Budgeted sensor placement for source localization on trees

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

[a]School of Computer and Communication Sciences, EPFL, Lausanne, Switzerland
[b]Faculty of Electrical Engineering and Computing, University of Zagreb, Croatia

Abstract

This paper considers the problem of source localization on graphs. We assume that a diffusion process on a graph is partially observed, i.e., that only the time at which some of the vertices are reached by the diffusion is known, and we want to identify the vertex that initiated the diffusion. In particular, we address the problem of choosing a fixed number of *sensor vertices* based on whose diffusion time one can optimally estimate the source.

Building on the definition of *Double Resolving Set* of a graph, we introduce a notion of *vertex resolvability* that turns out to be the key to study the optimality of a sensor set. For the case of trees we give polynomial time algorithms for both allocating a fixed budget k of sensors such that they maximize the probability of correct detection of the source, and for identifying the k sensors that minimize the expected distance between the real source and the estimated one.

Keywords: Source Localization, Sensor Placement, Double Resolvability

1. Introduction

We consider the general setting in which a diffusion (e.g., an infection process, a rumor propagation) spreads on a known graph topology. Source

Email addresses: elisa.celis@epfl.ch (L. E. Celis), fpavetic@google.com (F. Pavetić), brunella.spinelli@epfl.ch (B. Spinelli), patrick.thiran@epfl.ch (P. Thiran)

[1] 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.

localization is the problem of finding the vertices that initiate the diffusion based on a partial knowledge of the diffusion process itself. The applications of source localization cover many different problems, from estimating the first individual or community infected by a virulent disease to the identification of malicious users in social networks. In this paper we study the problem of optimally choosing a limited number of vertices, called *sensors*, whose infection times we want to use to locate the source of any possible diffusion spreading on the graph.

1.1. Model

We consider a graph $\mathcal{G}(V, E)$ with weighted edges 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^*. 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*. 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 $w_{u,v} \in \mathbb{R}_+$ is the weight of edge (u, v).

We assume that 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 $\{t_s, s \in S\}$. Knowing these infection times, we can identify a set of *candidate source vertices*, i.e., of source locations that are compatible with the known infection times.

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 precision in identifying the candidate source vertices.

1.2. Error Metrics

Depending on the context in which we want to locate 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:

1. the *error probability*, i.e., $\mathcal{P}_e = \mathcal{P}(\hat{s} \neq s^*)$;

2. 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 (see the beginning of Section 3).

1.3. Our contribution

It is easy to show that, if the sensors set S is a Double Resolving Set (DRS) for the graph, a perfect detection of the source is guaranteed [8]. However, finding the minimum-size DRS of a graph is a NP-hard problem [8]. Via a reduction to the DRS problem, one can show that the minimization of the error probability or of the expected error distance *given a finite budget* on the number of sensors is also NP-hard.[2]

In this paper we focus on trees. Building on the definition of DRS, in Section 2 we introduce a concept of *vertex resolvability.* First, we 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. Section 3 contains our main results. For the case of a tree $\mathcal{T}$ of size n with a uniform prior on the position of the source, we design an $O(nk^2)$ dynamic-programming algorithm to find k sensor vertices that minimize the number of unresolved 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 with bounded vertex degree, an optimal set S can be found with a polynomial-time algorithm. In Section 4 we generalize our results to (i) the case in which sensors have a non-unit cost and the budget limits the maximum cost of the sensor set and (ii) the case of general priors on the position of the source.

1.4. Related work

Source localization has received considerable attention in the last years, as documented in a recent survey by Jiang et al. [1].

Many approaches, starting with the seminal work by Shah and Zaman [2], rely on knowing the state of the entire graph at a given instant t in time. Generalizations to settings in which only the state of a fraction of vertices at a given time t is known have also been proposed (see e.g., [3, 4]). Pinto et al. [5] introduced a fundamentally different model that instead, estimates the source based on the infection times of a sparse set of *sensor vertices* that are *a priori* chosen in the graph. Such a setting is very interesting from an application point of view because it models all the problems in which

[2] If this was not the case one could find the minimum k such that one of the two metrics is minimized and solve DRS in polynomial time. In fact, for both metrics, it holds that if they are equal to 0 for a sensor set S, then S is a DRS.

knowing the state of the vertices might have a nontrivial cost and one may want to allocate a limited budget, i.e., choose *once and for all* some vertices to *observe*. Another model for source estimation uses only the information about a set of sensor vertices being or not being infected [6]. However, in both these works, the sensors are chosen according to heuristic centrality measures such as Degree-, Distance- or Betweenness-Centrality. Zhang et al. [7] recently proposed a new heuristic called *coverage rate*, linked to the total number of vertices neighboring observers. Importantly, none of this heuristics is *directly* linked to the source localization problem. In this work instead we consider the problem of selecting *which* vertices to observe in order to optimize the performance of source localization when a *budget* on the number (or cost) of allowed sensors is given. An approximation algorithm for minimising the cardinality of the sensor set that *perfectly* detects the source was given by Chen et al. [8] using the connection to the Doubly Resolving Set (DRS) of a graph [11]. The latter result was recently extended to the case of multiple sources by Zhang et al. [9]. When the *starting time* of the diffusion is known, a question similar to that of Chen at al. [8] was considered by [10]. In all these works budget constraints are not considered.

2. Preliminaries

Throughout this work, we assume that 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 denote with S the set of vertices for which the infection time is known, i.e., the sensors. We choose one sensor, say $s_1 \in S$ as *reference point* and define a vector of relative observed times as follows.

Definition 1 (observation vector). *Let $\mathcal{G}$ be a graph, $S \subseteq V$, $|S| = k \geq 2$ 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.*

Given $S \subseteq V$, each vertex is associated to a distance vector.

Definition 2 (distance vector). *Let $\mathcal{G}$ be a graph, $S \subseteq V$, $|S| = k \geq 2$ 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.*

Building on the previous definition, we can introduce a notion of *resolved / unresolved* vertices.

Definition 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 $u \sim v$ if and only if $\mathbf{d}_u = \mathbf{d}_v$ 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 $u \in V$, i.e., such that each vertex is resolved, is a Double Resolving Set, as clarified by the definition and lemma below.

Definition 4 (Double Resolvability)**.** *Given a graph $\mathcal{G}$, $S \subseteq V$, $|S| \geq 2$ 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 1 (Lemma 3.1 in [8])**.** *Let $S \subseteq V$, $|S| \geq 2$ and fix $s \in S$. Then for every vertices $x, y \in V$ doubly resolved by S there exists $w \in S \backslash \{s\}$ such that $d(x,s) - d(x,w) \neq d(y,s) - d(y,w)$.*

Proof. Since x, y are doubly resolved by S, there exist $u, v \in S$ s.t. $d(x,u) - d(x,v) \neq d(y,u) - d(y,v)$. This is equivalent to $d(x,u) - d(y,u) \neq d(x,v) - d(y,v)$. Either $d(x,s) - d(y,s) \neq d(x,u) - d(y,u)$ or $d(x,s) - d(y,s) \neq d(x,v) - d(y,v)$. Then by taking $w = u$ or $w = v$, x, y are doubly resolved by (s, w). □

As a consequence of Lemma 1, the definition of resolved and unresolved vertices (Definition 4) does not depend on the choice of *reference point* $s_1 \in S$ [8].

In the following lemma we make some key observations necessary for sensor placement on trees (see Figure 1 for an illustration).

Lemma 2. *Let $\mathcal{T}$ be a tree with n vertices, $S \subseteq V$, $|S| \geq 2$. We denote by $\mathcal{P}(u,v)$ the unique shortest path between u and v in $\mathcal{T}$.*

(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) a leaf-vertex ℓ is resolved if and only if $\ell \in S$.

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 $y = x$ or $y = t$. Then, $d(\ell, t) - d(\ell, \ell) = d(\ell, t) > d(y, t) - d(y, \ell) = d(x, t) - d(x, \ell)$, implying that ℓ is resolved by S. Suppose next that $\ell \notin S$. Then ℓ is equivalent to its only neighbor and is not resolved by S.

□

Recall that if a vertex v gets infected at time t_v, it infects each of its non-infected neighbor u at time $t_u = t_v + w_{u,v}$ where $w_{u,v} \in \mathbb{R}_+$ is the weight of edge (u, v). Then it is clear that based on the observation vector $\boldsymbol{\tau}$ one can identify a set of vertices that are candidate sources. 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 π on V for the position of s^*, we select an approximated source $\hat{s}$ by sampling the conditional distribution $\pi|_{[s^*]}$.

Remark 1. *On trees, this model and estimator tolerate a uniformly bounded amount of noise in the transmission delays: in fact, the estimation of the*

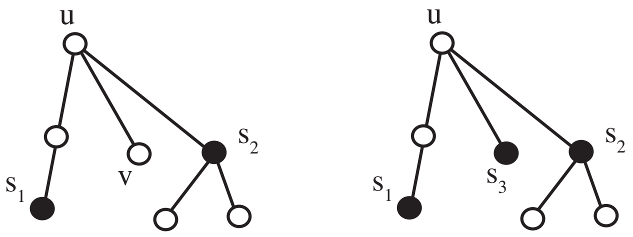

Figure 1: Illustration of Lemma 2. Black vertices represent sensors. On the left, illustration for statement *(i)*: v does not lie on the path between the two sensors and is not resolved; s_2 is also not resolved: in fact it is equivalent to its two leaf neighbors. On the right, illustration for statements *(ii)* and *(iii)*: the tree rooted at u contains a sensor in each subtree, hence u is resolved; leaves s_1 and s_3 are resolved, the other leaves are not.

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. Main Results

Our main results are polynomial time algorithms to find the optimal sensor placements that minimize the error probability and the expected error distance in source localization. We start with a comparison of the two metrics. We emphasize that: (i) each of the two metrics may be preferred over the other depending on the application considered; (ii) the sensor sets that optimize the two metrics for a given budget of sensors are *a priori* not equal.

(i) 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 maximize the probability of 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.

(ii) 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 2 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.

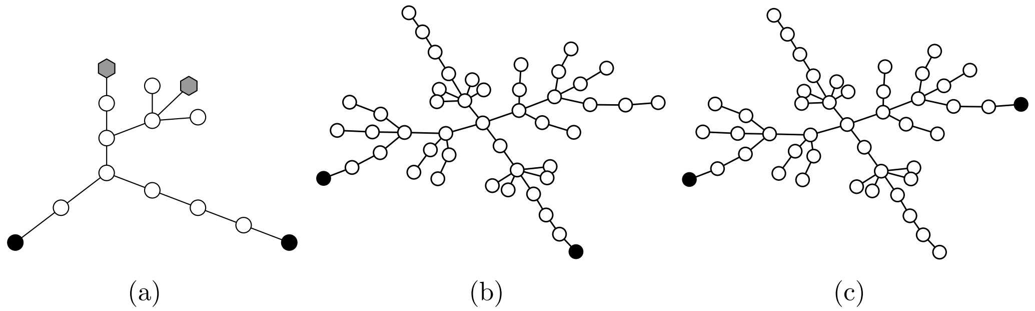

Figure 2: (a): Tree with 2 sensor vertices (black). Adding a sensor in one of the two gray vertices has the same effect on $\mathcal{P}_e$ but not on $\mathbb{E}[d(s^*, \hat{s})]$. (b): Placement that minimizes $\mathcal{P}_e$ for $k = 2$ sensors: $\mathcal{P}_e \approx 0.74$, $\mathbb{E}[d(s^*, \hat{s})]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

3.1. Error Probability Minimization

We first consider the minimization of the error probability. We start by deriving an expression of this metric on a general graph. Subsequently we explain how it can minimized in $O(nk^2)$ on trees of n vertices when a budget of k sensors is available.

Proposition 1. *Let $\mathcal{G}$ be a graph of size n, $S \subseteq V$, $|S| \geq 2$, and uniform prior π. 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 $\mathcal{P}_e(S) = \sum_{[u]_S \subseteq V} \frac{|[u]_S|-1}{|[u]_S|} \cdot \frac{|[u]_S|}{n} = \sum_{[u]_S \subseteq V} \frac{|[u]_S|-1}{n}$. □

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. Hence it is clear that an error in source estimation can occur only if the source s^* is not resolved by S.

Looking back to Lemma 2, if the graph is a tree $\mathcal{T}$, $\mathcal{P}_e$ is 0 if a sensor is placed on each leaf.[3] In fact, the minimum k required for $\mathcal{P}_e = 0$ is the number of leaves ℓ. Moreover, as a consequence of Lemma 2 it is also clear that, 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 runtime $O(nk^2)$.

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

Correctness. The statement is trivial for $k \geq \ell$ as the set of leaves resolves all the vertices. If $2 \leq k < \ell$, 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 $\textsc{OptErr}(\mathcal{T}_r, k)$. 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 (see the function $\textsc{OptErrChildren}$ in Algorithm 1). 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 $\mathcal{T}_{x,j}$), $p(\mathcal{T}_x, k) = \sum_{i \neq j}(|\mathcal{T}_{x,i}|/n) + p(\mathcal{T}_{x,j}, k) + 1/n$,

[3] See also [8] for a different proof.

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 $p(\mathcal{T}_x, k')$. □

Runtime Bound. A call to OptErrChildren is determined by the root x of the subtree, the subset c of its children considered and the budget $k' \leq k$. There are $n-1$ possible values for the pair (x, c) ($n-1$ is the number of edges $\mathcal{T}$). 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)[4] 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 runtime 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($x$))
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 $k'$
  $results \leftarrow results \cup \{$OptErr($\mathcal{T}_f, m$)+ OptErrChildren($\mathcal{T}_x, k' - m, oc$)$\}$
return $\min\{results\}$
```

3.2. Expected Distance Minimization

We now turn to the minimization of the expected error distance. After deriving an expression for this metric on a general graph, we explain why the minimization of this second metric is more challenging. We propose an algorithm that finds the optimal placement of k sensors on a tree of n vertices in time $O(2^D nk^2)$, D being the maximum vertex degree.

[4] We consider the children of any vertex to be ordered, e.g. according to the order induced by any embedding of $\mathcal{T}$ in the plane.

Proposition 2. *If $\mathcal{G}$ is a graph of size n with weighted distance d, S the set of sensors, $|S| = k \geq 2$, and the prior π is uniform, the expected distance between the real source s^* and the estimated source $\hat{s}$ is*

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

Proof. We have that

$$\begin{aligned} \mathbb{E}[d(s^*, \hat{s})] &= \sum_{[u]_S \subseteq V} \mathbb{E}[d(s^*, \hat{s}) | s^* \in [u]_S] \mathcal{P}(s^* \in [u]_S) \\ &= \sum_{[u]_S \subseteq V} \frac{|[u]_S|}{n} \Big(\sum_{s,t \in [u]_S} \frac{d(s,t)}{|[u]|_S^2} \Big) \\ &= \frac{1}{n} \sum_{[u]_S \subseteq V} \Big(\sum_{s,t \in [u]_S} \frac{d(s,t)}{|[u]|_S} \Big). \end{aligned} \tag{2}$$

□

In the error probability case, the contribution of each equivalence class was a function only of the number of its elements (see Prop. 1). Here instead, the contribution of each unresolved vertex to (1) 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.

Clearly, if $\mathcal{T}$ is with ℓ leaves, the leaf set minimizes $\mathbb{E}_S[d(s^*, \hat{s})]$ when $k \geq \ell$. Again, if $k \in \{2, .., \ell\}$, we observe that S_{opt}^k is contained in the leaves set: in fact, if it was not the case, there would be a sensor s equivalent to a leaf $\ell \notin S$ and by substituting the sensor in s with a sensor in ℓ we would break $[s]$ in two or more smaller equivalence classes and the expected error distance would decrease.[5]

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

[5] When adding a sensor in ℓ, the expected error distance can only decrease and the original sensor s becomes redundant and can be removed.

Correctness. Consider $\mathcal{T}$ as rooted at an arbitrary non-leaf vertex r. We claim that the main function of Algorithm 2, i.e., OptDist($\mathcal{T}_r, k$), finds the set S^k_{opt}. The structure of Algorithm 2 is very similar to that of Algorithm 1 as is the proof of correctness for the algorithm, 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 1, 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 $\mathcal{T}_{x,j}$), $e(\mathcal{T}_{x,j}, k)$ is computed taking into account that 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 since, 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.

Finally we look at the pre-computation of the contributions to the expected distance. For every subtree $\mathcal{T}_x$ and subset $N = \{x_1, .., x_m\}$ of children of x for which the subtree $\mathcal{T}_{x_i}$ does not contain sensors, the expected distance (denoted by ExpDist(x, N) in the pseudo-code) is recursively pre-computed as follows. The base case is a subtree of only one element, i.e. a leaf, for which the expected distance is 0. For a non-leaf vertex x, the contribution to the expected distance of the subtree rooted at x can be computed based on the contributions of the subtrees rooted at the children of x. We note that if $i, j \in \{1, .., m\}$, $i \neq j$,

$$\begin{aligned}\sum_{u \in \mathcal{T}_{x_i}, v \in \mathcal{T}_{x_j}} d(u, v) &= \sum_{u \in \mathcal{T}_{x_i}, v \in \mathcal{T}_{x_j}} d(u, x) + d(x, v) \\ &= |\mathcal{T}_{x_j}| \sum_{u \in \mathcal{T}_{x_i}} d(u, x) + |\mathcal{T}_{x_i}| \sum_{v \in \mathcal{T}_{x_j}} d(v, x).\end{aligned}$$

Then, if there is at least a sensor in $\mathcal{T}\backslash\mathcal{T}_x$, i.e., if $[x] \subseteq \mathcal{T}_x$, we have

$$\begin{aligned}\mathbb{E}[d(\hat{s},s^*)|s^* \in [x]] =& \sum_{j=1}^{m} \frac{|\mathcal{T}_{x_j}|}{|\mathcal{T}_x|}\mathbb{E}[d(\hat{s},s^*)|s^* \in \mathcal{T}_{x_j}] \\ &+ \frac{2}{|\mathcal{T}_x|}\sum_{j=1}^{m}\Big(|\mathcal{T}_x\backslash\mathcal{T}_{x_j}| \sum_{u\in\mathcal{T}_{x_j}} d(u,x)\Big) \\ &+ \frac{2}{|\mathcal{T}_x|}\sum_{j=1}^{m}\Big(\sum_{u\in\mathcal{T}_{x_j}} d(u,x)\Big),\end{aligned}$$

where the first term accounts for the cases in which s^* and $\hat{s}$ are in the same subtree, the second term for the cases in which s^* and $\hat{s}$ are in two different subtrees and the last term accounts for the cases in which either s^* or $\hat{s}$ are in x itself. The sums of distances from x to the vertices in a given subtree that appear in the latter expression can again be recursively pre-computed. If all the available sensors are in $\mathcal{T}_x$, i.e., if $[x] \nsubseteq \mathcal{T}_x$, also the contribution to the expected distance coming from the subtree rooted at the father of x and not containing x should be added. Once more, this contribution can be recursively pre-computed with similar techniques. □

Runtime bound. With respect to Theorem 1 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 $O(2^D nk^2)$. Finally, the expected distance for every x and all subsets of neighboring subtrees can be pre-computed with a runtime $O(2^D n)$. In conclusion the total running time for the algorithm is $O(2^D nk^2)$. □

In the pseudo-code below, when $|\mathcal{T}_x| = 1$ and $k' > 1$ we return ∞: 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 $k = 0$
    $N \leftarrow N \cup C$
    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 $\min\{results\}$
```

4. Extensions

This section presents the extension of our results to weighted vertices and to general priors on the position of the source.

4.1. Extension to weighted vertices

Theorem 1 and Theorem 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 graph.

We give a few remarks on how the structure of Algorithm 1 (respectively, Algorithm 2) can be adapted to this case without increasing the total runtime bound of the algorithm. At each call of the function OptErr (respectively, OptDist) 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 (respectively, OptDistChildren) for every vertex x. Hence the runtime bound of the algorithms is not affected.

4.2. Extension to non-uniform prior π

We now consider the case of a non-uniform prior probability π on the position of the source s^*.

Proposition 3. *If $\mathcal{G}$ is a graph of size n with weighted distance d, S the set of sensors, $|S| = k \geq 2$, and π a general prior on the position of the source, the error probability and the expected distance between the real source s^* and the estimated source $\hat{s}$ are given by the following equations:*

1. *Error probability*

$$\begin{aligned}\mathcal{P}(s^* \neq \hat{s}) &= \sum_{[u]_S \subseteq V} \Big(\sum_{s,t \in [u]_S, s \neq t} \frac{\pi(s)\pi(t)}{\pi([u]_S)} \Big) \\ &= \sum_{[u]_S \subseteq V} \Big(\sum_{s \in [u]_S} \frac{\pi(s)(\pi([u]_S) - \pi(s))}{\pi([u]_S)} \Big).\end{aligned} \tag{3}$$

2. *Expected distance between the real source s^* and the estimated source $\hat{s}$*

$$\mathbb{E}[d(s^*, \hat{s})] = \sum_{[u]_S \subseteq V} \Big(\sum_{s,t \in [u]_S} \frac{d(s,t)\pi(s)\pi(t)}{\pi([u]_S)} \Big). \tag{4}$$

In both Equation (3) and Equation (4) the contribution to each equivalence class depends on the prior probability of each element in the class and, in the case of Equation (4), on the distances between elements in a same class. If we now think of the recursive computation of the two metrics in a tree, in both cases, the contribution of a vertex u depends on which of the subtrees rooted at the children of u contain or do not contain sensors and not only on their cardinality. Hence, in view of Theorem 2 and Algorithm 2, it is clear that for both metrics, on a tree $\mathcal{T}$ with n vertices, it is possible to find an optimal set S^k_{opt} in time $O(2^D n k^2)$. The proof of the result, follows that of Theorem 2.

5. Conclusions and Future Work

In this paper we presented optimal algorithms to place a limited budget of sensors for the problem of source localization. Our algorithms build on the notion of vertex resolvability, which we showed to be the key to study optimal sensor placement for the source localization problem. We considered two metrics of practical relevance, error probability and expected error distance, and gave a polynomial time solution for optimizing them on trees. We showed that our approach is quite general, comprising interesting extensions such as to vertices having different costs and to general priors on the position of the source.

There are many possible directions for 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; e.g., minimizing the maximum distance between a real and estimated source could be of interest. In many practical cases there may be uncertainty about the infection delays or the infection delays may be subject to randomness. Moreover one may want to incorporate in the model the fact that some vertices may fail to infect some of their neighbors. Hence a key extension would study the optimal sensor placement when infection delays are *noisy* and there may be transmission failures.

[1] Jiang, J. S. Wen, S. Yu, Y. Xiang, and W. Zhou, *Identifying Propagation Sources in Networks: State-of-the-Art and Comparative Studies*, IEEE Communication Survey Tutorials (2014)

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

[3] Luo, W., W. Tay, and M. Leng, *How to identify an infection source with limited observations*, IEEE Journal of Selected Topics in Signal Processing **8**(4) (2014), 586-597

[4] Zhu, K., and L. Ying, *A robust information source estimator with sparse observations*, Computational Social Network (2014)

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

[6] Seo, E., P. Mohapatra, and T. Abdelzaher, *Identifying rumors and their sources in social networks*, SPIE Defense, Security, and Sensing (2012)

[7] Zhang, X. Y. Zhang, T. Lv, and Y. Yin, *Identification of efficient observers for locating spreading source in complex networks*, Physica A: Statistical Mechanics and its Applications **442** (2016), 100-109

[8] 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

[9] Z. Zhang, W. Xu, W. Wu, and D. Du, *A novel approach for detecting multiple rumor sources in networks with partial observations*, Journal of Combinatorial Optimization (2015)

[10] 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.

[11] 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.