# Budgeted sensor placement for source localization on trees

L. E. Celis [a,1] F. Pavetić [b,2] B. Spinelli [a,3] P. Thiran [a,4]

[a] *School of Computer and Communication Sciences, EPFL, Lausanne, Switzerland*

[b] *Faculty of Electrical Engineering and Computing, University of Zagreb, Croatia*

**Abstract**

We address the problem of choosing a fixed number of sensor vertices in a graph in order to detect the source of a partially-observed diffusion process on the graph itself. Building on the definition of *double resolvability* we introduce a notion of *vertex resolvability*. For the case of tree graphs we give polynomial time algorithms for both finding the sensors that maximize the probability of correct detection of the source and for identifying the sensor set that minimizes the expected distance between the real source and the estimated one.

*Keywords:* Double Resolvability, Sensor Placement, Source Localization.

[1] Email: elisa.celis@epfl.ch.

[2] Email: fpavetic@google.com. The work has been done during the author's enrollment in the Faculty of Electrical Engineering and Computing, University of Zagreb, Croatia and prior to the current employment by Google Switzerland GmbH.

[3] Email: brunella.spinelli@epfl.ch. The author was supported in part by the Bill & Melinda Gates Foundation, under Grant No. OPP1070273.

[4] Email: patrick.thiran@epfl.ch.

# 1 Introduction

The problem of source localization has received considerable attention in the last years. Most approaches, starting with the seminal work by Shah and Zaman [1], rely on knowing the state of the entire network at a given instant in time. Pinto [2] introduced a model that instead, estimates the source based on a sparse set of sensor vertices in the graph. However, the set of sensors is often assumed to be given. In this work we consider the problem of selecting *which* vertices to observe given a *budget* on the number (or cost) of allowed sensors in order to optimize source detection.

Consider a graph $\mathcal{G}(V, E)$ with weighted edges. We say that a vertex gets *infected* when it is reached by a diffusion process on the graph; the moment at which this happens is called the *infection time*. Given a budget $k \in \mathbb{N}$, we are interested in finding $S \subseteq V$ of size $k$ such that the infection times of the vertices of $S$ maximize the accuracy in source identification. Specifically, our algorithm allows us to identify a set of *candidate source vertices*.

Depending on the context in which we want to localize the source of a diffusion, we could be more interested in maximizing the chances of an exact identification of the source or in minimizing, in average, the distance between the real source $s^*$ and the estimated source $\hat{s}$. Hence, assuming that $s^*$ can randomly appear in $V$, we consider two metrics: (i) the *error probability*, i.e., $\mathcal{P}_e = \mathcal{P}(\hat{s} \neq s^*)$; (ii) the *expected distance between the real source* $s^*$ *and the estimated source* $\hat{s}$, i.e., $\mathbb{E}[d(s^*, \hat{s})]$, where $d$ is the weighted distance between two vertices in the graph. It is easy to see that the two metrics may require different sets of sensors.

An approximation algorithm for minimising the cardinality of the sensor set that *perfectly* detects the source was given in [3] using the connection to the Doubly Resolving Set (DRS) of a graph [5]. A similar question was first considered by [4] when the *starting time* of the diffusion is known. In both cases, budget constraints are not considered.

Based on the definition of DRS, we introduce a concept of *vertex resolvability* and show that the performance of a set of sensor vertices with respect to the error probability is directly linked to the number of *unresolved* vertices. For the case of a tree graph $\mathcal{T}$ of size $n$, we design an $O(nk^2)$ dynamic-programming algorithm to find $k$ sensor vertices that minimize the number of unresolvable vertices and hence the error probability. Minimizing the expected error distance, was, to the best of our knowledge, never considered before. Also for this metric, we show that if $\mathcal{G}$ is a tree, an optimal set $S$ can be found with a polynomial-time algorithm.

## 2 Preliminaries

Consider a weighted graph $\mathcal{G}(V, E)$ that models a contact network. A diffusion process on $\mathcal{G}$ is started by a single unknown vertex $s^*$, the *source*, at an unknown time $t^*$. If a vertex $v$ becomes infected at time $t_v$, each non-infected neighbor $u$ of $v$ gets infected at time $t_u = t_v + w_{u,v}$ where $w_{u,v} \in \mathcal{R}_+$ is the weight of edge $(u, v)$. For every vertex $s$ in the *sensor set* $S$ the infection time $t_s$ is known. We estimate the position of the source based only on the infection times $\{t_s, s \in S\}$. The time at which the diffusion starts is unknown, hence a single observed infection time does not give any information about the position of the source. We choose one sensor, say $s_1 \in S$ as *reference point* and define a vector of relative observed times as follows.

**Definition 2.1** [observation vector] Let $\mathcal{G}$ be a graph, $S \subseteq V$, $|S| = k$ a set of sensors and $\{t_s, s \in S\}$ the infection times observed during a diffusion process. Then $\boldsymbol{\tau} \in \mathbb{R}^{k-1}$ where $\tau_i = t_{s_{i+1}} - t_{s_1}$, $i \in [k-1]$, is the observation vector associated to the diffusion process.

**Definition 2.2** [distance vector] Let $\mathcal{G}$ be a graph, $S \subseteq V$, $|S| = k$ a set of sensors, for each candidate source $s$ we define its distance vector as $\mathbf{d}_s$ where $\mathbf{d}_{s,i} = d(s_{i+1}, s) - d(s_1, s)$, $i \in [k-1]$, and $d$ is the weighted graph distance.

**Definition 2.3** [resolved / unresolved vertex] A vertex $u$ is resolved by a set $S$ if $\mathbf{d}_u \neq \mathbf{d}_v$ for all $v \in V$, $v \neq u$, and unresolved otherwise.

Note that $u \sim v$ iff $\mathbf{d}_u = \mathbf{d}_v$ is an equivalence relation and we call $[u]_S$ the class of vertices equivalent to $u$. A set $S$ such that $[u]_S = \{u\}$ for each $u \in V$ is a Double Resolving Set, as clarified by the definition and lemma below.

**Definition 2.4** [Double Resolvability] Given a graph $\mathcal{G}$, $S \subseteq V$ is said to doubly resolve $\mathcal{G}$ if for any $x, y \in V$ there exist $u, v \in S$ s.t. $d(x, u) - d(x, v) \neq d(y, u) - d(y, v)$. Such a subset $S$ is a Double Resolving Set for $\mathcal{G}$ (DRS).

**Lemma 2.5 (Lemma 3.1 in [3])** *Let $S \subseteq V$ and fix $s \in S$. Then every vertex $u \in V$, $u \neq s$ resolved by $S$ is resolved by a pair in $\{(s, v) : v \in S \backslash \{s\}\}$.*

As a consequence of Lemma 2.5, the definition of resolved and unresolved vertices above does not depend on the choice of *reference point* $s_1 \in S$.

**Lemma 2.6** *Let $\mathcal{T}$ be a tree with $n$ vertices, $S \subseteq V$.*

(i) *Let $u \in S$ a non-leaf vertex or $u \in V \backslash S$. If there do not exist $s_1, s_2 \in S$,*

$s_1, s_2 \neq u$, *s.t.* $u \in \mathcal{P}(s_1, s_2)$ [5], *then* $u$ *is not resolved by* $S$*;*

(ii) *let* $u \in V$ *a non-leaf vertex and root* $\mathcal{T}$ *at* $u$*.* $u$ *is resolved if and only if every subtree* $\mathcal{T}_c$ *rooted at a child* $c$ *of* $u$ *contains at least one vertex of* $S$*;*

(iii) *let* $|S| > 1$*: a leaf-vertex* $\ell$ *is resolved if and only if* $\ell \in S$*.*

If the source of the diffusion is $s^*$ and the observation vector is $\boldsymbol{\tau}$, then all vertices in $[s^*]$ are *candidate source vertices* because their distance vectors are equal to $\boldsymbol{\tau}$. Given a prior distribution $\pi$ on $V$ for the position of $s^*$, we select an approximated source $\hat{s}$ by sampling the conditional distribution $\pi|_{[s^*]}$.

**Remark 2.7** On trees, this model and estimator tolerate a uniformly bounded amount of noise in the transmission delays: in fact, the estimation of the source would have the same accuracy if for a vertex $u$ infected by its neighbor $v$, $t_u = t_v + w_{u,v} + X_{u,v}$ where $X_{u,v} \in [-\varepsilon, \varepsilon]$ is a random variable and $\varepsilon < \min_{(u,v)\in E}[w_{u,v}/\text{diameter}(\mathcal{T})]$.

## 3 Error Probability Minimization

**Proposition 3.1** *Let* $\mathcal{G}$ *be a graph of size* $n$*,* $S \subseteq V$ *and uniform prior* $\pi$*. The probability of error* $\mathcal{P}_e(S)$ *is given by* $\mathcal{P}_e(S) = \frac{1}{n}\sum_{[u]_S \subseteq V}(|[u]_S| - 1)$*.*

If $q$ is the number of equivalence classes we have $\mathcal{P}_e = 1 - q/n$ and it is clear that the error probability is minimized if the number of equivalence classes is maximized. Looking back to Lemma 2.6, if the graph is a tree $\mathcal{T}$, $\mathcal{P}_e$ is 0 if a sensor is placed on each leaf. [6] In fact, the minimum $k$ required for $\mathcal{P}_e = 0$ is the number of leaves $\ell$. Moreover, if $k < \ell$, the vertices that minimize $\mathcal{P}_e$ are a subset of the leaves of $\mathcal{T}$. This suggests that, given a tree and a sensor set, if we root the tree at an arbitrary vertex it is possible to compute $\mathcal{P}_e$ as the sum of the probabilities of error of the different subtrees. Building on this observation we prove that, for any $n$-vertex tree $\mathcal{T}$ and budget $k \in \mathbb{N}$, a set $S^k_{opt}$ that minimizes $\mathcal{P}_e$ can be found with a recursive algorithm of total complexity $O(k^2 n)$.

**Theorem 3.2** *Let* $\mathcal{T}$ *be a tree with* $n$ *vertices and* $\ell$ *leaves and let the prior* $\pi$ *be uniform. If* $k \geq \ell$*, the leaf set is an optimal sensor set. If* $k \in [\ell - 1]$*, there exists an algorithm that finds* $S^k_{opt} \in \text{argmin}_{|S|=k}\mathcal{P}_e(S)$ *in time* $O(nk^2)$*.*

[5] $\mathcal{P}(s_1, s_2)$ denotes the unique shortest path between $s_1$ and $s_2$ on a tree $\mathcal{T}$.

[6] See also [3] for a different proof.

**Proof.** [Correctness] The statement is trivial for $k \geq \ell$ as the set of leaves resolves all the vertices. If $k < \ell$, call $\mathcal{T}_r$ the tree obtained rooting $\mathcal{T}$ at an arbitrary non-leaf vertex $r$. We claim that $S^k_{opt}$ is obtained through the main function of Algorithm 1, i.e., by computing OptErr$(\mathcal{T}_r, k)$. We prove the statement by strong induction on the height of the tree.

Fix a budget $k'$ and let $p(\mathcal{T}_x, k')$ be the contribution to the error probability from $\mathcal{T}_x$ assuming $k'$ sensors are placed optimally in $\mathcal{T}_x$. The base case is a subtree $\mathcal{T}_x$ of height 0, i.e., a leaf: if $k' \geq 1$ then we can place a sensor directly on the leaf. If $k' = 0$ then we cannot resolve $x$ and $p(\mathcal{T}_x, 0) = 1/n$. Now consider the general case of a rooted tree $\mathcal{T}_x$ of height $h > 0$, and assume we can find $p(\mathcal{T}_i, k'_i)$ for all trees $\mathcal{T}_i$ of height less than $h$. If $k' = 0$, then $p(\mathcal{T}_x, 0) = |T_x|/n$ since we have no way to distinguish between any vertices in $\mathcal{T}_x$. Otherwise, we recurse over all possible partitions of $k'$ between the subtrees rooted at the children of $x$.[7] In particular, if $g$ is the number of children of $x$ and $\mathcal{T}_{x,i}$, for $i \in [g]$, denotes the subtree rooted at the $i$*th* child of $x$, any configuration of $k'$ sensors in $\mathcal{T}_x$ has $0 \leq k'_i \leq k'$ sensors in subtree $\mathcal{T}_{x,i}$ with $\sum_{i=1}^{g} k'_i = k'$. If $k'_i \neq k$ for every $i$ (in particular if $k' < k$), $p(\mathcal{T}_x, k') = \sum_{k'_i=0}(|\mathcal{T}_{x,i}|/n) + \sum_{k'_i \neq 0} p(\mathcal{T}_{x,i}, k'_i)$. In fact, $x$ is equivalent to all vertices in the subtrees $\mathcal{T}_{x,i}$ (if any) for which $k'_i = 0$ and $|[x]| - 1 = \sum_{k'_i=0} |\mathcal{T}_{x,i}|$. Instead, if there exists $j$ in $[g]$ such that $k'_j = k$ (all sensors are placed in the subtree $\mathcal{T}_{x,j}$), $p(\mathcal{T}_x, k) = \sum_{i \neq j}(|\mathcal{T}_{x,i}|/n) + p(\mathcal{T}_{x,j}, k) + 1/n$, because the j$th$ child of $x$ is equivalent to $x$ and to all vertices in the subtrees $\mathcal{T}_{x,i}$ with $i \neq j$. Since the height of each $\mathcal{T}_{x,i}$ is less than $h$, by the induction hypothesis we can compute the optimal $p(\mathcal{T}_{x,i}, k'_i)$, and hence $p(\mathcal{T}_x, k')$. □

**Proof.** [Complexity] A call to OptErrChildren is determined by the root $x$ of the subtree, the subset $c$ of its children considered and the budget $k' \leq k$. The possible values for the pair $x, c$ is the number of edges $n-1$. In fact, we can assume that the children are ordered and the possible partitions are of the form ($c$, children at the right of $c$) so the number of pairs $(x, c)$ is bounded by $n-1$. Hence, there are $O(nk)$ possible calls of OptErrChildren. Combining this with the minimization on $m \leq k$ sensors sent to the leftmost sub-tree, the complexity is $O(nk^2)$. □

**Algorithm 1** *Minimizes $\mathcal{P}_e$ with budget $k$ on a tree of size $n$ rooted at $r$*

OptErr($\mathcal{T}_x$, $k'$)
**if** $k' = 0$ **return** $|\mathcal{T}_x|/n$
**if** $|\mathcal{T}_x| = 1$, $e \leftarrow 0$ **else** $e \leftarrow$ OptErrChildren($\mathcal{T}_x$, $k'$, *children*($x$))

[7] See the function OptErrChildren in Algorithm 1.

**if** $x \neq r$ **and** $k' = k$ **return** $e + 1/n$, **else return** $e$
  OptErrChildren($\mathcal{T}_x$, $k'$, $C$)
**if** $|C| = 0$ **return** 0, **else if** $k' = 0$ **return** $\sum_{c \in C} |subtree(c)|/n$
$f \leftarrow$ *first child*, $oc \leftarrow$ *other children*, $results \leftarrow \{\}$
**for** $m$ *from* 0 *to* $k'$
  $results \leftarrow results \cup \{\text{OptErr}(\mathcal{T}_f, m) + \text{OptErrChildren}(\mathcal{T}_x, k' - m, oc)\}$
**return** $\min\{results\}$

## 4 Expected Distance Minimization

If $\mathcal{G}$ is a graph of size $n$ with weighted distance $d$, $S$ the set of sensors, $|S| = k$, and the prior $\pi$ uniform, the expected distance between the real source $s^*$ and the estimated source $\hat{s}$ is

$$\mathbb{E}[d(s^*, \hat{s})] = \frac{1}{n} \sum_{[u]_S} \Big( \sum_{s,t \in [u]_S} \frac{d(s,t)}{|[u]|_S} \Big).$$

In this case, the contribution of each unresolved vertex depends on the sum of distances between the vertices in an equivalence class *in addition to* the size of the class; this makes the problem more challenging. It can be proven that if $\mathcal{T}$ is a tree of size $n$ with maximum degree $D$, $\ell$ leaves and uniform prior $\pi$, the leaf set minimizes $\mathbb{E}_S[d(s^*, \hat{s})]$ when $k \geq \ell$; if $k \in [\ell - 1]$, there exists an algorithm that finds $S^k_{opt} \in \text{argmin}_{|S|=k} \mathbb{E}_S[d(s^*, \hat{s})]$ in time $O(2^D n k^2)$.

## 5 Future Work

An important open problem is extending our results to general graphs. Other interesting directions include optimizing *worst case* metrics rather than *average case* metrics, accounting for noisy infection delays and transmission failures, and non-uniform prior distributions on the position of $s^*$.

## References


[1] Shah, D., and T. Zaman, *Rumors in a network: who's the culprit?*, IEEE Transactions on information theory **57**(8) (2011), 5163-5181.

[2] Pinto, P., P. Thiran, and M. Vetterli, *Locating the Source of Diffusion in Large-Scale Networks*, Physical Review Letters, **109** (2012), 068702.

[3] Chen, X., X. Hu, and C. Wang, *Approximability of the Minimum Weighted Doubly Resolving Set Problem*, Proc. $20^{th}$ Int. Computing & Combinatorics Conf. (2014), 357-368.

[4] Zejnilovic, S., J.P. Gomes, and B. Sinopoli, *Network observability and localization of the source of diffusion based on a subset of vertices*, Proc. of the $51^{st}$ Allerton Conf. on Communication, Control & Computing (2013), 847-852.

[5] Cáceres, J., M.C. Hernando, M. Mora, I.M. Pelayo, M.L. Puertas, C. Seara, and D.R. Wood, *On the metric dimension of cartesian products of graphs*, SIAM J. Discrete Math. **21**(2) (2007), 423-441.

# A Error Probability vs Expected Distance

Depending on the context in which we study source localization, we could be more interested in maximizing the chances of an exact identification of the source (i.e., in minimizing the error probability) or in minimizing the expected distance between the real and the estimated source. Typically, if we want to identify the culprit of a malicious diffusion we want to minimize the probability of not estimating the identity of the source correctly, while for the diffusion of a contaminant in a physical network we give more importance to having an estimation *close enough* to the actual source, so that containment measures can be put into place.

The sensor choice which minimizes the error probability may be different from the one that minimizes $\mathbb{E}[d(s^*, \hat{s})]$. In particular for trees, given the same budget of sensors, the choice of the leaves in the sensor set depends on the metric that we want to minimize. Figure A.1 shows: (a) an example in which adding a sensor in one of two possible vertices has a different impact on the two metrics and (b,c) a tree on which the optimal placement of $k = 2$ sensors is different for the two metrics.

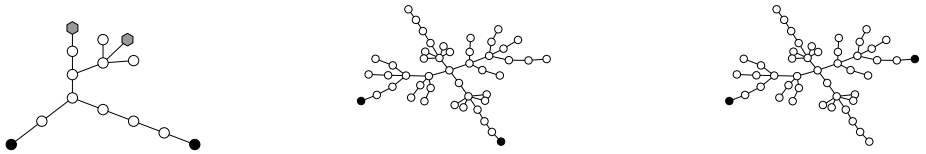
# B Proofs of Preliminary Results

**Lemma B.1 (Lemma 2.6)** *Let $\mathcal{T}$ be a tree with $n$ vertices, $S \subseteq V$.*

(i) *Let $u \in S$ a non-leaf vertex or $u \in V \backslash S$. If there do not exist $s_1, s_2 \in S$, $s_1, s_2 \neq u$, s.t. $u \in \mathcal{P}(s_1, s_2)$, then $u$ is not resolved by $S$;*

(ii) *let $u \in V$ a non-leaf vertex and root $\mathcal{T}$ at $u$. $u$ is resolved if and only if every subtree $\mathcal{T}_c$ rooted at a child $c$ of $u$ contains at least one vertex of $S$;*

(iii) *let $|S| > 1$: a leaf-vertex $\ell$ is resolved if and only if $\ell \in S$.*



(a) (b) (c)

Fig. A.1. (a): Tree with 2 sensor vertices (black). Adding a sensor in one of the two gray vertices has the same effect on $\mathcal{P}_e$ but not on $\mathbb{E}[d(s^*, \hat{s})]$. (b): Placement that minimizes $\mathcal{P}_e$ for $k = 2$ sensors: $\mathcal{P}_e \approx 0.74$, $\mathbb{E}[d(s^*, \hat{s})] \approx 2.46$ (c): Placement that minimizes $\mathbb{E}[d(s^*, \hat{s})]$ for $k = 2$ sensors: $\mathcal{P}_e \approx 0.76$, $\mathbb{E}[d(s^*, \hat{s})] \approx 2.41$.

**Proof.**

(i) Suppose first that $u \in V \backslash S$. If there do not exist $s_1, s_2 \in S$, s.t. $u \in \mathcal{P}(s_1, s_2)$ it means that all vertices in $S$ are in a subtree rooted at a neighbor of $u$, say $c$, and not containing $u$ itself. Hence it is easy to see that $u$ is equivalent to $c$ and is not resolved. Next, suppose that $u \in S$ is a non-leaf vertex. If there do not exist $s_1, s_2 \in S$, $s_1, s_2 \neq u$, s.t. $u \in \mathcal{P}(s_1, s_2)$, it means that there is at least a neighbor of $u$, say $c$, such that no vertex of $S$ is contained in the subtree rooted at $c$ and not containing $u$. Hence $u$ is not resolved because it is equivalent to $c$;

(ii) let $u$ be a resolved non-leaf vertex. By contradiction, let $u$ have a neighbor $c$ such that the subtree rooted at $c$ and not containing $u$ has empty intersection with $S$. Then, as in (i), $c$ is equivalent to $u$, giving a contradiction with the fact that $u$ is resolved. For the backward direction take a non-leaf vertex $u$ and a vertex $v$ in a subtree $\mathcal{T}_c$ rooted at a neighbor $c$ of $u$. Since $u$ is not a leaf, it has at least one other neighbor $h \neq c$. To resolve the pair $(u, v)$ it is then enough to take a sensor in $\mathcal{T}_c$ and one in $\mathcal{T}_h$;

(iii) let $\ell \in S$ be a leaf-vertex. Since $|S| > 1$ there exists $t \in S$, $t \neq \ell$. Remember that $\mathcal{T}$ is a tree: for every $x \in V$, $x \neq \ell, t$ there exists a vertex $y$ such that $y \in \mathcal{P}(\ell, t) \cap \mathcal{P}(x, \ell) \cap \mathcal{P}(x, t)$, with possibly $y = x$. Then, $d(\ell, t) - d(\ell, \ell) = d(\ell, t) > d(y, t) - d(y, \ell) = d(x, t) - d(x, \ell)$, implying that $\ell$ is resolved by $S$. Suppose next that $\ell \notin S$. Then because of (i) $\ell$ is not resolved by $S$.

□

**Proposition B.2 (Prop. 3.1)** *Let $\mathcal{G}$ be a graph of size $n$, $S \subseteq V$, $|S| = k$*

*and let the prior $\pi$ be uniform. The probability of error $\mathcal{P}_e(S)$ is given by* $\mathcal{P}_e(S) = \frac{1}{n}\sum_{[u]_S \subseteq V}(|[u]_S| - 1)$.

**Proof.** We have that $\mathcal{P}_e(S) = \sum_{[u]_S \subseteq V} \mathcal{P}(\hat{s} \neq s^* | s^* \in [u]_S)\mathcal{P}(s^* \in [u]_S)$. Hence $\mathcal{P}_e(S) = \sum_{[u]_S \subseteq V} \frac{|[u]_S|-1}{|[u]_S|} \cdot \frac{|[u]_S|}{n} = \sum_{[u]_S \subseteq V} \frac{|[u]_S|-1}{n}$. □

## C Extension of Theorem 3.2 to Weighted Nodes

Theorem 3.2 can be extended to the more challenging case where vertices have different integer costs and $k \in \mathbb{N}$ represents the total budget allowed. The additional difficulty comes from the fact that, if each vertex $u$ has a cost $c(u) \in \mathbb{N}$, the optimal observer placement $S_{opt}$ will not necessarily be contained in the leaf set, especially if the leaves have high cost compared to other vertices of the network.

We give a few remarks on how the structure of Algorithm 1 can be adapted to meet this case without increasing the total complexity of the algorithm. At each call of the function OptErr on a subtree $\mathcal{T}_x$ with budget $k' > c(x)$ two possible cases need to be considered: 1) vertex $x$ itself is chosen as a sensor and the remaining budget $k' - c(x)$ is distributed in the subtrees rooted at the children of $x$; 2) vertex $x$ is not chosen as a sensor and we distribute the entire budget $k'$ among the children of $x$. At an algorithm level this translates in an additional call to the function OptErrChildren for every vertex $x$. Hence the complexity of the algorithm is not affected.

## D Minimization of the Expected Distance between $s^*$ and $\hat{s}$: Proofs and Details

As in the case of error probability, if $\mathcal{T}$ has $\ell$ leaves, it is easy to see that the sensor set $S$ of cardinality $k < \ell$ that minimizes $\mathbb{E}[d(s^*, \hat{s})]$, is contained in the leaf set.

We give an algorithm that, given a sensor budget $k$, finds the set of sensors $S$ that minimizes the expected error distance. The structure of this algorithm is similar to that of Algorithm 1. However, in the error probability case, the contribution of each equivalence class was a function only of the number of its elements. For the expected distance the contribution of each unresolved vertex to the expected error distance depends on the sum of distances between the vertices of its equivalence class and this makes the minimization problem more challenging.

**Theorem D.1** *Let $\mathcal{T}$ be a tree of maximum degree $D$ with $n$ vertices and $\ell$ leaves and let the prior $\pi$ be uniform. If $k \geq \ell$, the leaf set is an optimal sensor set. If $k \in [\ell - 1]$, there exists an algorithm that finds the set $S^k_{opt} \in \operatorname{argmin}_{|S|=k} \mathbb{E}_S[d(s^*, \hat{s})]$ in time $O(2^D n k^2)$.*

**Proof.** Consider the tree as rooted at an arbitrary non-leaf vertex $r$. We claim that the main function of Algorithm 2, i.e., OptDist$(\mathcal{T}_r, k)$, gives the desired result. The structure of Algorithm 2 is very similar to that of Algorithm 1 as is the proof of the result, so we limit ourselves to highlighting the differences. When computing the expected distance of a tree rooted at $x$ we need to keep track of all the subtrees rooted at the children of $x$ where there is not any sensor (see the variable *unsensored-neighbors* in the pseudo-code). With the notation of the proof of Theorem 3.2, if $k'_i \neq k$ for every subtree $\mathcal{T}_{x,i}$, the expected distance $e(T_x, k')$ of the classes entirely contained in $\mathcal{T}_x$ is computed as

$$e(\mathcal{T}_x, k') = \sum_{k'_i \neq 0} e(\mathcal{T}_{x,i}, k'_i) + \mathbb{E}[d(\hat{s}, s^*) | s^* \in \{x, V(\mathcal{T}_{x,i}) : k'_i = 0\}].$$

Instead, if there exist a child $x_j$ of $x$ such that $k'_j = k$ (all sensors are in $\mathcal{T}_{x,j}$),

$$e(\mathcal{T}_x, k) = e(\mathcal{T}_{x,j}, k) + \mathbb{E}[d(\hat{s}, s^*) | s^* \in [x_j]].$$

Note that in this case the subtree $\mathcal{S}_{x_j}$ rooted at $x_j$ and containing the root vertex $r$ is entirely contained in the class $[x_j]$. The case $k' = 0$ never arises in the calls to OptDist: when no sensor is assigned to a given subtree, it is enough to add its root to the list of *unsensored-neighbors* in the next calls of OptDistChildren so that the entire subtree will contribute to the final computation of the expected distance. With respect to Theorem 3.2 the call to OptDistChildren has an additional argument which corresponds to the list of neighboring subtrees that have already been considered and to which no sensor has been assigned. Since the number of neighbors of $x$ is less than the maximum degree $D$, the number of possible calls to the algorithm for a given $x$ and a sensor budget $k$ is upper-bounded by $2^D$ and the total number of calls is $O(2^D n k^2)$. Finally, the expected distance for every $x$ and all subsets of neighboring subtrees can be precomputed with a complexity of $O(2^D n)$ so that the total running time for the algorithm is $O(2^D n k^2)$. □

In the pseudo-code below, when $|\mathcal{T}_x| = 1$ and $k' > 1$ we return $\infty$: in this way the cases in which the budget is not completely allocated are directly excluded.

**Algorithm 2** *Minimizes the expected distance for initial budget $k$ on a tree of size $n$*

OptDist($\mathcal{T}_x$, $k'$)
**if** $|\mathcal{T}_x| = 1$
  **if** $k' = 1$ **return** 0
  **else return** $\infty$
**if** $x \neq r$ **and** $k' = k$
  *non-sensored-neighbors* $\leftarrow$ [*parent(x)*]
**else** *non-sensored-neighbors* $\leftarrow$ [ ]
**return** OptDistChildren($\mathcal{T}_x$, $k$, *children*$(x)$, *non-sensored-neighbors*)

OptDistChildren($\mathcal{T}_x$, $k'$, $C$, $N$)
**if** $|C|$=0 ***and*** $k' > 0$ **return** $\infty$
**if** $k = 0$ **return** ExpDist$(x, N)$
**if** $x \neq r$ **and** $k' = k$
  **for** $c$ *in* $C$
    *results* $\leftarrow \{$OptDist($\mathcal{T}_c$, $k$)$\}$
$f \leftarrow$ *first child*, $oc \leftarrow$ *other children*
*results* $\leftarrow \{$OptDistChildren($\mathcal{T}_x$, $k'$, $oc$, $N \cup \{f\}$)$\}$
$h \leftarrow \min(k', k-1)$
**for** $m$ *from* 1 *to* $h$
  $err_1 \leftarrow$ OptDist($\mathcal{T}_f$, $m$)
  $err_2 \leftarrow$ OptDistChildren($\mathcal{T}_x$, $k' - m$, $oc$, $N$)
  $results \leftarrow err_1 + err_2 \cup results$
**return** $\min\{results\}$