either there is an in-twin or an in-arc which resolves u and v, a contradiction, or we are in the fourth or fifth special case considered in Algorithm 2, and thus u and v are resolved, a contradiction.
2. $u, v \notin N^+(x)$, hence, u and v each have a predecessor (respectively, u' and v') on P_u^b and P_v^b. By hypothesis, there exists a vertex $b' \in \mathcal{B}$ that resolves u' and v' but not u and v, so there is a path of length k from b' to u' and a path of length $k+1$ from b' to v that does not go through v'. Note that b' cannot be behind x in P_u^b or P_v^b since otherwise it would not resolve u' and v', and it cannot be after in P_u^b or P_v^b since otherwise it would resolve u and v, a contradiction. Hence, b' (or a representative along the path from b' to v) and x are the two sources of the cycle $\overrightarrow{C}$, u' and v are its two sinks, and $\overrightarrow{C}$ contains at least six vertices. Like in the previous case, it is necessary that, either we are in the sixth special case considered in Algorithm 2, or, in $\overrightarrow{C}$, a vertex outside of the two sources is an in-twin or has an in-arc coming from outside of $\overrightarrow{C}$, which resolves u and v, a contradiction.

Hence, Algorithm 2 resolves every pair of vertices in $\overrightarrow{U}$, and thus it returns a metric basis of a unicyclic graph. Finally, it is easy to see that it computes $\mathcal{B}$ in linear time; for the concerning paths in the sixth special case, we can do a breadth-first search of the graph starting from the sink c_{i+1} to identify them, then go through the search tree again to compute, for each set of in-twins, which are all comprised of vertices in concerning paths (giving us unfixable paths) and which are not (giving us fixable paths), and a third loop to correctly relabel the concerning paths. □

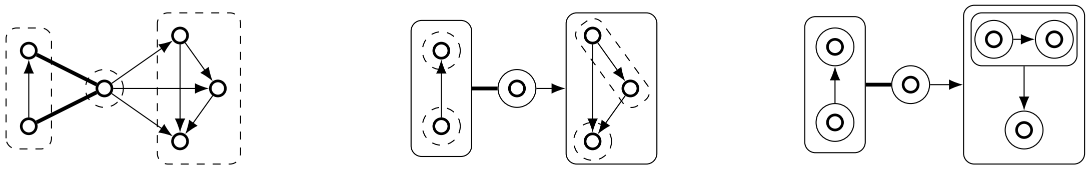


Fig. 6. An example on how to decompose and draw a digraph using modules.

4. FPT algorithm for 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 $\mathcal{X} = \{X_1, \dots, X_s\}$ 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 $\mathcal{X} = \{X_1, \dots, X_s\}$ 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. An example of a modular decomposition of a digraph is depicted on Fig. 6. 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 and strong and weak metric dimensions.

The following result lists several useful observations.

Proposition 8. *Let $\mathcal{X} = \{X_1, \dots, X_s\}$ be a factorization of G, and let $W \subseteq V(G)$ be a resolving set of G. Denote $W_i = W \cap X_i$ for each $i \in \{1, \dots, s\}$.*

(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 $\mathrm{dist}_G(x,y) \le \mathrm{mw}(G)$ or $\mathrm{dist}_G(x,y) = \infty$.
(iv) The set $\{X_i \in \mathcal{X} \mid W_i \neq \emptyset\}$ 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 non-trivial, we have $\mathrm{dist}_G(w,x) \neq \mathrm{dist}_G(w,y)$ for some $w \in W_i$.
(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 non-trivial modules must contain elements of a resolving set, as modules must be resolved locally (Proposition 8 (i)). While combining the local solutions of non-trivial modules, we need to make sure that a vertex $x \in X_i$, where X_i is non-trivial, is resolved from all $y \notin X_i$. If x and y are resolved as described in Proposition 8 (vi), then we need to do nothing special. However, if $x \in X_i$ is such that $\mathrm{dist}_G(w,x) = d$ for all $w \in W_i$ and a fixed $d \in \{1, \dots, \mathrm{mw}(G), \infty\}$, 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 $\mathrm{mw}(G)+1$ d-constant vertices in each factor due to Proposition 8 (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 non-trivial 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 and not the distances in the factors or the auxiliary graphs.

Theorem 9. *The metric dimension of a digraph G with $\mathrm{mw}(G) \le t$ 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. Let us consider one level of an optimal modular decomposition of G. Let H be a factor somewhere in the decomposition, and let $\mathcal{X} = \{X_1, \dots, X_s\}$ be the factorization of H according to the modular decomposition. For the graph H (and its non-trivial 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) $\mathbf{p} = (p_1, \dots, p_{\mathrm{mw}(G)}, p_\infty)$ where $p_d = \mathit{true}$ 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 non-trivial modules X_i. Let the factorization $\mathcal{X}$ be labeled so that the modules X_i are trivial for $i \in \{1, \dots, h\}$ and non-trivial for $i \in \{h+1, \dots, s\}$. Let $I \subseteq \{1, \dots, h\}$ and

$$P = \begin{pmatrix} \mathbf{p}^{h+1} \\ \vdots \\ \mathbf{p}^{s} \end{pmatrix}.$$

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

(a) The set $Z = \{X_i \in \mathcal{X} \mid i \in I \cup \{h+1, \dots, s\}\}$ resolves the quotient $H/\mathcal{X}$ with respect to the distances in G.
(b) For $d \in \{1, \dots, \mathrm{mw}(G), \infty\}$ and $i \in \{h+1, \dots, s\}$, if $p^i_d = \mathit{true}$, then for each trivial module $X_j = \{x_j\}$ where $j \notin I$ we have $\mathrm{dist}_G(x_i, x_j) \ne d$ or there exists $X_k \in Z \setminus \{X_i\}$ such that $\mathrm{dist}_G(x_k, x_i) \ne \mathrm{dist}_G(x_k, x_j)$.
(c) For $d_1, d_2 \in \{1, \dots, \mathrm{mw}(G), \infty\}$ and distinct $i, j \in \{h+1, \dots, s\}$, if $p^i_{d_1} = p^j_{d_2} = \mathit{true}$, then $\mathrm{dist}_G(x_i, x_j) \ne d_1$, or $\mathrm{dist}_G(x_j, x_i) \ne d_2$, or there exists $X_k \in Z \setminus \{X_i, X_j\}$ such that $\mathrm{dist}_G(x_k, x_i) \ne \mathrm{dist}_G(x_k, x_j)$.
(d) For all $d \in \{1, \dots, \mathrm{mw}(G), \infty\}$, we have $p_d = \mathit{true}$ (in $\mathbf{p}$) if and only if for some $i \in \{1, \dots, h\} \setminus I$ we have $\mathrm{dist}_G(x_j, x_i) = d$ for all $X_j \in Z$, or for some $i \in \{h+1, \dots, s\}$ we have $p^i_d = \mathit{true}$ 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 $\omega(\mathbf{p}, I, P) = \infty$.

Here we give an outline of a proof for the equality $w(H, \mathbf{p}) = \min_{I,P} \omega(\mathbf{p}, I, P)$.

We will first show that $w(H, \mathbf{p}) \le \min_{I,P} \omega(\mathbf{p}, I, P)$. This clearly holds if $\min_{I,P} \omega(\mathbf{p}, I, P) = \infty$. So assume that I and P are such that $\omega(\mathbf{p}, I, P)$ is as small as possible. Let $W \subseteq V(G)$ be such that each $W_i = X_i \cap W$ is a $w(H[X_i], \mathbf{p}^i)$-set for $i \in \{h+1, \dots, s\}$, $|W| = \omega(\mathbf{p}, I, P)$, and W fulfills the conditions (a)-(d). We will show that W fulfills the conditions (i) and (ii), and thus $|W| \ge w(H, \mathbf{p})$.

(i) W resolves $V(H)$ in G: If $x, y \in X_i$, then $i \in \{h+1, \dots, s\}$ and x and y are resolved by W_i. Assume that $x \in X_i$ and $y \in X_j$, $i \ne j$. Suppose that x and y are not resolved due to Proposition 8 (vi). Then at least one of them is d-constant or they are both in trivial modules. Now, if $i, j \in \{1, \dots, h\}$, then x and y are resolved due to condition (a). If $i \in \{1, \dots, h\}$ and $j \in \{h+1, \dots, s\}$, then x and y are resolved due to condition (b). If $i, j \in \{h+1, \dots, s\}$, then x and y are resolved due to condition (c).
(ii) This holds due to condition (d).

Therefore, $w(H, \mathbf{p}) \le \min_{I,P} \omega(\mathbf{p}, I, P)$.

Let us then show that $w(H, \mathbf{p}) \ge \min_{I,P} \omega(\mathbf{p}, I, P)$. Again, if $w(H, \mathbf{p}) = \infty$, then the claim clearly holds. So assume that $W \subseteq V(G)$ is such that $|W| = w(H, \mathbf{p})$ and W fulfills conditions (i) and (ii). Consider the sets W_i for $i \in \{h+1, \dots, s\}$. Each W_i resolves X_i in G due to Proposition 8 (v), and thus $|W_i| \ge w(H[X_i], \mathbf{p}^i)$ for $\mathbf{p}^i$ defined with respect to W_i. Let P be defined with these $\mathbf{p}^i$'s, and let $I = \{i \in \{1, \dots, h\} \mid W_i \ne \emptyset\}$. Now, $|W| \ge |I| + \sum_{i=h+1}^{s} w(H[X_i], \mathbf{p}^i)$. Moreover, the conditions (a)-(d) hold:

(a) Holds due to condition (i).
(b) Assume to the contrary that $p^i_d = \mathit{true}$ and $j \in \{1, \dots, h\} \setminus I$ is such that $\mathrm{dist}_G(x_k, x_i) = \mathrm{dist}_G(x_k, x_j)$ for all $X_k \in Z \setminus \{X_i\}$. Let $x \in X_i$ be d-constant. Since W resolves x and y, there exists $w \in W_i$ such that $\mathrm{dist}_G(w, x) \ne \mathrm{dist}_G(w, x_j)$, and thus $\mathrm{dist}_G(x_i, x_j) = \mathrm{dist}_G(w, x_j) \ne d$.
(c) Can be shown with the same technique as (b).
(d) Clear.

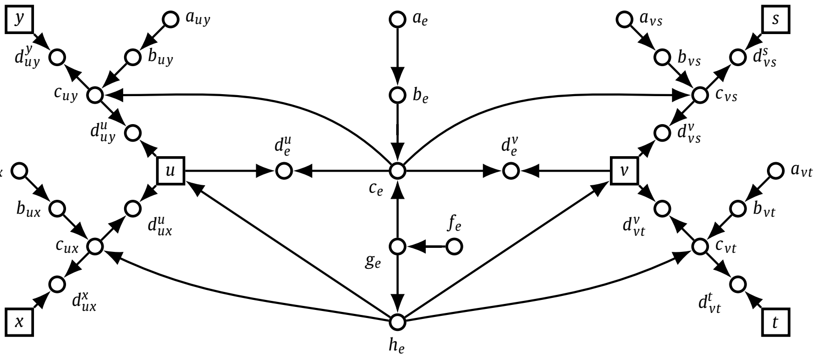


Fig. 7. Illustration of the reduction for an edge $e = uv$ of G in M, and the surrounding edges ux, uy, vs and vt. Squared vertices are the original ones from G.

Therefore, $|W| \geq \omega(\mathbf{p}, I, P)$ and $w(H, \mathbf{p}) \geq \min_{I,P} \omega(\mathbf{p}, I, P)$.

Let us then discuss the complexity of this algorithm. As a preprocessing step, we need to compute the distances between all pairs of vertices. This can be done using the Floyd-Warshall algorithm in $\mathcal{O}(n^3)$ time. An optimal modular decomposition can be computed in $\mathcal{O}(n+m)$ time [21]. We then need to compute the values $w(H, \mathbf{p})$ for each factor H starting from the trivial modules and working our way up in the decomposition. The values $w(H, \mathbf{p})$ are computed using the auxiliary values $\omega(\mathbf{p}, I, P)$. There are $\mathcal{O}(2^{t^2})$ different possibilities for I and P and their combinations (note that the vector $\mathbf{p}$ is determined based on I and P). For each pair I, P, we need to check the conditions (a)-(d), out of which (c) is the most costly time-wise and can be checked in $\mathcal{O}(t^5)$ time. Thus, computing the values $w(H, \mathbf{p})$ can be done in $\mathcal{O}(t^5 2^{t^2})$ time for each H. The total computing time then follows from the fact that there are at most $2n$ factors in any modular decomposition of a graph with n vertices. (The decomposition can be presented as a rooted tree where the vertices represent the factors and edges represent inclusion. In this tree the leaves are exactly the trivial modules, and there are n of them. Every internal vertex has degree at least 3, except the root has degree at least 2. Using the handshake lemma it is then straightforward to show that this tree can have at most $2n$ vertices.) □

The original algorithm of Belmonte et al. has conditions (a)-(g), of which (a) is (essentially) the same as (a) above, (b) and (c) are covered by (b), (d) and (e) by (c), and (f) and (g) by (d). Notice that our condition (c) is true whenever $\text{dist}_G(x_i, x_j) = \text{dist}_G(x_j, x_i)$ and $d_1 \neq d_2$. Specifically, if G is undirected, we do not need to care about (c) for pairs where $d_1 \neq d_2$.

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 10. Metric Dimension *is NP-complete, even on planar triangle-free DAGs of maximum degree 6 and maximum distance 4.*

Proof. The problem is clearly in NP: a certificate is a set of vertices, for which we can check in polynomial time if it is of the required size and if it resolves all vertices by computing the distance vectors and comparing them.

For completeness, 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 $V(G') = V(G) \bigcup_{e=uv \in E(G)} \{a_e, b_e, c_e, d^u_e, d^v_e\} \bigcup_{e=uv \in M} \{f_e, g_e, h_e\}$. For every edge $e = uv$ of G, we add the arcs $\{\overrightarrow{a_e b_e}, \overrightarrow{b_e c_e}, \overrightarrow{c_e d^u_e}, \overrightarrow{c_e d^v_e}, \overrightarrow{u d^u_e}, \overrightarrow{v d^v_e}\}$. For every edge $e = uv$ of the perfect matching M of G, assuming the neighbors 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 when $e \in M$), has no triangles, and no shortest directed path of length 5. See Fig. 7 for an illustration.

Now, we claim that G has a vertex cover of size at most k if and only if G' has metric dimension at most $k + |E(G)| + |M| = k + 4|E(G)|/3 = k + 2|V(G)|$.

If G has a vertex cover C of size k, we construct a resolving set $R(C)$ of G' as follows. Include the vertices of C in $R(C)$, as well as all vertices of $\{a_e \mid e \in E(G)\} \cup \{f_e \mid e \in M\}$. The vertices in $R(C)$ are clearly uniquely resolved. For a given edge e

of G, the vertices b_e and c_e are uniquely at distance 1 and 2 from a_e, respectively, so all vertices of these types are uniquely resolved. Among the other vertices associated to e, d_e^u and d_e^v are the only ones at distance 3 from a_e; moreover, d_e^u is at distance 1 from u and d_e^v at distance 1 from v, but not vice-versa, so $C \cap \{u, v\}$ resolves d_e^u and d_e^v. Thus, all vertices of these types are uniquely resolved. Among the remaining vertices, g_e and h_e are uniquely at distance 1 and 2 from f_e, respectively, so all vertices of these types are uniquely resolved. Finally, the vertices in $V(G) \setminus R(C)$ are resolved by the unique vertex of type f_e from which each of them is at distance 3. Hence, all vertices are uniquely resolved and $R(C)$ is indeed a resolving set of G'.

Conversely, let R be a resolving set of G' of size at most $k + |E(G)| + |M|$. Notice that for each edge e of G, one of a_e, b_e belongs to R in order to resolve this pair, and similarly, for each edge e in M, one of f_e, g_e belongs to R (in the case of strong metric dimension, a_e and f_e belong to the solution, since they are sources).

We construct a potential vertex cover $C(R)$ by taking $R \cap V(G)$. Moreover, for each edge $e = uv$ in M, we add to $C(R)$ any of u, v (if possible, one that is not yet in $C(R)$) in case the set $\{a_e, b_e, c_e, f_e, g_e, h_e, d_e^u, d_e^v\}$ contains three vertices of R. If it contains at least four, both u, v are put into $C(R)$. Similarly, for each edge $e = uv$ in $E(G) \setminus M$, we add to $C(R)$ any of u, v in case the set $\{a_e, b_e, c_e, d_e^u, d_e^v\}$ contains two vertices of R (if possible, we add one that is not yet in $C(R)$), and we add both u, v if it contains more than two vertices of R.

By the above paragraph, the resulting set $C(R)$ contains at most $|R| - |E(G)| - |M| \le k$ vertices. Now, consider a pair d_e^u, d_e^v for some edge e. If u or v is in R, it is also in $C(R)$, and e is covered by $C(R)$. Assume now that none of u, v is in R. If $e \in M$, necessarily one of u, v, d_e^u, d_e^v belongs to R to resolve that pair, and so, as none of u, v are in R, $|a_e, b_e, c_e, f_e, g_e, h_e, d_e^u, d_e^v| \ge 3$ and by our construction, either u or v (or both) have been added to $C(R)$. Thus, e is covered by $C(R)$. If $e \notin M$, the only vertices that can resolve d_e^u, d_e^v are again u, v, d_e^u, d_e^v, or a vertex $h_{e'}$ where $e' \ne e$ is an edge of G in M incident with u or v and $h_{e'}$ is not adjacent to c_e. Again, as none of u, v is in R, if one of d_e^u, d_e^v belongs to R, $|a_e, b_e, c_e, d_e^u, d_e^v| \ge 2$ and by our construction, either u or v (or both) have been added to $C(R)$. Otherwise, it must be that some vertex $h_{e'}$ is in R, where $e' \ne e$ is an edge of G in M incident with u or v (say, u and $e' = uw$) and $h_{e'}$ is not adjacent to c_e. But notice that $h_{e'}$ does not resolve $d_{e'}^u$ and $d_{e'}^w$, as $e' \in M$. Thus, either $w \in R$ and $|a_{e'}, b_{e'}, c_{'e}, f_{e'}, g_{e'}, h_{e'}, d_{e'}^u, d_{e'}^w| \ge 3$, or $w \notin R$ and $|a_{e'}, b_{e'}, c_{'e}, f_{e'}, g_{e'}, h_{e'}, d_{e'}^u, d_{e'}^w| \ge 4$. In both cases, by our construction, we would have added u to $C(R)$. Thus, in all cases, one of u, v belongs to $C(R)$ and e is covered. Thus, $C(R)$ is a vertex cover of size at most k, as needed. □

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 non-trivial.

One open question is whether Metric Dimension is 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.

Dedication to Rolf Niedermeier

This work may not have been performed without the influence of Rolf Niedermeier on Florent Foucaud. As a PhD student in 2012, Florent visited Rolf's group in TU Berlin for two weeks, and again for a month in 2013. Despite Florent not being experienced in the field of parameterized complexity, Rolf warmly welcomed these visits and made Florent feel at ease. Florent started a collaboration on the parameterized complexity of the Metric Dimension problem with some members of Rolf's group (inspired by two of them, who had just obtained an important result in the area [15]). This collaboration did not lead to any publication, nevertheless, the discussions with Rolf, his students and visitors and the friendly atmosphere in the group certainly influenced Florent's later research and inspired him to work more on the parameterized complexity of graph problems. This includes the present paper, which is also about Metric Dimension. Rolf was also a positive model on how to have a dynamic and positive research group. We have been very saddened by Rolf's unexpected and too early passing, and dedicate this paper to his memory.

CRediT authorship contribution statement

Antoine Dailly: Investigation, Writing – review & editing, Writing – original draft. **Florent Foucaud:** Writing – original draft, Investigation, Writing – review & editing. **Anni Hakanen:** Writing – review & editing, Investigation, Writing – original draft.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Antoine Dailly reports financial support was provided by French National Research Agency. Florent Foucaud reports financial support was provided by French National Research Agency. Anni Hakanen reports financial support was provided by French National Research Agency. Florent Foucaud reports financial support was provided by IDEX-ISITE CAP20-25. Anni Hakanen reports financial support was provided by Jenny and Antti Wihuri Foundation. Anni Hakanen reports financial support was provided by Academy of Finland.

Acknowledgments

We thank the anonymous referee for their careful reading of the proofs and their helpful suggestions.

Data availability

No data was used for the research described in the article.

References

[1] J. Araujo, J. Bensmail, V. Campos, F. Havet, A.K. Maia, N. Nisse, A. Silva, On finding the best and worst orientations for the metric dimension, Algorithmica 85 (10) (2023) 2962–3002.
[2] R. Belmonte, F.V. Fomin, P.A. Golovach, M.S. Ramanujan, Metric dimension of bounded tree-length graphs, SIAM J. Discrete Math. 31 (2) (2017) 1217–1243.
[3] L.M. Blumenthal, Theory and Applications of Distance Geometry, Oxford University Press, United Kingdom, 1953.
[4] G. Chartrand, L. Eroh, M.A. Johnson, O.R. Oellermann, Resolvability in graphs and the metric dimension of a graph, Discrete Appl. Math. 105 (1) (2000) 99–113.
[5] G. Chartrand, M. Raines, P. Zhang, The directed distance dimension of oriented graphs, Math. Bohem. 125 (2000) 155–168.
[6] A. Dailly, F. Foucaud, A. Hakanen, Algorithms and hardness for metric dimension on digraphs, in: International Workshop on Graph-Theoretic Concepts in Computer Science, Springer, 2023, pp. 232–245.
[7] J. Díaz, O. Pottonen, M.J. Serna, E.J. van Leeuwen, Complexity of metric dimension on planar graphs, J. Comput. Syst. Sci. 83 (1) (2017) 132–158.
[8] D. Eppstein, Metric dimension parameterized by max leaf number, J. Graph Algorithms Appl. 19 (1) (2015) 313–323.
[9] L. Epstein, A. Levin, G.J. Woeginger, The (weighted) metric dimension of graphs: hard and easy cases, Algorithmica 72 (4) (2015) 1130–1171.
[10] H. Fernau, P. Heggernes, P. van 't Hof, D. Meister, R. Saei, Computing the metric dimension for chain graphs, Inf. Process. Lett. 115 (9) (2015) 671–676.
[11] F. Foucaud, G.B. Mertzios, R. Naserasr, A. Parreau, P. Valicov, Identification, location-domination and metric dimension on interval and permutation graphs. II. Algorithms and complexity, Algorithmica 78 (3) (2017) 914–944.
[12] E. Galby, L. Khazaliya, F. Mc Inerney, R. Sharma, P. Tale, Metric dimension parameterized by feedback vertex set and other structural parameters, SIAM J. Discrete Math. 37 (4) (2023) 2241–2264.
[13] T. Gima, T. Hanaka, M. Kiyomi, Y. Kobayashi, Y. Otachi, Exploring the gap between treedepth and vertex cover through vertex integrity, Theor. Comput. Sci. 918 (2022) 60–76.
[14] F. Harary, R.A. Melter, On the metric dimension of a graph, Ars Comb. 2 (1976) 191–195.
[15] S. Hartung, A. Nichterlein, 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, IEEE Computer Society, 2013, pp. 266–276.
[16] S. Hoffmann, A. Elterman, E. Wanke, A linear time algorithm for metric dimension of cactus block graphs, Theor. Comput. Sci. 630 (2016) 43–62.
[17] S. Hoffmann, E. Wanke, Metric dimension for Gabriel unit disk graphs is NP-complete, in: A. Bar-Noy, M.M. Halldórsson (Eds.), 8th International Symposium on Algorithms for Sensor Systems, Wireless Ad Hoc Networks and Autonomous Mobile Entities (ALGOSENSORS 2012), Springer Berlin Heidelberg, Berlin, Heidelberg, 2013, pp. 90–92.
[18] D. Jean, A. Lobstein, Watching systems, identifying, locating-dominating and discriminating codes in graphs: a bibliography, Published electronically at https://dragazo.github.io/bibdom/main.pdf, 2025.
[19] S. Khuller, B. Raghavachari, A. Rosenfeld, Landmarks in graphs, Discrete Appl. Math. 70 (3) (1996) 217–229.
[20] S. Li, M. Pilipczuk, Hardness of metric dimension in graphs of constant treewidth, Algorithmica 84 (11) (2022) 3110–3155.
[21] R.M. McConnell, F. de Montgolfier, Linear-time modular decomposition of directed graphs, Discrete Appl. Math. 145 (2) (2005) 198–209.
[22] R.A. Melter, I. Tomescu, Metric bases in digital geometry, Comput. Vis. Graph. Image Process. 25 (1) (1984) 113–121.
[23] B. Mohar, Face covers and the genus problem for apex graphs, J. Comb. Theory, Ser. B 82 (1) (2001) 102–117.
[24] M. Moscarini, Computing a metric basis of a bipartite distance-hereditary graph, Theor. Comput. Sci. 900 (2022) 20–24.
[25] O.R. Oellermann, J. Peters-Fransen, The strong metric dimension of graphs and digraphs, Discrete Appl. Math. 155 (3) (2007) 356–364.
[26] C. Poisson, P. Zhang, The metric dimension of unicyclic graphs, J. Comb. Math. Comb. Comput. 40 (2002) 17–32.
[27] B. Rajan, I. Rajasingh, J.A. Cynthia, P. Manuel, Metric dimension of directed graphs, Int. J. Comput. Math. 91 (7) (2014) 1397–1406.
[28] J. Sedlar, R. Škrekovski, Bounds on metric dimensions of graphs with edge disjoint cycles, Appl. Math. Comput. 396 (2021) 125908.
[29] J. Sedlar, R. Škrekovski, Vertex and edge metric dimensions of unicyclic graphs, Discrete Appl. Math. 314 (2022) 81–92.
[30] P.J. Slater, Leaves of trees, Congr. Numer. 14 (1975) 549–559.
[31] R. Steiner, S. Wiederrecht, Parameterized algorithms for directed modular width, in: Algorithms and Discrete Applied Mathematics - 6th International Conference, CALDAM 2020, Hyderabad, India, February 13-15, 2020, Proceedings, in: Lecture Notes in Computer Science, vol. 12016, Springer, 2020, pp. 415–426.