Sequential Metric Dimension

Julien Bensmail, Dorian Mazauric, Fionn Mc Inerney, Nicolas Nisse, Stéphane Pérennes

Sequential Metric Dimension *

Julien Bensmail[1], Dorian Mazauric[2], Fionn Mc Inerney[1], Nicolas Nisse[1], and Stéphane Pérennes[1]

[1] Université Côte d'Azur, Inria, CNRS, I3S, France
[2] Université Côte d'Azur, Inria, France
2004 route des Lucioles, 06902 Sophia Antipolis, France
Emails: {julien.bensmail, dorian.mazauric, fionn.mc-inerney, nicolas.nisse, stephane.perennes}@inria.fr

Abstract. Seager introduced the following game in 2013. An invisible and immobile target is hidden at some vertex of a graph G. Every step, one vertex v of G can be probed which results in the knowledge of the distance between v and the target. The objective of the game is to minimize the number of steps needed to locate the target, wherever it is.
We address the generalization of this game where $k \geq 1$ vertices can be probed at every step. Our game also generalizes the notion of the *metric dimension* of a graph. Precisely, given a graph G and two integers $k, \ell \geq 1$, the Localization Problem asks whether there exists a strategy to locate a target hidden in G in at most ℓ steps by probing at most k vertices per step. We show this problem is NP-complete when k (resp., ℓ) is a fixed parameter.
Our main results are for the class of trees where we prove this problem is NP-complete when k and ℓ are part of the input but, despite this, we design a polynomial-time $(+1)$-approximation algorithm in trees which gives a solution using at most one more step than the optimal one. It follows that the Localization Problem is polynomial-time solvable in trees if k is fixed.

Keywords: Games in graphs, metric dimension, complexity.

1 Introduction

Localization (or *Identification*) problems consist of distinguishing the vertices of a connected graph $G = (V, E)$ using a smallest subset $R \subseteq V$ of its vertices. Many variants have been studied depending on how a subset of vertices allows to identify other vertices. For instance, *identifying codes* [16], *adaptive identifying codes* [2], and *locating dominating sets* [21] ask for the vertices to be distinguished by their neighbourhood in R. Another well studied example is the one of a *resolving set* [13, 20] which aims at distinguishing each vertex of a graph by its distance to each vertex of this set. Given a graph G, the main problem is

* This work has been partially supported by ANR program "Investments for the Future" under reference ANR-11- LABX-0031-01, the Inria Associated Team AlDyNet. Due to lack of space, several proofs have been omitted and can be found in [3].

to compute a resolving set with minimum size, called the *metric dimension* of G [13, 20]. The corresponding decision problem (first shown to be NP-complete in [12]) is NP-complete in planar graphs [8] and in graphs of diameter 2 [11], and W[2]-hard (parameterized by the solution's size) [14]. On the positive side, the problem is FPT in the class of graphs with bounded treelength [1]. Bounds on the metric dimension have also been determined for various graph classes [10]. In this paper, we address a *sequential* variant of this problem.

Let us consider a graph $G = (V, E)$ where an unknown vertex $t \in V$ hosts a hidden (invisible) and immobile target. *Probing* one vertex $v \in V$ results in the knowledge of the distance between t and v, denoted by $d_G(v, t)$. Probing a set $R \subseteq V$ of vertices results in the distance vector $(d_G(v, t))_{v \in R}$ and a set is a *resolving set* if the distance vectors are pairwise distinct for every $t \in V$. The *metric dimension* of G, denoted by $MD(G)$, is then the minimum number of vertices that must be probed simultaneously (in one step) to determine the location t of the target wherever it is. For instance, in the case of a path, probing one of its ends is sufficient to locate the target, *i.e.*, $MD(P) = 1$ for every path P. Another example (that we use throughout the paper) is the case of a star (tree with a universal vertex) with n leaves, denoted by S_n, for which it is necessary and sufficient to probe every leaf but one, *i.e.*, $MD(S_n) = n - 1$.

If less than $MD(G)$ vertices can be probed at once, it is natural to allow more than one step. Obviously, if at most $1 \leq k < MD(G)$ vertices can be probed at once, it is always feasible to locate an immobile target in $\lceil MD(G)/k \rceil$ steps by sequentially probing k different vertices of a smallest resolving set at each step. However, there are graphs for which the target can be located much faster. In [18], Seager initiated the study of the following sequential locating game: at each step, one vertex of a graph can be probed, and the objective is to minimize the number of steps required to locate the target, wherever it is. Seager gave bounds and exact values on this minimum number of steps in particular subclasses of trees (e.g., subdivisions of caterpillars) [18] but left the problem open in trees in general. In this paper, we study the generalization of this game where $k \geq 1$ vertices can be probed at each step.

Precisely, let $k \geq 1$ be an integer and let $G = (V, E)$ be a graph hosting an invisible and immobile target hidden at $t \in V$. A *k-strategy* is allowed to probe at most k vertices at each step of the game (where the choice of the probed vertices at some step may obviously depend on the results of the probes during previous steps) until the location t of the target is uniquely determined. Let $\lambda_k(G)$ denote the minimum integer h such that there exists a k-strategy that locates the target in G in at most h steps, wherever it is. Given G, k and $\ell \geq 1$, the Localization Problem asks whether $\lambda_k(G) \leq \ell$. We also consider the dual parameter $\kappa_\ell(G)$ defined as the minimum integer h such that there exists an h-strategy that locates the target in G in at most ℓ steps. Note that, for every graph G, $\kappa_1(G)$ is exactly the metric dimension $MD(G)$ of G, and $\lambda_k(G) \leq \ell$ if and only if $\kappa_\ell(G) \leq k$. We are interested in the complexity of the Localization Problem in general graphs and particularly in trees. Note that by the remarks above , the Localization Problem and Metric Dimension Problem (for which $\ell = 1$)

behave very differently, so knowing that Metric Dimension Problem is NP-complete does not imply the same for the Localization Problem.

1.1 Related Work

The literature related to localizing a target in a graph is vast. Below, we focus on the related work that we find the most relevant.

Moving target. Sequential games related to resolving sets have first been introduced and mainly studied in the case of a mobile target. That is, at every step, some vertices may be probed and, if the target has not been located yet, it may move to one of its neighbours (sometimes, it is required that the target cannot move to a vertex that has been probed during the previous step which is called the "no-backtrack condition") [17]. In this setting, locating the target may not be feasible. For instance, it is not possible to locate a moving target in a triangle when probing one vertex per step if the target may "backtrack". The question of how many times all the edges of a graph must be subdivided to ensure locating a moving target probing 1 vertex (resp., k vertices) per step has been addressed in [7] (resp. [15]). Let a graph be called *locatable* if there exists a 1-strategy for locating the target in a finite number of steps with the "no-backtrack condition". The case of trees with the "no-backtrack condition" has first been studied in [17] where it was shown that all trees T are locatable, and in [6], the upper bound on the number of steps it takes to locate the target in T was improved. In [19], the case of trees where the target may "backtrack" was considered. Let $\zeta(G)$ be the minimum integer k such that there exists a k-strategy for locating a moving target in G. In [5], it was shown that deciding whether $\zeta(G) \leq k$ is NP-hard and that $\zeta(G)$ is not bounded in the class of graphs with treewidth 2. Moreover, $\zeta(G) \leq 3$ for any outerplanar graph G [4].

Relative distance and centroidal dimension. Foucaud *et al.* defined a variant of resolving sets, called *centroidal basis*, where the vertices of a graph must be distinguished by their *relative* distance to the probed vertices [9]. In this setting, given an integer $k \geq 2$, probing a set $B = \{v_1, \ldots, v_k\}$ of vertices results in the vector $(\delta_{i,j}(t))_{1 \leq i < j \leq k}$ where, for every $1 \leq i < j \leq k$, $\delta_{i,j}(t) = 0$ if $d_G(t, v_i) = d_G(t, v_j)$, $\delta_{i,j}(t) = 1$ if $d_G(t, v_i) > d_G(t, v_j)$ and $\delta_{i,j}(t) = -1$ otherwise. In other words, probing any two vertices returns the information of which one is closer to the target or whether they are equidistant from it. The set B is a *centroidal basis* if the vectors of relative distances for every $t \in V$ are pairwise distinct. The *centroidal dimension* of a graph G, denoted by $CD(G) \geq 2$, is the minimum size of a centroidal basis of G [9] (this is well defined since, clearly, V is a centroidal basis of G). The decision problem associated to the centroidal dimension is NP-complete and almost tight bounds on the centroidal dimension of paths have been computed [9].

A sequential variant of the centroidal basis can naturally be defined. This variant has been studied in the case of a moving target in [4].

Here, we also initiate the study of this variant when the target is immobile. Let $k \geq 2$ be an integer and G be a graph. Let $\lambda_k^{rel}(G)$ denote the minimum

integer h such that there exists a k-strategy that locates (using relative distances) a hidden immobile target in G in at most h steps, whatever be the location of the target. Given G, k, and ℓ, the Relative-Localization Problem asks whether $\lambda_k^{rel}(G) \leq \ell$. The dual parameter $\kappa_\ell^{rel}(G)$ is defined as the minimum integer h such that there exists an h-strategy (with relative distances) that locates the target in G in at most ℓ steps. Note that, for every graph G, $\kappa_1^{rel}(G)$ is exactly the centroidal dimension $CD(G)$ of G, and $\lambda_k^{rel}(G) \leq \ell$ if and only if $\kappa_\ell^{rel}(G) \leq k$.

1.2 Our results

In the whole paper, G denotes a connected undirected simple graph. We consider the computational complexity of the Localization Problem. In Section 2, we show that it is polynomial-time solvable when both k and ℓ are fixed parameters but that it is NP-complete when only one of those two parameters is fixed. Precisely:

- Let $k \geq 1$ and $\ell \geq 1$ be two fixed integers. Given a graph G as an input, the problem of deciding whether $\lambda_k(G) \leq \ell$ is polynomial-time solvable (in time $n^{O(k\ell)}$) (Theorem 1).
- Let $k \geq 1$ be a fixed integer. Given a graph G with a universal vertex and an integer $\ell \geq 1$ as inputs, the problem of deciding whether $\lambda_k(G) \leq \ell$ is NP-complete (Theorem 2).
- Let $\ell \geq 1$ be a fixed integer. Given a graph G with a universal vertex and an integer $k \geq 1$ as inputs, the problem of deciding whether $\kappa_\ell(G) \leq k$ is NP-complete (Theorem 4).

On the way, we also show that the Relative-Localization Problem is polynomial-time solvable when both k and ℓ are fixed parameters (Theorem 1) but that it is NP-complete when only one of those two parameters is fixed (Theorems 3 and 5).

In Section 3, we then focus on the Localization Problem in the class of trees. Surprisingly, in trees, the complexity of the Localization Problem only comes from the first step. We show that, after the first step, the problem becomes polynomial-time solvable. This allows us to design a polynomial-time approximation algorithm for the problem. More precisely, we show that

- deciding whether $\lambda_k(T) \leq \ell$ is NP-complete in the class of trees T when both k and ℓ are part of the input (Theorem 6);
- there exists an algorithm that computes, in time $O(n \log n)$ (independent of k), a k-strategy for locating a target in at most $\lambda_k(T)+1$ steps in any n-node tree (possibly edge-weighted)(Theorem 9);
- deciding whether $\lambda_k(T) \leq \ell$ can be solved in time $O(n^{k+2} \log n)$ (independent of ℓ) in the class of n-node trees (possibly edge-weighted) (Corollary 1).

2 Complexity of the Localization Problem

This section is devoted to prove that the (Relative) Localization Problem is polynomial-time solvable when both k and ℓ are fixed parameters but that it

is NP-complete when only one of those two parameters is fixed. The proof when ℓ is fixed is an almost straightforward reduction from the Metric Dimension Problem. In the case when k is fixed, it is a much more involved reduction from the 3-dimensional matching Problem. The proof that the (Relative) Localization Problem is in NP is given as a separate claim (Claim 1) as it is used in all of the NP-completeness proofs.

Theorem 1. *Let $k \geq 1$ ($k \geq 2$ for the* Relative Localization *Problem) and $\ell \geq 1$ be two fixed integers. The (*Relative) Localization *Problem is polynomial-time solvable (in time $n^{O(k\ell)}$).*

Proof. Let G be any n-node graph. Let us consider the following tree $\mathcal{T}$ that will be used to represent all possible strategies that probe exactly k vertices per step and last at most ℓ steps in G.

The tree $\mathcal{T}$ is rooted in r and all leaves are at distance 2ℓ from the root. The two types of vertices of $\mathcal{T}$ are labelled by subsets of vertices of $V(G)$. For any vertex $v \in V(\mathcal{T})$ at even distance from r, its label $L(v) \subseteq V(G)$ represents the set of possible locations of the target at this moment. For any vertex $v \in V(\mathcal{T})$ at odd distance from r, its label $L(v) \subseteq V(G)$, of size k, represents the set of vertices that are probed at this moment.

Precisely, $\mathcal{T}$ is defined as follows. Its root r is labelled with $L(r) = V(G)$ (initially, the target may be anywhere). Then, given a vertex $v \in V(\mathcal{T})$ at even distance from r and such that $L(v) = S \subseteq V(G)$, the node v has exactly $\binom{n}{k}$ children labelled by each of the subsets of size k of $V(G)$. Then, for every $Q \in V(G)^k$, let w be the child of v such that $L(w) = Q$. The at most n children of w are defined as follows. Let $(S_1, \cdots, S_q)$ be the partition of S such that, for any $x, y \in S$, the vertices x and y belong to the same S_i if and only if probing the vertices of Q knowing that the target is in S gives the same answer (distance vector) for x and y. Then, w has exactly q children $s_1, \ldots, s_q$ such that $L(s_i) = S_i$ for every $1 \leq i \leq q$. Intuitively, each child of w corresponds to the possible locations of the target in response to the probing of the vertices of Q.

First, note that $|V(\mathcal{T})|$ is polynomial in n when k and ℓ are fixed. Precisely, since $\mathcal{T}$ has at most $(\binom{n}{k}n)^\ell$ leaves (due to the degree of the nodes and the height of $\mathcal{T}$) and all leaves are at distance 2ℓ from r, $|V(\mathcal{T})|$ is upper bounded by $O(2\ell(\binom{n}{k}n)^\ell) = n^{O(k\ell)}$.

Secondly, every strategy (of length ℓ and probing k vertices per turn) is "contained" in $\mathcal{T}$. Indeed, any subtree T' of $\mathcal{T}$ built as follows represents a strategy: start with T' reduced to the root r, then while possible, for any leaf v of T', if v is at an even distance from r, choose a single child of v and add it to T' (this is the probing that the strategy performs in this situation), otherwise, if v is at odd distance from r, add all its children to T'. It is easy to see that, in this way, any strategy, winning (locating the target in at most ℓ turns, wherever it is) or not, can be represented.

By the same reasoning, for every node v at even distance $2(\ell - \ell')$ from r, the subtree of $\mathcal{T}$ rooted in v "contains" all strategies of length ℓ' and probing k vertices per turn, assuming that, initially, the target occupies a vertex in $L(v)$. Let us say that v is *valid* if it contains at least one such winning strategy.

To find out if there is a winning strategy in G, let us proceed by dynamic programming, bottom-up from the leaves of this tree to the root. A leaf v of $\mathcal{T}$ is valid if and only if $L(v)$ is a singleton (indeed, the leaves of $\mathcal{T}$ represent strategies without any probe so the location of the target must be uniquely identified). Then, a vertex v at odd distance from the root is valid if and only if all its children are valid (after a probing, there must be a winning strategy, whatever be the answer). Finally, a vertex v at even distance from the root is valid if and only if at least one of its children is valid. Indeed, the subtree rooted at v contains a winning strategy if, knowing that the target is in $L(v)$, there exists at least one possible probing (one set of k vertices to be probed) that leads toward a winning strategy, whatever be the answer to this probing.

Therefore, there is a winning strategy for G if and only if the root is valid which can be decided in time $|V(\mathcal{T})| = n^{O(k\ell)}$. □

Claim **1** *The (*Relative) Localization *Problem is in* NP.

Proof of claim. The proof is done for the Localization Problem. The certificate is a k-strategy which can be described by a rooted decision tree T as follows. The nodes of T are labelled by sets of k vertices (the vertices to be probed at a given step) and its edges are labelled by sets of vertices representing the possible locations of the target. Precisely, the root node represents the first k vertices to be probed in G according to the k-strategy. For every node $v \in V(T)$ (but the root), the label $L_e \subseteq V(G)$ of the parent-edge e of v represents the current possible locations of the target and the label $L_v \subseteq V(G)$, $|L_v| \leq k$, is the set of vertices to be probed according to the strategy, given that the target occupies a vertex in L_e. Then, every child w of v corresponds to a possible outcome (after probing the vertices in L_v). That is, L_{vw} is the new set of possible locations after having probed L_v (given that the target was in L_e). Note that, clearly, $L_{vw} \subseteq L_e$. Moreover, we may restrict our attention to *progressive* strategies, *i.e.*, strategies for which, for every non-root vertex v with parent-edge e, and for every child-edge f of v, $L_f \subset L_e$. Indeed, otherwise, the vertices probed in L_v are not relevant and a better choice would be any subset containing at least one vertex of L_e (two vertices of L_e in the case of the Relative Localization Problem, where by definition $k \geq 2$, and this is the only part of the proof that differs between the two problems).

The previous remark shows that we can restrict ourselves to k-strategies represented by rooted trees where all non-leaf nodes have at least two children. Moreover, any such tree representing a winning strategy (a k-strategy that locates the target) has exactly $|V(G)|$ leaves since there is a one-to-one correspondence between a path from the root to a leaf of T with the location of the target in G. A trivial induction on $|V(T)|$ allows to show that any rooted tree with n leaves and where all non-leaf nodes have at least two children, has at most $2n$ nodes. Thus, any winning k-strategy may be encoded polynomially and the Localization Problem is in NP. ◇

2.1 When the number k of probed vertices per step is fixed

Let $k \geq 1$ be a fixed integer. The k-Probe Localization Problem takes a graph G and an integer $\ell \geq 1$ as inputs and asks whether $\lambda_k(G) \leq \ell$.

Theorem 2. *Let $k \geq 1$ be a fixed integer. The k-Probe Localization Problem is NP-complete in the class of graphs with a universal vertex.*

Sketch of proof. The problem is in NP by Claim 1. Let us prove it is NP-hard by a reduction from the 3-dimensional matching Problem (3DMP) which is a well known NP-hard problem. The 3DMP takes a set $\mathcal{X} = I_1 \cup I_2 \cup I_3$ of $3n$ elements ($|I_1| = |I_2| = |I_3| = n$) and a set $\mathcal{S}$ of triples $(x, y, z) \in I_1 \times I_2 \times I_3$ as inputs and asks whether there are n triples of $\mathcal{S}$ that are pairwise disjoint.

Let $k \geq 1$ be a fixed integer and let $\mathcal{I} = (\mathcal{X}, \mathcal{S})$ be an instance of 3DMP. First, we may assume that $|\mathcal{X}| = 3kn$ since, if not, it is sufficient to take k disjoint copies of $(\mathcal{X}, \mathcal{S})$. Moreover, we may assume that $m = |\mathcal{S}|$ is such that $2m - 1 \equiv 0 \mod k$ (for instance by adding dummy triples if needed). Let $\mathcal{X} = \{x_1, \ldots, x_{3kn}\}$ and $\mathcal{S} = \{S_1, \ldots, S_m\}$.

Let us build the graph $G = (V, E)$ as follows. Let the vertex-set $V = X \cup X'' \cup S \cup \{s\} \cup \{q\}$ be such that $X = X^1 \cup \cdots \cup X^{k+2}$ with $X^i = \{x^i_1, \ldots, x^i_{3kn}\}$ for every $1 \leq i \leq k+2$; $X'' = \{x''_1, \ldots, x''_{(k+2)m}\}$; and $S = S^1 \cup \cdots \cup S^{k+2}$ with $S^i = \{s^i_j, 1 \leq j \leq m\}$ for every $i \in [\![1, k+2]\!]$. The vertex s is universal (*i.e.*, adjacent to every other vertex), the vertex q is adjacent to every vertex in $X \cup X''$ and, for every $j \in [\![1, 3kn]\!]$ and every $g \in [\![1, m]\!]$ such that $x_j \in S_g$, there is an edge between x^i_j and s^i_g for every $i \in [\![1, k+2]\!]$. Intuitively, X^i is a "copy" of $\mathcal{X}$ and S^i is a "copy" of $\mathcal{S}$ for every $1 \leq i \leq k+2$.

Let $p = \frac{m(k+2)-1}{k} \in \mathbb{N}$. We prove the theorem by showing that $\mathcal{I} = (\mathcal{X}, \mathcal{S})$ admits a 3DM if and only if $\lambda_k(G) \leq (k+2)n + p + 1$. □

The same proof also works for the case with relative distances. Hence,

Theorem 3. *Let $k \geq 2$ be a fixed integer. Given a graph G with a universal vertex and $1 \leq \ell \in \mathbb{N}$, the problem of deciding if $\lambda^{rel}_k(G) \leq \ell$ is NP-complete.*

2.2 When the number ℓ of steps is fixed

Let $\ell \geq 1$ be a fixed integer. The ℓ-Step Localization Problem takes a graph G and an integer $k \geq 1$ as inputs and asks whether $\kappa_\ell(G) \leq k$.

Theorem 4. *Let $\ell \geq 1$ be a fixed parameter. The ℓ-Step Localization Problem is NP-complete in the class of graphs with a universal vertex.*

Sketch of proof. For $\ell = 1$, the result follows from the fact that $\kappa_1(G)$ is exactly the metric dimension and from its NP-completeness [8].

Let $\ell \geq 2$ be fixed. The problem is in NP by Claim 1. To prove the NP-hardness, let us reduce the Metric Dimension Problem restricted to the class of graphs that contain a universal vertex, which is known to be NP-hard [11]. Let $< G, k >$ be an instance of Metric Dimension where G contains a universal

vertex. We construct, in polynomial time, an instance $< G', k >$ of the ℓ-Step Localization Problem such that $MD(G) \leq k$ if and only if $\kappa_\ell(G') \leq k$.

The construction of G' is as follows. Start from $k(\ell-1)+1$ disjoint copies $G_1, \ldots, G_{k(\ell-1)+1}$ of G. Let v be a universal vertex of G, and for $1 \leq i \leq k(\ell-1)+1$, let v_i denote the copy of v in G_i. Finally, add a universal vertex u to the graph. □

A similar proof (but based on a reduction of Centroidal Dimension) works for the case with relative distances. Hence,

Theorem 5. *Let $\ell \geq 1$ be a fixed integer. Given a graph G with a universal vertex and $2 \leq k \in \mathbb{N}$, the problem of deciding if $\kappa_\ell^{rel}(G) \leq k$ is NP-complete.*

3 The Localization Problem in trees

This section is devoted to the study of the Localization Problem in the class of trees. Note that, if the number of steps is $\ell = 1$, the problem is equivalent to the one of Metric dimension which can easily be solved in polynomial-time in trees [13, 20]. We first show that, if k and ℓ are part of the input, deciding whether $\lambda_k(T) \leq \ell$ is NP-complete in the class of trees T. Our reduction actually shows that the difficulty of the problem comes from the choice of the vertices to be probed during the first step. Surprisingly, we show that the first step is actually the only source of complexity. More precisely, our main result is that, if the first step is given (intuitively, either given by an oracle or imposed by an adversary), then an optimal strategy (according to this first pre-defined step) can be computed in polynomial-time. This allows us to design a $(+1)$-approximation algorithm for the Localization Problem in trees and to prove that, in contrast with general graphs (Theorem 2), the k-Probe Localization Problem is polynomial-time solvable in the class of trees (Corollary 1).

3.1 NP-hardness

Theorem 6. *The Localization Problem is NP-complete in the class of trees.*

Sketch of proof. Again, the problem is in NP by Claim 1. To prove the NP-hardness, let us reduce the Hitting-Set Problem. The inputs are an integer $k \geq 1$, a ground-set $B = (b_1, \ldots, b_n)$, and a set $\mathcal{S} = \{S_1, \ldots, S_m\}$ of subsets of B, *i.e.*, $S_i \subseteq B$ for every $i \leq m$. The Hitting-Set Problem aims at deciding if there exists a set $H \subseteq B$, $|H| \leq k$, and $H \cap S_i \neq \emptyset$ for every $1 \leq i \leq m$.

Adding one new element to the ground-set and adding this element to one single subset clearly does not change the solution. Therefore, by adding some dummy elements (each one belonging to a single subset), we may assume that all the subsets are of the same size σ and that $\sigma - 1 \equiv 0 \mod k$.

Let γ be any integer such that $\gamma - 1 \equiv 0 \mod k$ and $\gamma > n - k - 1$.

The instance T of the Localization Problem is built as follows. Let us start with n vertex-disjoint paths $B_1, \cdots, B_n$ (the *branches*) of length $2m$, where

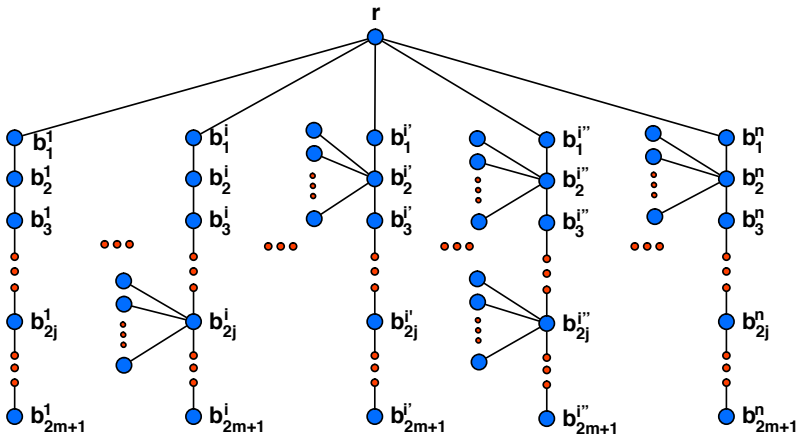

Fig. 1: An example of a tree T built from an instance $(k, B, \mathcal{S})$ of HITTING SET in the proof of Thm. 6. The elements $b_{i'}, b_{i''}$, and b_n belong to the set S_1 (but not the elements b_1 and b_i) as figured by the three "stars" at level 2. The elements b_i and $b_{i''}$ belong to S_j (stars at level $2j$), but not the elements $b_1, b_{i'}$, and b_n.

$B_i = (b^i_1, \ldots, b^i_{2m+1})$ for each $1 \leq i \leq n$. Then, let us add one new vertex c adjacent to b^i_1 for all $1 \leq i \leq n$. For every $1 \leq j \leq m$ and for every $1 \leq i \leq n$ such that $b_i \in S_j$, let us add γ new vertices adjacent to b^i_{2j}. The subgraph induced by b^i_{2j} and by the γ leaves adjacent to it is referred to as the *star* representing the element i in the set S_j (or representing the set S_j in the branch i). The obtained tree T is depicted in Figure 1.

Intuitively, it will always be better for the target to be located in a leaf of some star because γ is "huge". During the first turn of any strategy, the *level* (roughly, the distance to the root) of the target can be identified. Each even level $2j$ corresponds to a set S_j. If, during the first turn, one star corresponding to each even level can be eliminated from the possible locations (which corresponds to hitting every subset), then the strategy finishes one step earlier than if all subsets cannot be hit (if so, all stars would have to be checked).

Precisely, we show that $\lambda_k(T) \leq 1 + \frac{\sigma-1}{k} + \frac{\gamma-1}{k}$ if and only if there is a hitting set of size at most k. □

3.2 Algorithm in trees

The proof above actually shows that, in our reduction, choosing the vertices to be probed during the first step to ensure an optimal strategy is equivalent to finding a minimum hitting set. We show below that the first step is actually the only "part" of the problem that is difficult.

The key argument is the following easy remark. Let us consider a tree T where a target is hidden and assume that a single vertex $r \in V(T)$ is probed.

After this single probe, the distance $d \in \mathbb{N}$ between the target and r is known. Therefore, from the second step, the instance becomes equivalent to a tree T' (a subtree of T) rooted in r, with all leaves the same distance d from r, and where the target is known to occupy some leaf of T'. We first present an algorithm that computes in polynomial-time (independent of k and ℓ) an optimal strategy to locate the target in such instances.

Let $\mathcal{T}$ be the set of rooted trees with all leaves the same distance from the root. Given a rooted tree $(T, r) \in \mathcal{T}$ (in what follows, we omit r when it is clear from the context), let $\lambda_k^L(T)$ be the minimum integer h such that there exists a k-strategy that locates a target in at most h steps knowing *a priori* that the target occupies some leaf of T. The next claim is a key argument for why the problem is easier in the class $\mathcal{T}$ when the target is known to occupy a leaf.

Claim **2** *Let $(T, r) \in \mathcal{T}$ rooted in r, let v be a child of r and T_v the subtree rooted in v. If the target is known to occupy a leaf of T, then probing any vertex in T_v allows to learn if the target occupies a leaf of T_v or a leaf of $T \setminus T_v$.*

Proof of claim. Let d be the distance between r and the leaves of T. Let w be any vertex of T_v and let d' be the distance between w and r. The target occupies a leaf of T_v if and only if its distance to w is strictly less than $d + d'$. $\diamond$

Let $T \in \mathcal{T}$ rooted in r, let v be a child of r, and let us assume that the secret location of the target is some leaf of T_v. Note that $(T_v, v) \in \mathcal{T}$. Let us assume that T_v is not a path and let s be the first step of an optimal strategy ϕ in T that probes some vertex of T_v (such a step s must exist since otherwise the target would never be detected in T_v). By Claim 2, it is sufficient to probe a single vertex of T_v to learn whether the target occupies a leaf of T_v. Then, applying an optimal strategy ϕ_v in T_v will locate the target in a total of $s + \lambda_k^L(T_v) - 1$ steps if the first step of ϕ_v only requires probing a single vertex of T_v and $s + \lambda_k^L(T_v)$ steps otherwise. So, it may be possible to do better. Indeed, probing several vertices of T_v during the s^{th} step of ϕ may serve not only to detect the target in T_v but also to "play" the first step of ϕ_v. Doing so, the strategy will take only $s + \lambda_k^L(T_v) - 1$ steps. So, elaborating, an optimal strategy will consist of doing a tradeoff between probing one single vertex in a subtree (and detecting "quickly" in which subtree the target is hidden since several subtrees are considered simultaneously) and probing more vertices in a subtree in order to get a head start for the strategy in the case the target is in this subtree.

For any tree T, let $\pi(T)$ be the minimum integer q such that there exists a k-strategy that locates a target in at most $\lambda_k^L(T)$ steps, knowing *a priori* that the target occupies some leaf of T, and such that at most q vertices are probed during the first step.

To illustrate the need of a tradeoff, let us consider the following simple example utilizing π. Consider two children v_1 and v_2 of r such that $(\lambda_k^L(T_{v_1}), \pi(T_{v_1})) = (6, 4)$ and $(\lambda_k^L(T_{v_2}), \pi(T_{v_2})) = (6, 3)$. Let $k = 6$. Then, at the first step, we cannot probe $\pi(T_{v_1}) + \pi(T_{v_2}) = 7$ vertices. W.l.o.g., let us assume that at most $3 < \pi(T_{v_1})$ vertices of T_{v_1} have been probed during the first step. Thus, by defi-

Algorithm 1 $\mathcal{A}_1(k,(T,r))$.

Require: An integer k and a tree $T \in \mathcal{T}$ rooted in r with children $v_1,\ldots,v_{d^*}$
Ensure: $(\lambda_k^L(T),\pi(T))$
1: **if** (T,r) is a rooted path **then**
2: **return** $(0,0)$
3: **for** $i=1$ to d^* **do**
4: Let $(\lambda_i,\pi_i)=\mathcal{A}_1(k,(T[i],v_i))$
5: Sort the $(\lambda_i,\pi_i)_{1\leq i\leq d^*}$ in non-increasing lexicographical order
6: **return** $\mathcal{A}_2(k,(T,r),(\lambda_i,\pi_i)_{1\leq i\leq d^*})$

nition of π, a total of $\lambda_k^L(T_{v_1})+1=7$ steps are necessary if we learn at the first step that the target occupies some leaf of T_{v_1}.

Let $T\in\mathcal{T}$ rooted in r and let $v_1,\ldots,v_{d^*}$ be the children of r. From previous arguments, the computation of an optimal strategy for T consists of determining, for each subtree T_{v_i} $(1\leq i\leq d^*)$, the first step for which a vertex of T_{v_i} will be probed (if the target has not been located in a different subtree at a previous step). If 1 vertex is probed during this step, then $\lambda_k^L(T_{v_i})$ extra steps are needed if the target occupies some leaf of T_{v_i} (unless $\pi(T_{v_i})=1$ in which case $\lambda_k^L(T_{v_i})-1$ extra steps are needed). If $\pi(T_{v_i})$ vertices of T_{v_i} are probed during this step, then $\lambda_k^L(T_{v_i})-1$ extra steps are needed if the target occupies some leaf of T_{v_i}.

Description of Algorithm 1. The main algorithm $\mathcal{A}_1(k,(T,r))$ takes an integer $k\geq 1$ and a rooted tree $(T,r)\in\mathcal{T}$ as inputs and computes $(\lambda_k^L(T),\pi(T))$ and a corresponding k-strategy. It proceeds bottom-up by dynamic programming from the leaves to the root. Precisely, let $v_1,\ldots,v_{d^*}$ be the children of r. For any $1\leq i\leq j\leq d^*$, let $T[i]=T_{v_i}$ be the subtree rooted at v_i, and let $T[i,j]=\{r\}\cup T_{v_i}\cup\cdots\cup T_{v_j}$ $(T[i,j]=\emptyset$ if $i>j)$. To lighten the notations, let us set $\lambda_i=\lambda_k^L(T[i])$ and $\pi_i=\pi(T[i])$ for every $1\leq i\leq d^*$. Assume that, $(\Lambda,\Pi)=(\lambda_i,\pi_i)_{1\leq i\leq d^*}$ have been computed recursively and sorted in non-increasing lexicographical order. Then, $\mathcal{A}_2(k,(T,r),(\Lambda,\Pi))$, described in Algorithm 2, takes the integer $k\geq 1$, the rooted tree $(T,r)\in\mathcal{T}$, and the sorted tuple (Λ,Π) as inputs and computes $(\lambda_k^L(T),\pi(T))$ and a corresponding strategy.

Description of Algorithm 2. We now informally describe $\mathcal{A}_2(k,(T,r),(\Lambda,\Pi))$. First, Line 2 to Line 5 deals with the subtrees $T_{v_{d+1}},\ldots,T_{v_d^*}$ that are rooted paths (path rooted at one of its vertices of degree one, the other vertex is the leaf). In other words, it concerns all the subtrees T_{v_i} such that $(\lambda_i,\pi_i)=(0,0)$. Indeed, this case is somehow pathologic. Claim 3 proves that Line 2 to Line 5 computes $(\lambda_k^L(T[v_{d+1},d^*]),\pi(T[v_{d+1},d^*]))$. Let us define $\mathcal{S}\subset\mathcal{T}$ as the set of subdivided stars S (*i.e.*, trees with at most one vertex of degree at least 3) with all leaves the same distance from the root, where the root of S is the (unique) vertex with degree >2 or one of the two ends if S is a path.

Claim **3** *Let $S\in\mathcal{S}$ and let δ be the degree of the root r. Then, $\lambda_k^L(S)=\lceil\frac{\delta-1}{k}\rceil$ and $\pi(S)=-k(\lceil\frac{\delta-1}{k}\rceil-\lceil\frac{\delta-1}{\delta}\rceil)+(\delta-1)$.*

Algorithm 2 $\mathcal{A}_2(k,(T,r),(\Lambda,\Pi))$.

Require: $k \in \mathbb{N}^*$, a rooted tree (T,r) with $v_1,\ldots,v_{d^*}$ the children of r such that $(\Lambda,\Pi)=(\lambda_i,\pi_i)_{1\leq i\leq d^*}$ is sorted in non-increasing lexicographical order.

1: $l \leftarrow 1$, $p \leftarrow k$, $d \leftarrow d^*$
2: **if** $T[d^*]$ is a rooted path **then**
3: $\quad d \leftarrow z$ with $0 \leq z < d^*$ the smallest integer such that $T[z+1]$ is a rooted path
4: $\quad l \leftarrow 1 + \lceil \frac{d^*-d-1}{k} \rceil$
5: $\quad p \leftarrow k + k(\lceil \frac{d^*-d-1}{k} \rceil - \lceil \frac{d^*-d-1}{d^*-d} \rceil) - (d^*-d-1)$
6: **for** $i = d$ down to 1 **do**
7: $\quad$ **if** $p = 0$ or $l < \lambda_i + 1$ **then**
8: $\quad\quad p \leftarrow k$, $l \leftarrow \max(l+1, \lambda_i+1)$
9: $\quad \alpha \leftarrow \pi_i - (\pi_i - 1)\lceil (l-(\lambda_i+1))/l \rceil$
10: $\quad$ **if** $\alpha \leq p$ **then**
11: $\quad\quad p \leftarrow p - \alpha$
12: $\quad$ **else**
13: $\quad\quad p \leftarrow k-1$, $l \leftarrow l+1$
14: **return** $(l-1, k-p)$

We are now able to detail the second part of the algorithm (from Line 6). Informally, $\mathcal{A}_2(k,(T,r),(\Lambda,\Pi))$ recursively builds, for $i = d$ down to 1, an optimal k-strategy ϕ for $T[i,d^*]$ from an optimal k-strategy ϕ' of $T[i+1,d^*]$ and from an optimal k-strategy ϕ'' of $T[i]$ (the latter one being given as input through (λ_i,π_i)). In other words, $(\lambda_k^L(T[i,d^*]), \pi(T[i,d^*]))$ is computed from $(\lambda_k^L(T[i+1,d^*]), \pi(T[i+1,d^*]))$ and (λ_i,π_i). For every $1 \leq i \leq d+1$, let l_i (resp., p_i) denote the value of l (resp. of p) just before the $(d+2-i)^{th}$ iteration of the for-loop (so, l_1 and p_1 are the final values of l and p). Intuitively, let us assume that an optimal strategy for $T[i+1,d^*]$ has been computed, takes at most $l_{i+1}-1$ steps and requires $k - p_{i+1} = \pi(T[i+1,d^*])$ vertices to be probed during its first step. Roughly, there are five cases to be considered.

- If $\pi_i \leq p_{i+1}$ and $\lambda_i = l_{i+1} - 1$, the strategy ϕ follows ϕ' but, in addition, probes π_i vertices of $T[i]$ during its first step. If the target is in $T[i]$, then ϕ follows ϕ'' (and takes a total of at most λ_i steps), otherwise, it proceeds as ϕ' (and takes a total of at most $l_{i+1}-1$ steps). We get $l_i = l_{i+1}$ and $p_i = p_{i+1} - \pi_i$.
- Else if $\pi_i > p_{i+1} > 0$ and $\lambda_i = l_{i+1} - 1$, the first step of ϕ probes a unique vertex in $T[i]$. If the target is in $T[i]$, then ϕ follows ϕ'' (and takes a total of at most $\lambda_i + 1$ steps). Otherwise, it proceeds as ϕ' (and takes a total of at most l_{i+1} steps). We get $l_i = l_{i+1} + 1$ and $p_i = k - 1$.
- Else, if $p_{i+1} = 0$ and $\lambda_i \leq l_{i+1} - 1$, the first step of ϕ probes a unique vertex in $T[i]$. If the target is in $T[i]$, then ϕ follows ϕ'' (and takes a total of at most $\lambda_i + 1$ steps). Otherwise, it proceeds as ϕ' (and takes a total of at most l_{i+1} steps). We get $l_i = l_{i+1} + 1$ and $p_i = k - 1$.
- Else, if $\lambda_i < l_{i+1} - 1$ and $p_{i+1} > 0$, the strategy ϕ follows ϕ' but, in addition, probes one vertex of $T[i]$ during its first step. If the target is in $T[i]$, then ϕ follows ϕ'' (and takes a total of at most $\lambda_i + 1$ steps), otherwise, it proceeds

as ϕ' (and takes a total of at most $l_{i+1} - 1$ steps). We get $l_i = l_{i+1}$ and $p_i = p_{i+1} - 1$.
- Else ($\lambda_i > l_{i+1} - 1$), the strategy ϕ probes π_i vertices in $T[i]$ during the first step. If the target is in $T[i]$, then ϕ follows ϕ'' (and takes a total of at most λ_i steps), otherwise, it proceeds as ϕ' (and takes a total of at most l_{i+1} steps). We get $l_i = \lambda_i + 1$ and $p_i = k - \pi_i$.

As the subtrees are sorted in non-increasing lexicographical order (of (λ_i, π_i)), we prove in Lemma 1 that the strategy ϕ described before is optimal for $T[i, d^*]$, that is, it computes $(\lambda_k^L(T[i, d^*]), \pi(T[i, d^*]))$.

Lemma 1. *For every* $1 \leq i \leq d+1$, $\lambda_k^L(T[i, d^*]) = l_i - 1$ *and* $\pi(T[i, d^*]) = k - p_i$.

Correctness and complexity of Algorithm 1 and Algorithm 2. We prove in Theorem 8 that $\mathcal{A}_1(k, (T, r))$ computes $(\lambda_k^L(T), \pi(T))$ and a corresponding k-strategy in time $O(n \log n)$, where n is the number of vertices. To do that, Theorem 7 proves the correctness and the linear (in the number of children of r) time complexity of $\mathcal{A}_2(k, (T, r), (\Lambda, \Pi))$.

Theorem 7. *Let* $k \geq 1$, *let* $(T, r) \in \mathcal{T}$ *be a rooted tree, and let* $v_1, \ldots, v_{d^*}$ *be the children of* r *such that the tuples* $(\Lambda, \Pi) = (\lambda_i, \pi_i)_{1 \leq i \leq d^*}$ *are sorted in non-increasing lexicographical ordering. Then,* $\mathcal{A}_2(k, (T, r), (\Lambda, \Pi))$ *returns* $(\lambda_k^L(T), \pi(T))$ *and a corresponding strategy. Furthermore, the time-complexity of* $\mathcal{A}_2$ *is* $O(d^*)$ *(independent of* k*).*

Proof. The time-complexity is obvious and the correctness follows from Lemma 1 for $i = 1$. The fact that the strategy is also returned is not explicitly described in Algorithm 2 but directly follows from the proof of Lemma 1. □

Theorem 8. *Let* $k \geq 1$, *and let* $(T, r) \in \mathcal{T}$ *be an* n*-node rooted tree. Then,* $\mathcal{A}_1(k, (T, r))$ *returns* $(\lambda_k^L(T), \pi(T))$ *and a corresponding strategy. Furthermore, the time-complexity of* $\mathcal{A}_1$ *is* $O(n \log n)$ *(independent of* k*).*

Proof. The correctness is simply proved by induction and by Theorem 7. For the time-complexity, at every recursive call on a subtree T_v rooted at v (with d_v children), the additional number of operations is $O(d_v \log d_v)$ (sorting) plus $O(d_v)$ (Algorithm $\mathcal{A}_2$, by Theorem 7). Since in a tree, $\sum_{v \in V(T)} d_v = 2(n-1)$, this gives a total complexity of $O(\sum_{v \in V(T)} d_v \log d_v) = O(n \log n)$. Again, the strategy is not explicit in our presentation but can be easily computed. □

Main results. From $\mathcal{A}_1(k, (T, r))$ presented before, it is easy to get an efficient approximation algorithm when k and ℓ are part of the input and a polynomial-time algorithm when k is fixed.

Theorem 9. *There exists an algorithm that, given any integer* $k \geq 1$ *and any* n*-node tree* T*, computes a* k*-strategy that locates a target in* T *in at most* $\lambda_k(T) + 1$ *steps. Furthermore, the time-complexity of the algorithm is* $O(n \log n)$.

Proof. The strategy proceeds as follows. The first step probes any arbitrary vertex r of T. Let d be the distance between r and the target, let $L \subseteq V(T)$ be the set of vertices at distance exactly d from r, and let T^d be the subtree induced by r and every vertex on a path between r and the vertices in L. Note that $(T^d, r) \in \mathcal{T}$ and that the target is occupying a leaf of T^d. Hence, it is sufficient to apply $\mathcal{A}_1(k, (T^d, r))$. By Theorem 8, the above strategy will locate the target in at most $1 + \max_d \lambda_k^L(T^d) \leq 1 + \lambda_k(T)$ steps. □

Corollary 1. *There exists an algorithm that, given any integer $k \geq 1$ and any n-node tree T, computes an optimal k-strategy for locating a target in T in at most $\lambda_k(T)$ steps. Furthermore, the time-complexity of the algorithm is $O(n^{k+2} \log n)$.*

4 Further Work

Our results in trees leave the open question of whether $\lambda_k(T)$ is Fixed Parameter Tractable (in k) in the class of n-node trees T. Moreover, it would be interesting to study the LOCALIZATION Problem in other graph classes such as interval graphs and planar graphs. Also, what is the complexity of the ℓ-STEP LOCALIZATION Problem in trees?

The RELATIVE-LOCALIZATION Problem is much more intricate even for simple topologies. A first step towards a better understanding of this problem would be to fully solve it in the case of paths (*i.e.*, to determine $\kappa_1^{rel}(P)$ for every path P), which has been partially solved in [9], before studying it in the class of trees.

References

1. Rémy Belmonte, Fedor V. Fomin, Petr A. Golovach, and M. S. Ramanujan. Metric dimension of bounded tree-length graphs. *SIAM J. Discrete Math.*, 31(2):1217–1243, 2017.
2. Y. Ben-Haim, S. Gravier, A. Lobstein, and J. Moncel. Adaptive identification in graphs. *Journal of Combinatorial Theory, Series A*, 115(7):1114–1126, 2008.
3. J. Bensmail, D. Mazauric, F. Mc Inerney, N. Nisse, and S. Pérennes. Sequential metric dimension. Technical report, INRIA, 2018. RR, https://hal.archives-ouvertes.fr/hal-01717629.
4. Bartlomiej Bosek, Przemyslaw Gordinowicz, Jaroslaw Grytczuk, Nicolas Nisse, Joanna Sokól, and Malgorzata Sleszynska-Nowak. Centroidal localization game. *CoRR*, abs/1711.08836, 2017.
5. Bartlomiej Bosek, Przemyslaw Gordinowicz, Jaroslaw Grytczuk, Nicolas Nisse, Joanna Sokól, and Malgorzata Sleszynska-Nowak. Localization game on geometric and planar graphs. *CoRR*, abs/1709.05904, 2017.
6. A. Brandt, J. Diemunsch, C. Erbes, J. LeGrand, and C. Moffatt. A robber locating strategy for trees. *Discrete Applied Mathematics*, 232:99 – 106, 2017.
7. James M. Carraher, Ilkyoo Choi, Michelle Delcourt, Lawrence H. Erickson, and Douglas B. West. Locating a robber on a graph via distance queries. *Theor. Comput. Sci.*, 463:54–61, 2012.
8. Josep Díaz, Olli Pottonen, Maria J. Serna, and Erik Jan van Leeuwen. Complexity of metric dimension on planar graphs. *J. Comput. Syst. Sci.*, 83(1):132–158, 2017.
9. Florent Foucaud, Ralf Klasing, and Peter J. Slater. Centroidal bases in graphs. *Networks*, 64(2):96–108, 2014.
10. Florent Foucaud, George B. Mertzios, Reza Naserasr, Aline Parreau, and Petru Valicov. Identification, location-domination and metric dimension on interval and permutation graphs. i. bounds. *Theor. Comput. Sci.*, 668:43–58, 2017.
11. Florent Foucaud, George B. Mertzios, Reza Naserasr, Aline Parreau, and Petru Valicov. Identification, location-domination and metric dimension on interval and permutation graphs. II. algorithms and complexity. *Algorithmica*, 78(3):914–944, 2017.
12. M. R. Garey and D. S. Johnson. *Computers and Intractability - A guide to NP-completeness*. W.H. Freeman and Company, 1979.
13. Frank Harary and Robert A. Melter. On the metric dimension of a graph. *Ars Combinatoria*, 2:191–195, 1976.
14. Sepp Hartung and André Nichterlein. On the parameterized and approximation hardness of metric dimension. In *Proceedings of the 28th Conference on Computational Complexity, CCC*, pages 266–276. IEEE Computer Society, 2013.
15. John Haslegrave, Richard A. B. Johnson, and Sebastian Koch. Locating a robber with multiple probes. *Discrete Mathematics*, 341(1):184–193, 2018.
16. Mark G. Karpovsky, Krishnendu Chakrabarty, and Lev B. Levitin. On a new class of codes for identifying vertices in graphs. *IEEE Trans. Information Theory*, 44(2):599–611, 1998.
17. Suzanne M. Seager. Locating a robber on a graph. *Discrete Mathematics*, 312(22):3265–3269, 2012.
18. Suzanne M. Seager. A sequential locating game on graphs. *Ars Comb.*, 110:45–54, 2013.
19. Suzanne M. Seager. Locating a backtracking robber on a tree. *Theor. Comput. Sci.*, 539:28–37, 2014.

20. Peter J. Slater. Leaves of trees. pages 549–559. Congressus Numerantium, No. XIV, 1975.
21. Peter J. Slater. Domination and location in acyclic graphs. *Networks*, 17(1):55–64, 1987.