The (Weighted) Metric Dimension of Graphs: Hard and Easy Cases

Leah Epstein[1], Asaf Levin[2], and Gerhard J. Woeginger[3]

[1] Department of Mathematics, University of Haifa, 31905 Haifa, Israel
lea@math.haifa.ac.il
[2] Faculty of Industrial Engineering and Management,
The Technion, 32000 Haifa, Israel
levinas@ie.technion.ac.il
[3] Department of Mathematics and Computer Science, TU Eindhoven P.O. Box 513,
5600 MB Eindhoven, The Netherlands
gwoegi@win.tue.nl

Abstract. For an undirected graph $G = (V, E)$, we say that for $\ell, u, v \in V$, ℓ separates u from v if the distance between u and ℓ differs from the distance from v to ℓ. A set of vertices $L \subseteq V$ is a feasible solution if for every pair of vertices $u, v \in V$ there is $\ell \in L$ that separates u from v. The metric dimension of a graph is the minimum cardinality of such a feasible solution. Here, we extend this well-studied problem to the case where each vertex v has a non-negative cost, and the goal is to find a feasible solution with a minimum total cost. This weighted version is NP-hard since the unweighted variant is known to be NP-hard. We show polynomial time algorithms for the cases where G is a path, a tree, a cycle, a cograph, a k-edge-augmented tree (that is, a tree with additional k edges) for a constant value of k, and a (not necessarily complete) wheel. The results for paths, trees, cycles, and complete wheels extend known polynomial time algorithms for the unweighted version, whereas the other results are the first known polynomial time algorithms for these classes of graphs even for the unweighted version. Next, we extend the set of graph classes for which computing the unweighted metric dimension of a graph is known to be NPC by showing that for split graphs, bipartite graphs, co-bipartite graphs, and line graphs of bipartite graphs, the problem of computing the unweighted metric dimension of the graph is NPC.

1 Introduction

Let $G = (V, E)$ be a simple, loopless, undirected graph. A vertex $\ell \in V$ is called a *separating landmark* for two vertices $u, v \in V$ with $u \neq v$, if the length of the shortest path from u to ℓ differs from the length of the shortest path from v to ℓ; sometimes we will then also say that vertex ℓ *separates* or *distinguishes* u from v. We denote the number of vertices in G by $n = |V|$ and the number of its edges by $m = |E|$. A subset $L \subseteq V$ is a *landmark set* for the graph G, if for any two vertices $u, v \in V$ with $u \neq v$ there exists a separating landmark $\ell \in L$ that separates u from v. The *metric dimension* $\mathtt{md}(G)$ of the graph G is the cardinality of the smallest landmark set in G. Note that $\mathtt{md}(G)$ is well-defined,

M.C. Golumbic et al. (Eds.): WG 2012, LNCS 7551, pp. 114–125, 2012.

as $L = V$ trivially forms a landmark set for G. Additionally, $\texttt{md}(G) = 0$ iff $|V| = 1$. We consider the problem of computing $\texttt{md}(G)$ of an input graph G. Applications of this optimization problem arise in diverse areas. See [2] for an application of this problem in network verification, [6] for an application in strategies for the Mastermind game, [9] for an application in metric geometry, [12] for an application in digital geometry, namely in digitizing of images, [11] for an application in robot navigation, and [4] for an application in drug discovery.

This metric dimension problem was introduced by Harary and Melter [9] and by Slater [15], and studied widely in the combinatorics literature. In this line of research, the exact values of the metric dimension or bounds on it for specific graph classes are obtained. We refer to [1,3,4] for results additional to those stated here (see also the survey [5]). It was shown that $\texttt{md}(G) = 1$ iff G is a path [11]. Tree input graphs (which are not paths) were considered in [9,15,11,4]. It turns out that it is possible to characterize the feasibility of a landmark set for a tree using a notion of *legs*, which are paths of vertices of degree at most 2 connected to a vertex of a higher degree. When the input graph is a cycle then $\texttt{md}(G) = 2$ [9]. Wheel graphs were mentioned in [9] and further studied in [14]. Melter and Tomescu [12] considered the problem for grid-graphs induced by lattice points in the plane when the distances are measured in the L_1 norm or in the L_∞ norm. Khuller, Raghavachari, and Rosenfeld [11] generalized one result of [12] to lattice points contained inside a d-dimensional rectangle where the distance is according to the L_1 norm (that is, the grid-graph over points in d-dimensional rectangle). The grid-graph with Euclidean metrics was studied in [13] where they relate the problem to the combinatorial coin weighing problem. Recently, Diaz et al. [8] developed a polynomial time algorithm for outerplanar graphs.

As for the complexity of the problem, Khuller, Raghavachari, and Rosenfeld [11] proved that the problem is NP-hard for general graphs and showed that one can apply the greedy algorithm for a set cover instance (where we would like to cover the pairs of vertices in a graph using sets which are defined using a single landmark). Thus, there is an $(2 \ln n + O(1))$-approximation algorithm for general graphs. Beerliova et al. [2] showed that if $P \neq NP$ then there is no $o(\log n)$-approximation algorithm for the problem. These results were strengthened in [10], where it was shown that under another complexity condition there is no $((1 - \varepsilon) \ln n)$-approximation algorithm for any $\varepsilon > 0$. They give an improved $(1 + (1 + o(1)) \ln n)$-approximation algorithm. Diaz et al. [8] showed that the problem is NP-hard even when restricted to planar graphs.

We generalize the problem to a weighted variant. Given a non-negative cost function $c : V \to \mathbf{R}_+$ the goal is to compute a landmark set L such that $\sum_{\ell \in L} c(\ell)$ is minimized. We let $\texttt{wmd}(G)$ denotes this minimum cost and we say that $\texttt{wmd}(G)$ is the weighted metric dimension of G. The `wmd` problem is to compute a landmark set L of minimum total cost, and to find $\texttt{wmd}(G)$. Our polynomial time algorithms for special classes of graphs will be for solving `wmd` while our NP-hardness proofs will hold even for the unweighted version of computing $\texttt{md}(G)$ and thus the same holds for the weighted variant as well. We are not aware of any previous work on the weighted version. We say that a feasible landmark set

is minimal if it is minimal with respect to inclusion. Note that there is always an optimal solution which is also minimal (since the cost function is non-negative), and thus we sometimes characterize the set of minimal solutions.

The problem is clearly in NP because given a landmark set, we can verify its feasibility in polynomial time. To do this, we find the vectors of the distances of each vertex in V from each of the landmarks. Afterwards, we check that there is no pair of identical vectors. In some of our algorithms we perform an exhaustive enumeration of landmark sets (among a restricted family of vertex subsets), and we can always find a cheapest feasible solution among such a restricted family of subsets (we first ensure that it contains at least one of the optimal solutions).

Our Results. We generalize the known polynomial time algorithms for `md` to `wmd` for the cases where G is a path, a tree, a cycle, or a complete wheel. We develop polynomial time algorithms for the weighted problem when G is a cograph, a k-edge-augmented tree (that is, a tree with additional k edges) for a constant value of k, and a (not necessarily complete) wheel. These results are the first polynomial time algorithms even for the unweighted version when G belongs to these classes of graphs. Next, we extend the set of graph classes for which `md` is known to be NP-complete. We show that `md` is NP-complete when the input graph for split graphs, bipartite graphs, co-bipartite graphs, and line graphs of bipartite graphs. Omitted proofs and the hardness results will be given in the full version of this paper.

Definitions and Notation. Given a graph $G = (V, E)$, a vertex $v \in V$ is a leaf if its degree is 1, it is an isolated vertex if its degree is 0, if its degree is 2 it is a path vertex, and higher degree vertices are called core vertices. For a pair of vertices u, v we denote by $d_{u,v}$ the length of the shortest path in G from u to v (i.e., the number of edges in it).

2 Paths, Trees, Cycles, and Cographs

In this section we generalize some polynomial solvable cases of $\texttt{md}(G)$ to the weighted case. These simple cases emphasize the differences between the weighted and the unweighted cases. Additionally we consider cographs.

Paths. Assume that G is a path. It suffices to have one landmark vertex at one of the end-vertices of the path [11]. Our algorithm for computing $\texttt{wmd}(G)$ for a path G is as follows: We consider two alternative solutions and pick the better one. The first one has a single vertex as a landmark: a minimum cost end point v of the path (breaking ties arbitrarily). The second solution picks the cheapest pair of distinct vertices v and v'.

Trees. Assume now that the input graph G is a tree which is not a path (and so it has at least one core vertex). We say that a (non-empty) path in the tree between a vertex adjacent to a core vertex u and a leaf v is a *leg of* u if the vertices of the path have degree at most 2 (u is not considered to be a part of the leg). For a vertex u we denote the number of legs of u by leg_u. Assume that $leg_u \geq 2$, then [11] showed that there is a landmark at each of the legs of u except for at most one such leg. In fact, summing $leg_u - 1$ over all core vertices

u with at least two legs, gives a tight bound on $\texttt{md}(G)$. Here, we generalize their approach. Consider the following algorithm. Compute leg_u for every core vertex u. Each core vertex u with $leg_u \geq 2$ is allocated $leg_u - 1$ landmarks. To place these landmarks, we find a minimum cost set of $leg_u - 1$ vertices in the legs of u, at most one vertex per leg (recall that u does not belong to its legs).

Proposition 1. *The above algorithms solve* `wmd` *for any tree G in $O(n)$ time.*

Cycles. We assume that the input graph G is a cycle. We next characterize a minimal (with respect to inclusion) feasible landmark set L. We say that a pair of vertices u, v are opposite if their distance is exactly $\frac{n}{2}$, and otherwise they are non-opposite (in which case the shortest path from u to v is unique). Note that if G is an odd-length cycle, then every pair of vertices are non-opposite.

Lemma 1. *A minimal feasible landmark set is a pair of non-opposite vertices.*

Our algorithm for solving `wmd` for a cycle G simply finds the cheapest pair of non-opposite vertices in the cycle. By the above lemma, it finds an optimal solution. Note that the running time of the algorithm is $O(n)$ by first identifying the cheapest set of three vertices (breaking ties arbitrarily), and finding the cheapest pair of non-opposite vertices among them (breaking ties arbitrarily). Our tie breaking rule is justified because even if the set of minimum weight vertices has more than three vertices, it is sufficient to consider only three of them, and similar arguments apply for the other extreme cases.

Dealing with Disconnected Input Graphs. Consider a disconnected graph $G = (V, E)$. A connected component of G is called *non-trivial connected component* if it contains at least two vertices, and otherwise it is an isolated vertex. We denote by $(V_1, E_1), \ldots, (V_p, E_p)$ the non-trivial connected components of G (for $p \geq 0$), and by $v_1, \ldots, v_t$ its isolated vertices (for $t \geq 0$), where if $p = 0$ or $t = 0$ then there is no non-trivial connected component or an isolated vertex, respectively. Without loss of generality, we assume that $c(v_t) = \max_{i:1 \leq i \leq t} c(v_i)$. In this section we show that it is sufficient to solve the weighted metric dimension problem for each non-trivial connected component of G. The time complexity of solving `wmd` is $O(n + m)$ plus the total running times of solving `wmd` for each of the non-trivial connected components of G.

Proposition 2. *An optimal solution L for* `wmd` *is achieved by the union solutions for the non-trivial connected components (V_i, E_i) of G and $v_1, v_2, \ldots, v_{t-1}$.*

Cographs. For two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ with $V_1 \cap V_2 = \emptyset$, the disjoint union $G_1 \cup G_2$ is the graph $(V_1 \cup V_2, E_1 \cup E_2)$. The product $G_1 \times G_2$ of these two graphs is obtained by first taking the disjoint union of G_1 and G_2 and then adding all the edges $\{v_1, v_2\}$ with $v_1 \in V_1$ and $v_2 \in V_2$. A graph G is a *cograph*, if (i) G consists of a single vertex or (ii) G is the disjoint union of two cographs, or (iii) G is the product of two cographs. An equivalent characterization states that G is a cograph iff it does not contain the path P_4 on four vertices as an induced subgraph. Note that the complement graph $G' = (V, V \times V - E)$ of a cograph $G = (V, E)$ is a cograph as well.

The *cotree* of a cograph G is a rooted binary tree whose leaves correspond to single-vertex graphs and whose inner vertices correspond to subgraphs of G. Every inner vertex of the cotree is labeled either by $\cup$ (union) or by $\times$ (product) and has exactly two children: if it is labeled by $\cup$ then it corresponds to the disjoint union of the two cographs that correspond to its two children, and if it is labeled by $\times$ then it corresponds to the product of the two cographs. Corneil, Perl, and Stewart [7] showed how to compute a cotree for a given cograph in linear time $O(m+n)$. Note that the number of inner vertices of the cotree is $n-1$. By Proposition 2, we conclude that we may restrict ourselves to connected cographs. Since a connected cograph is a product of two non-empty graphs, the distance between a pair of vertices v, v' of a connected cograph G is either 1 or 2. We define a *binary landmark set* L of an arbitrary cograph G (not necessarily a connected one) to be a set of vertices such that for every pair of vertices $v, v' \in V \setminus L$ there is a landmark $\ell \in L$ such that either both $\{v, \ell\} \in E$ and $\{v', \ell\} \notin E$ or both $\{v, \ell\} \notin E$ and $\{v', \ell\} \in E$. In this case we say that ℓ separates v from v'. Given a connected cograph, a set of vertices is a feasible landmark set iff it is a feasible binary landmark set. In the remainder of this section we will present a linear time algorithm for computing a binary landmark set of a minimum total cost. For a cograph G and its complement graph G', a set $L \subseteq V$ is a binary landmark set for G iff L is a binary landmark set for G'.

We adapt our decomposition of the problem for disconnected graphs to the problem of computing a binary landmark set.

Lemma 2. *Assume that G is a disjoint union of G_1 and G_2 where $G_i = (V_i, E_i)$. (i)If L is a feasible binary landmark set for G, then $L_i = L \cap V_i$ is a feasible binary landmark set for G_i, for $i = 1, 2$. (ii) Assume that L_1 and L_2 are feasible binary landmark set for G_1 and G_2, respectively. Then, $L = L_1 \cup L_2$ is a feasible binary landmark set for G iff there exists $i \in \{1, 2\}$ such that in G_i every vertex $v \in V_i \setminus L_i$ is adjacent to a vertex of L_i.*

Our algorithm for solving `wmd` of a cograph uses the cotree structure. Let a *point* be a vertex which is not a landmark. If a subgraph is a disjoint union of two subgraphs, then we treat recursively each of the two subgraphs but we keep track of the existence of a point which is not adjacent to any landmark in its subgraph. If a subgraph is the product of two subgraphs, then we transform our problem to the complement of the subgraph, and apply the first case. Since by moving from a graph to its complement, the role of a point which is not adjacent to any landmark switches with the role of a point which is adjacent to all landmarks, we will keep track of the number of points (zero or one) of each of these types, and every subgraph in the cotree corresponds to four optimization problems. Thus, `wmd` can be solved in linear time $O(m+n)$ for cographs.

3 *k*-Edge-Augmented Trees

In this section we consider the class of connected graphs for which a removal of at most k edges results in a spanning tree. We call this class of graphs *k-edge-augmented trees*. Our polynomial time algorithm for computing `wmd` first

applies a preprocessing step which handles the tree-like part, and then uses an exhaustive enumeration approach for selecting an optimal landmark set in a reduced problem. Clearly our algorithm is polynomial only if k is a constant and it is unlikely that there is an algorithm which is polynomial in n and k for solving this problem (since any connected graph is also a k-edge-augmented tree for a sufficiently large value of k, and the problem is NP-hard for general graphs).

Preprocessing step. Our preprocessing step uses the following procedure which can be applied on a vertex u of degree at least 3 with p legs (where a leg is a path consisting of at least one vertex, starting at a neighbor of u and ending at a leaf, where all vertices except for the leaf have degree 2) for $p \geq 2$. Consider subsets of $p-1$ vertices which belong to the legs of u, at most one vertex per leg. We choose L_u to be a set of minimum cost among the sets which satisfy these constraints. We will place landmarks at L_u and remove from the graph every vertex belonging to a leg of u which contains a vertex of L_u (that is, the leg which does not contain a vertex of L_u is not removed). In the remaining graph we change the cost of u to zero, and thus allow the solution to place a landmark at u without increasing its cost. We apply this preprocessing on one vertex at a time until there is no vertex which has at least two legs, and in the remaining graph every vertex has at most one leg. Since we already dealt with trees in Proposition 1, we assume that the input graph is not a tree. The following lemma holds in fact for trees as well (but in order to prove it for trees as well, the special case of a spider graph should be treated separately).

Lemma 3. *Denote by $G = (V, E)$ the input graph which is not a tree, and by $G' = (V', E')$ the graph obtained after applying the procedure at a vertex u (that is, the set of vertices after removing all the legs of u but one). Let L_u denote the set of landmarks which we placed on the removed legs of u, and let L' denote the set of landmarks in an optimal solution of the remaining graph. Then, $L = L_u \cup L' \setminus \{u\}$ is an optimal landmark set in G.*

The case of $k = 1$. The unweighted version of computing $\texttt{md}(G)$ for the case of 1-edge-augmented tree is discussed in [4] (where such graphs are called unicycles). Here we consider the weighted case. If $k = 1$, then at the end of the preprocessing phase we are left with a cycle C where some of its vertices have legs (at most one leg for each vertex of C). Recall that some of the vertices of G may have zero cost resulting from the preprocessing step, and if we choose to place a landmark at such a vertex u, then the solution returned by the algorithm skips it (but has at least one landmark in the tree-like part connected to u which was removed in the preprocessing step). In what follows we only consider the graph G resulting from the preprocessing. We next characterize a minimal landmark set for this type of graphs. This characterization shows that any minimal landmark set for the resulting graph has at most three vertices. By enumerating all subsets consisting of two or three vertices we can choose the cheapest feasible landmark set and solve `wmd` in polynomial time.

We denote by n_C the number of vertices in C. For a vertex v, we let v' be its *cycle vertex* which is defined as follows. v' is the closest vertex in C to v, that is if $v \in C$ then $v' = v$ and otherwise v belongs to some leg of a vertex in

C, and we let v' denote this vertex of C. Consider a path P from u to v, then its C-length is defined as the number of edges of C which belong to P. We say that a path P from u to v is a clockwise path if it traverses the edges in $P \cap C$ in a clockwise order, and otherwise it is a counterclockwise path. We say that two vertices are non-opposite in G if their cycle vertices are distinct, and the C-length of the shortest path from u to v is not equal to $\frac{n_C}{2}$.

Lemma 4. *Let $L \subseteq V$ be a feasible minimal landmark set. There is no cycle vertex u such that L contains a pair of vertices v_1, v_2 whose cycle vertex is u.*

In what follows, we focus on a minimal landmark set, and thus assume that it does not contain two vertices with a common cycle vertex.

Lemma 5. *(i) If $L \subseteq V$ contains a pair of non-opposite vertices u_1, u_2, then for every pair of vertices $x_1, x_2 \in C$, there exists $w \in \{u_1, u_2\}$ that separates x_1 from x_2. (ii) If L consists of at least three vertices, no pair of which have the same cycle vertex, then for every pair of vertices $x, y \in C$ there is $\ell \in L$ that separates x from y.*

We next define a *covering of the legs by landmarks*. We say that a landmark ℓ clockwise-covers (counterclockwise-covers) the leg of a vertex u if one of the following conditions hold: Either ℓ is one of the vertices of the leg of u, or the clockwise path (counterclockwise path) from u to ℓ has C-length of at least 1 and at most $\frac{n_C+1}{2}$ (u cannot cover its own leg). We say that a leg is *covered* by $L \subseteq V$ if there is a landmark in L which clockwise-covers the leg and (perhaps another) landmark in L which counterclockwise-covers the leg.

Lemma 6. *Let L be a set of vertices, where $|L| \geq 3$ and no two of which have the same cycle vertex. L is a feasible landmark set iff every leg is covered by L.*

Lemma 7. *Let L be a minimal landmark set. Then $|L| \leq 3$.*

To summarize, our algorithm for computing $\texttt{wmd}(G)$ where G is a 1-edge-augmented tree is to apply the preprocessing step, and afterwards try all possibilities of sets L such that $|L| \leq 3$, and for each of them test its feasibility in polynomial time, and among the feasible solution we pick a cheapest one. Clearly, the algorithm runs in polynomial time and computes a cheapest minimal feasible landmark set. Therefore, we have established the following.

Proposition 3. *There is a polynomial time algorithm for solving* `wmd` *for 1-edge-augmented trees.*

The General Case. Assume that $G = (V, E)$ is the graph resulting from applying the preprocessing step, so in G every vertex has at most one leg. The case of $k = 1$ is already solved, and here we assume that $k \geq 2$ is a fixed constant. We next define a subgraph of G called the base graph $G_b = (V_b, E_b)$ resulting from G by removing the vertices of all legs. We next characterize the structure of this base graph. That is, we will show that it consists of $O(k)$ edge disjoint paths connecting core vertices where all internal vertices are path vertices.

The Structure of the Base Graph. For a vertex $u \in V_b$, we denote by $deg_b(u)$ its degree in G_b. Note that G_b contains no leaves, and thus the degree of every vertex is at least 2. Moreover, since G_b is a connected subgraph of G and every cycle of G belongs to G_b as well, G_b results from a tree T by adding exactly k edges. Thus the tree T has $|V_b| - 1$ edges, and thus the number of edges in G_b is $|V_b| + k - 1$. Therefore, $\sum_{u \in V_b} deg_b(u) = 2|V_b| + 2k - 2$ and thus $\sum_{u \in V_b}(deg_b(u) - 2) = 2k - 2$.

Lemma 8. *The base graph G_b is decomposed into $q \leq 3k-3$ edge disjoint paths, where in G_b every internal vertex of a path has degree 2, and the end-vertices of such a path are core vertices.*

Given a vertex $v \in V$, we define its base vertex as the vertex $v' \in V_b$ which is the closest to v. Given the path decomposition defined above, we associate each vertex v in V with one of the paths in the following way. If the base vertex of v belongs to exactly one of the paths then we associate v with this path. Otherwise, the base vertex v' of v is a core vertex in G_b, and we associate all the vertices in G whose base vertex is v' with one of the paths incident to v'.
Bounding the Number of Landmarks Associated with One Path. Next, we consider a minimal feasible landmark set L, and one specific path P in the path decomposition of G_b. Our goal is to bound the number of landmarks in L which are associated with P. The following lemma generalizes Lemma 4 to the case of $k \geq 2$.

Lemma 9. *(i) Let L be a minimal feasible landmark set, and let P be a path in the path decomposition of G_b. Then, the number of vertices in L which are associated with P is at most six. (ii) Let L be a minimal landmark set of a graph G which results from a k-edge-augmented tree by the preprocessing step (where $k \geq 2$). Then $|L| \leq 18k - 18$.*

To summarize, our algorithm for computing $\texttt{wmd}(G)$ where G is a k-edge-augmented tree (for $k \geq 2$) is to apply the preprocessing step, afterwards, try all possibilities of sets L such that $|L| \leq 18k - 18$, for each of them test its feasibility in polynomial time, and among the feasible solution we pick a cheapest one. Clearly, the algorithm runs in polynomial time (for a constant value of k) and computes a cheapest minimal feasible landmark set.

4 Wheels

Complete Wheels. In this section we consider complete wheels. A (complete) wheel on n vertices $\{1, 2, \ldots, n\}$ is defined as follows. There is a cycle C over the vertices $1, 2, \ldots, n-1$ (the clockwise order of the vertices along C is $1, 2, \ldots, n-1, 1$), and vertex n is adjacent to all other vertices. Vertex n is called the *hub* of the wheel, whereas the other vertices are called *cycle-vertices*. The distances in G are either 1 or 2, clearly the distance between every cycle-vertex and the hub is 1, the distance of every cycle-vertex and its two neighbors along the cycle is 1,

and between every pair of (non-adjacent) cycle-vertices are reachable via a two-edge path through the hub. Consider a feasible landmark set L. A gap between consecutive landmarks is defined as follows: If $\ell, \ell' \in L$ are cycle vertices such that along the clockwise path from ℓ to ℓ' there is no other landmark, then the set of internal vertices of the clockwise path from ℓ to ℓ' is the *gap* (between ℓ and ℓ'), and we say that the gap is adjacent to ℓ and ℓ'. The length of the gap is the number of vertices in the gap. Here, we characterize the lengths of the gaps in a feasible solution L.

Lemma 10. *Assume that G is a wheel over at least 8 vertices. Let $L \subseteq V$. L is a feasible landmark set iff the following three conditions hold with respect to L: 1. There is no gap of length at least 4. 2. There is at most one gap of length 3. 3. Two gaps of length at least two are not adjacent to a common vertex of $L \cap C$.*

Theorem 1. *Given a complete wheel G,* `wmd` *can be solved in linear time.*

A minimal landmark set L does not contain the hub, and based this characterization, our algorithm is a simple dynamic program. By [14], for a complete wheel G on n vertices, $\mathtt{md}(G) = \Theta(n)$, and since it is a $(n-1)$-edge-augmented tree, a k-edge-augmented tree needs $\Omega(k)$ landmarks in any feasible solution.

The general case A (non-complete) wheel on n vertices $\{1, 2, \ldots, n\}$ is defined as follows. There is a cycle C over the vertices $1, 2, \ldots, n-1$ (the clockwise order of the vertices along C is $1, 2, \ldots, n-1, 1$), and vertex n is adjacent to some of the other vertices. Vertex n is called the *hub* of the wheel, whereas the other vertices are called *cycle-vertices*. The neighbors of n are called *connectors*, and the edges incident at n are called *chords*. We let *layer j* be the set of vertices of distance j from n, and denote it by V_j (i.e., $V_j = \{u \in V : d_{u,n} = j\}$, and thus $V_0 = \{n\}$ and V_1 is the set of connectors). Let $L \subseteq V$, we say that a cycle vertex u is close to $\ell \in L$ if $d_{\ell,u} < d_{\ell,n} + d_{n,u}$. We say that u is close to L if there is $\ell \in L$ such that u is close to ℓ. In this section we consider wheels with at least 22 connectors, and present a polynomial time algorithm for solving `wmd` for such a graph. Since wheels with at most 21 connectors are k-edge-augmented trees for a value of k such that $k \leq 21$, we conclude that we will obtain a polynomial time algorithm for solving `wmd` on (arbitrary) wheels. Thus let $G = (V, E)$ be a wheel with at least 22 connectors. We first characterize a minimal landmark set.

Lemma 11. *(i) Let L be a feasible landmark set. Then for every j there is at most one vertex of V_j which is not close to L. Moreover, the vertices in V which are not close to L form a shortest path from some vertex v to the hub. (ii) Let $\ell \in V$. The set of vertices which are close to ℓ is a subpath P of C containing ℓ. Moreover, consider P as a clockwise path from u to v, then the subpath of P from u to ℓ (including u, ℓ) has at most two connectors, and the subpath of P from ℓ to v has at most two connectors. Thus, P contains at most four connectors. (iii) L is a feasible landmark set iff the following two conditions hold: For every layer V_j, there is at most one vertex of V_j which is not close to L. For every $\ell \in L$ and every j, if $u_1, u_2 \in V_j$ are close to ℓ and $d_{\ell,u_1} = d_{\ell,u_2}$, then there is*

$\ell' \in L \setminus \{\ell\}$ which is close to at least one of the vertices u_1 or u_2. (iv) A minimal landmark set L does not contain the hub, and $|L| \geq 6$.

Let L^* be a fixed optimal solution. We say that two landmarks ℓ_1 and ℓ_2 are consecutive if the clockwise path from ℓ_1 to ℓ_2 does not contain an additional landmark, such a path is called the *natural path between the landmarks*. Given a landmark $\ell \in L^*$, we say that a pair of cycle-vertices x, y is a bad pair with respect to ℓ (or of ℓ), if x, y are close to ℓ, belong to a common layer, $d_{x,\ell} = d_{y,\ell}$, and the clockwise path from x to y traverses ℓ. Recall that in this case there must be a landmark $\ell' \neq \ell$ which is close to at least one of x and y, and we say that ℓ' covers the bad pair x, y of ℓ. A *minimal bad pair of a landmark* ℓ is a bad pair x, y such that $d_{x,\ell}$ is minimal among all bad pairs of ℓ.

Lemma 12. *(i) Let x, y be a bad pair with respect to a landmark ℓ. Let $\ell' \neq \ell$ be a landmark which is close to at least one of x and y. Let ℓ_1 and ℓ_2 be landmarks such that ℓ_1 and ℓ are consecutive landmarks and ℓ and ℓ_2 are consecutive landmarks. Then the landmark ℓ_1 is close to x, or the landmark ℓ_2 is close to y (or both). (ii) Let $\ell' \neq \ell$ be a landmark that covers a minimal bad pair x, y of ℓ such that either ℓ', ℓ or ℓ, ℓ' are consecutive, then ℓ' covers every bad pair of ℓ.*

We say that a minimal bad pair x, y of ℓ which is covered by ℓ' is *covered from the left by* ℓ' if ℓ' and ℓ are consecutive, and otherwise if ℓ and ℓ' are consecutive we say that x, y are covered from the right by ℓ'. By Lemma 12, a bad pair which is covered is either covered from the right or from the left, by some ℓ'.

Corollary 1. *A landmark set $L \subseteq V$ is a minimal feasible landmark set iff it satisfies the following three conditions. 1. $n \notin L$. 2. The set of vertices which are not close to L forms a shortest path from n to some vertex v (possibly $v = n$), in particular, if $v \neq n$ then the path contains exactly one connector and no landmark. 3. For every $\ell \in L$, there is $\ell' \in L \setminus \{\ell\}$ which covers the minimal bad pair of ℓ either from the left or from the right (if there exists a bad pair of ℓ).*

If there is a cycle-vertex which is not close to L^*, then all such cycle-vertices form a path not containing a landmark. We guess a pair of consecutive landmarks in L^* such that if there exists a cycle-vertex not close to L^*, then all such vertices appear along the natural path between the two guessed landmarks. Without loss of generality assume that the two guessed landmarks are 1 and $k \leq n-1$ and the natural path between them is the clockwise path from k to 1. The number of possibilities for the selection of $1, k$ is $O(n^2)$. We verify that the set of vertices along the natural path from k to 1 which are not close to 1 or to k (if such a cycle-vertex exists) form a shortest path from some vertex to n (and it contains at most one connector since all connectors are in V_1). If this condition does not hold, then this possibility is impossible, and we stop considering it. In what follows, we consider a possibility which passed this test. Since the number of connectors along the natural path from k to 1 is at most 5 connectors, the clockwise path from 1 to k contains at least 17 connectors, and it is not a shortest path.

Let $1 < v < k$. We define $low(v)$ as the minimum index i such that $1 \leq i \leq v$ and the clockwise path from i to v is a unique shortest path. We let $high(v)$

be the maximum index i such that $v \leq i \leq k$ and the clockwise path from v to i is the unique shortest path. $\delta(v)$ is the minimum number $i \geq 1$ such that $v - i \geq low(v)$ and $v + i \leq high(v)$ belong to a common layer. If such i does not exist, we let $\delta(v) = \infty$. The motivation for this definition of $\delta(v)$ is to identify the minimal bad pair of v (if it exists).

Claim. Let $1 < v < k$ be a landmark such that ℓ_1, v are consecutive landmarks and v, ℓ_2 are consecutive landmarks ($1 \leq \ell_1 < v < \ell_2 \leq k$ since $1, k$ are landmarks), assume that v has a bad pair, and let x, y be the minimal bad pair of v. If $\{x, y\}$ is not contained in the clockwise path from 1 to k, then either 1 or k covers x, y. Otherwise, x, y is covered iff at least one of the following conditions holds: (i) $x \leq high(\ell_1)$, (ii) $y \geq low(\ell_2)$.

For the vertex 1, we define the parameters $Low(1), High(1), \Delta(1)$ as follows. $Low(1)$ is the minimum index i such that the clockwise path from i to 1 is the unique shortest path (by definition $n-1$ satisfies this condition, hence $Low(1)$ is well-defined, and thus the vertices $i, i+1, \ldots, n-1$ are close to 1). $High(1)$ is the maximum index i such that the clockwise path from 1 to i is the unique shortest path (similarly to the existence of $Low(1)$, the value of $High(1)$ is well-defined, vertices $2, 3, \ldots, High(1)$ are close to 1, and $High(1) < k$). $\Delta(1)$ is the minimum number $i \geq 1$ such that $i+1$ and $n-i \geq k+1$ are both close to 1 (and have the same distance from 1) and belong to a common layer. If such i does not exist, we let $\Delta(1) = \infty$. It can be the case that there exists a bad pair of 1 even if $\Delta(1) = \infty$ but in this case we show later that k covers all such bad pairs.

For the vertex k, we define similar parameters $Low(k), High(k), \Delta(k)$ as follows. $Low(k)$ is the minimum index $i > 1$ such that the clockwise path from i to k is the unique shortest path (by definition $k-1$ satisfies this condition, hence $Low(k)$ is well-defined). $High(k)$ is the maximum index $i \leq n-1$ such that the clockwise path from k to i is the unique shortest path (note that it is possible that vertex 1 is also close to k, using the clockwise path from k to 1, but we are not interested whether this holds or not). $Delta(k)$ is the minimum number $i \geq 1$ such that $k + i \leq n-1$ and $k - i \geq 1$ are both close to k (and have the same distance from k) and belong to a common layer. If such i does not exist, we let $\Delta(k) = \infty$. We also let $low(1) = 1$, $high(1) = High(1)$, $low(k) = Low(k)$ and $high(k) = k$. We define $\delta(1)$ and $\delta(k)$ in the following way. We let $\delta(1) = \infty$ if $High(k) \geq n - \Delta(1)$. Otherwise, $\delta(1) = \Delta(1)$. The motivation is to define $\delta(1)$ to be infinite if there is no bad pair of 1, or if k covers the minimal bad pair of 1. Similarly, we define $\delta(k) = \infty$ if $Low(1) \leq k + \Delta(k)$. Otherwise, we let $\delta(k) = \Delta(k)$. The differences between Δ and δ, and the property that $\Delta(1), \Delta(k)$ can be infinite even if there is a bad pair of 1, and k, respectively, follow since $High(k) \geq n - \Delta(1)$ iff the minimal bad pair of 1 is covered by k, and $Low(1) \leq k + \Delta(k)$ iff the minimal bad pair of k is covered by 1.

We define a function $F : \{1, 2, \ldots, k\} \times \{left, right\} \rightarrow \mathbf{R}^+$ as follows. $F(v, right)$ ($F(v, left)$) is the minimum cost of a set L such that $1, v \in L$, every $1 < i < v$ is close to at least one of the vertices in L, and for every $\ell \in L \setminus \{v\}$ ($\ell \in L$, respectively) such that $\delta(\ell)$ is finite the minimal bad pair of ℓ is covered by a vertex in $L \setminus \{\ell\}$. We compute

the values of F using the following dynamic program. If $\delta(1) = \infty$, then $F(1, right) = F(1, left) = c(1)$, and if $\delta(1)$ is finite, then $F(1, left) = \infty$ and $F(1, right) = c(1)$. For $\ell > 1$ we define sets of feasible values of $\ell' < \ell$ (to be consecutive landmarks) $\alpha(\ell) = \{\ell' \in \{1, 2, \ldots, \ell - 1\} : high(\ell') \geq low(\ell) - 1\}$, $\beta(\ell) = \{\ell' \in \alpha(\ell) : high(\ell') \geq \ell - \delta(\ell)\}$, $\gamma(\ell) = \{\ell' \in \alpha(\ell) : low(\ell) \leq \ell' + \delta(\ell')\}$. The recursive formula is as follows.

1. $F(\ell, left) = \min\left\{\min_{\ell' \in \beta(\ell)} F(\ell', left), \min_{\ell' \in \beta(\ell) \cap \gamma(\ell)} F(\ell', right)\right\} + c(\ell)$,
2. $F(\ell, right) = \min\left\{\min_{\ell' \in \alpha(\ell)} F(\ell', left), \min_{\ell' \in \gamma(\ell)} F(\ell', right)\right\} + c(\ell)$.

If $\delta(k) = \infty$, then we are looking for $F(k, right)$, and otherwise we are looking for $F(k, left)$. Then, by backtracking we compute the optimal landmark set. We conclude that the algorithm computes an optimal solution in polynomial time. Thus, we proved that for a wheel G, `wmd` can be solved in polynomial time.

References

1. Babai, L.: On the order of uniprimitive permutation groups. Annals of Mathematics 113(3), 553–568
2. Beerliova, Z., Eberhard, F., Erlebach, T., Hall, A., Hoffmann, M., Mihalák, M., Ram, L.S.: Network discovery and verification. IEEE Journal on Selected Areas in Communications 24(12), 2168–2181 (2006)
3. Cáceres, J., Hernando, M.C., Mora, M., Pelayo, I.M., Puertas, M.L., Seara, C., Wood, D.R.: On the metric dimension of cartesian products of graphs. SIAM Journal on Discrete Mathematics 21(2), 423–441 (2007)
4. Chartrand, G., Eroh, L., Johnson, M.A., Oellermann, O.R.: Resolvability in graphs and the metric dimension of a graph. Discrete Applied Mathematics 105(1-3), 99–113 (2000)
5. Chartrand, G., Zhang, P.: The theory and applications of resolvability in graphs: A survey. Congressus Numerantium 160, 47–68 (2003)
6. Chvátal, V.: Mastermind. Combinatorica 3(3), 325–329 (1983)
7. Corneil, D.G., Perl, Y., Stewart, L.K.: A linear recognition algorithm for cographs. SIAM Journal on Computing 14(4), 926–934 (1985)
8. Diaz, J., Pottonen, O., Serna, M., van Leeuwen, E.J.: On the complexity of metric dimension. CoRR, abs/1107.2256. Proc. of ESA 2012 (to appear, 2012)
9. Harary, F., Melter, R.A.: The metric dimension of a graph. Ars Combinatoria 2, 191–195 (1976)
10. Hauptmann, M., Schmied, R., Viehmann, C.: Approximation complexity of metric dimension problem. Journal of Discrete Algorithms (2011) (to appear)
11. Khuller, S., Raghavachari, B., Rosenfeld, A.: Landmarks in graphs. Discrete Applied Mathematics 70(3), 217–229 (1996)
12. Melter, R.A., Tomescu, I.: Metric bases in digital geometry. Computer Vision, Graphics, and Image Processing 25, 113–121 (1984)
13. Sebö, A., Tannier, E.: On metric generators of graphs. Mathematics of Operations Research 29(2), 383–393 (2004)
14. Shanmukha, B., Sooryanarayana, B., Harinath, K.S.: Metric dimension of wheels. Far East Journal of Applied Mathematics 8(3), 217–229 (2002)
15. Slater, P.J.: Leaves of trees. Congressus Numerantium 14, 549–559 (1975)