Graph Tikhonov Regularization and Interpolation via Random Spanning Forests

Yusuf Yigit Pilavci, Pierre-Olivier Amblard, Simon Barthelmé, Nicolas Tremblay
CNRS, Univ. Grenoble Alpes, Grenoble INP, GIPSA-lab, Grenoble, France

***Abstract*—Novel Monte Carlo estimators are proposed to solve both the Tikhonov regularization (TR) and the interpolation problems on graphs. These estimators are based on random spanning forests (RSF), the theoretical properties of which enable to analyze the estimators' theoretical mean and variance. We also show how to perform hyperparameter tuning for these RSF-based estimators. Finally, TR or interpolation being a building block of several algorithms, we show how the proposed estimators can be easily adapted to avoid expensive intermediate steps in well-known algorithms such as generalized semi-supervised learning, label propagation, Newton's method and iteratively reweighted least square. In the experiments, we illustrate the proposed methods on several problems and provide observations on their run time, which are comparable with the state-of-the-art.**

***Index Terms*—graph signal processing, random spanning forests, smoothing, interpolation, semi-supervised learning, label propagation, newton's method, IRLS**

I. Introduction

GRAPHS are ubiquitous models of complex structures, *e.g.* social, transportation, sensors or neuronal networks. The vertices and edges, the main components of graphs, are natural representations of the elements of a network and the links between them, respectively. In many applications, these networks often come with data on the elements. For example, in a transportation network (the roads and their intersections are respectively the nodes and edges), the data can be traffic flow observations of each road [1]; or in a brain network, it could be the activity of each individual brain region [2]. Such type of data over vertices are called graph signals [3], [4].

Graph Tikhonov regularization. Consider a graph of size n represented by its Laplacian matrix $\mathsf{L} = \mathsf{D} - \mathsf{W} \in \mathbb{R}^{n\times n}$ (W is the weighted adjacency matrix and D the diagonal matrix of degrees –see formal definitions in Section II). Given noisy signal measurements $\mathbf{y} = (y_1, y_2, \dots, y_n)^\top$ on the n vertices of this graph, a classical denoising scheme is graph Tikhonov regularization [3], [5]:

$$\hat{\mathbf{x}} = \underset{\mathbf{z}\in\mathbb{R}^n}{\operatorname{argmin}}\, \mu||\mathbf{y}-\mathbf{z}||^2 + \mathbf{z}^\top \mathsf{L}\mathbf{z} \tag{1}$$

where $\mu > 0$ is a hyper-parameter tuning the relative importance the user wants to set between the data-fidelity term $||\mathbf{y}-\mathbf{z}||^2$ and the regularization term $\mathbf{z}^\top\mathsf{L}\mathbf{z}$. Note that $\mathbf{z}^\top\mathsf{L}\mathbf{z} = \sum_{i,j}\mathsf{W}_{ij}(z_i - z_j)^2$, which explains its regularizing property: this term penalizes large variations along the edges of the graph. The exact solution to (1) is:

$$\hat{\mathbf{x}} = \mathsf{K}\mathbf{y} \text{ with } \mathsf{K} = (\mathsf{L}+\mu\mathsf{I})^{-1}\mu\mathsf{I} \in \mathbb{R}^{n\times n}$$

which requires the inversion of the regularized Laplacian.

Graph interpolation. Consider a graph signal $\mathbf{x} \in \mathbb{R}^n$ that is only known over a vertex subset $\ell \subset \mathcal{V}$ (with typically $|\ell| \ll n$). The graph interpolation problem consists in combining this prior information with an underlying smoothness assumption in order to infer the signal on the remaining vertices. One way to formulate this problem is given by Pesenson [6]:

$$\begin{aligned}\hat{\mathbf{x}} &= \arg\min_{z\in\mathbb{R}^n} \mathbf{z}^\top(\mathsf{L}+\mu\mathsf{I})\mathbf{z} \\ &\text{subject to } \forall i \in \ell,\ z_i = x_i\end{aligned} \tag{2}$$

where $\mu \geq 0$ is a user-defined parameter. Note that Pesenson chooses to further parametrize the interpolation problem by considering $(\mathsf{L}+\mu\mathsf{I})^t$ for $t > 0$ instead of $(\mathsf{L}+\mu\mathsf{I})$ in Eq. (2). In this paper, we only consider the case $t = 1$. The exact solution of (2) can be found in [6] and requires the inversion of $\mathsf{L}+\mu\mathsf{I}$ restricted to the rows and columns indexed by the vertices that are not in ℓ (more details are provided in Section III).

Graph TR and interpolation as a building block. Both graph TR and interpolation problems often appear in different graph problems as building blocks.

Node classification is one problem which is sometimes solved by using graph interpolation. A well-known Semi-Supervised Learning (SSL) algorithm, called label propagation[1], corresponds to the Dirichlet boundary problem on graphs [7] whose solution can be viewed as finding interior values that *smoothly* interpolate between the known boundary values [9]. This problem is a sub-case of Eq. (2), for $\mu = 0$.

Graph TR, on top of its use for graph signal denoising, also appears in SSL algorithms for node classification, such as in the work of Zhou *et al.* [10], later generalized by Avrachenkov *et al.* [11]. Moreover, graph TR is also used in graph optimization algorithms. Two examples are Newton's method [12] and iteratively reweighted least squares (IRLS) [13]. In both methods, the computationally expensive steps can be formulated and solved as graph TR problems.

Classical approaches. Explicitly computing the exact solution for graph TR requires the inversion of the regularized Laplacian $\mathsf{L}+\mu\mathsf{I}$ of size $n \times n$, and thus, $\mathcal{O}(n^3)$ elementary operations. For graph interpolation, the overall cost is also $\mathcal{O}(n^3)$ in the typical setting where $|\ell| \ll n$. For large graphs (*i.e.* with $n \geq 10^4$), this is prohibitive and the state-of-the-art relies on approximate methods. These approaches may be roughly separated in two groups, iterative methods (*e.g.* conjugate gradient method with preconditioning [14])

[1]In fact, label propagation may refer to a more generic set of algorithms. In this paper, we refer to the algorithm proposed by Zhu *et al.* [8]

and polynomial approximations (*e.g.* Chebyshev polynomials [15]). Both class of methods run in linear time with the number of edges.

Random processes on graphs. A longstanding and fruitful approach to studying the properties of graphs has been to study the properties of random processes on graphs (via random walks, for instance). This paper will take such a perspective to propose novel estimators for the two problems presented.

For instance, a very well-known fact is the link between the smallest non-null eigenvalue of the Laplacian matrix and the mixing time of a random walk on the graph (see, *e. g.,* [16]). Other examples include properties of electrical networks, such as potential functions or effective resistances, that can be interpreted in terms of probabilistic quantities defined for random walks such as hitting time probabilities [17]. Closer to our work, Wu *et al.* [18], [19] show that the (i, j)-th entry of K equals to the probability of having an interrupted random walk (partially absorbed random walk) starting at node i and ending in node j; and further leverages random walks to give practical insights for the algorithms of several applications including image retrieval, local clustering and semi-supervised learning.

Random spanning forests. In this paper, we will focus on random spanning forests (RSFs): random collections of disjoint trees that cover all nodes in the graph (a formal definition is in section II). The link between the matrix K and random spanning forests has been observed by several authors in the past, such as Avrachenkov *et al.* [20]. Avena *et al.* [21], [22] analyze precisely several aspects of this connection. RSFs not only have a rich theoretical background (connections with Markov chains, determinantal point processes, spectral graph theory), they also come with an efficient sampling algorithm [21]: a variant of Wilson's algorithm [23] based on loop-erased random walks.

Our contributions. In this work,

- We propose two novel Monte Carlo estimators based on RSFs to approximate the solution of graph Tikhonov regularization and interpolation.
  - We provide a rigorous analysis on their performances by building upon known results on RSFs.
  - In terms of computational cost, they are comparable with state-of-the-art methods.
  - By coupling these estimators with certain statistics of RSFs, we provide a scheme to correctly tune the hyperparameters of the problems.
- We show how versatile these estimators are by adapting them to several graph-based problems such as generalized semi-supervised learning, label propagation, Newton's method and Iteratively Reweighted Least Squares (IRLS).

A preliminary version of some of these results can be found in [24].

Organization of the paper. We start with the necessary background on graphs and RSFs in Section II. Then, we introduce the proposed methods in Section III. In Section IV, we examine several extensions to different graph-related problems. Finally, in Section V, we illustrate the methods in different use cases and we conclude in Section VI.

II. Background on RSFs

This section contains background on graph theory, random spanning trees and forests.

A. Graph theory

A directed weighted graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, w)$ consists of a set of vertices $\mathcal{V} = \{1, 2, ..., n\}$ and edges $\mathcal{E} = \{(i, j) \in \mathcal{V} \times \mathcal{V}\}$. The weight function $w : \mathcal{V} \times \mathcal{V} \to \mathbb{R}^+$ maps each edge in $\mathcal{E}$ to a positive weight and others to 0. A graph is called undirected if $w(i, j) = w(j, i)$ for all distinct vertices i and j. In the following, unless otherwise specified only undirected graphs are considered. Graphs are often represented using matrices, and several matrix descriptions are available.

The weighted adjacency matrix or weight matrix is $\mathsf{W} = [w(i, j)]_{i,j} \in \mathbb{R}^{n\times n}$. The degree matrix is the diagonal matrix $\mathsf{D} \in \mathbb{R}^{n\times n}$ with $\mathsf{D}_{i,i} = \sum_{j\in\mathcal{N}(i)} w(i, j)$ and $\mathcal{N}(i)$ is the set of nodes connected to i. The graph Laplacian matrix is defined as $\mathsf{L} = \mathsf{D}-\mathsf{W}$. It is semi-positive definite [16] and its eigenvalues and eigenvectors are usually denoted by $0 = \lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n$ and $\mathsf{U} = (\mathbf{u}_1|\mathbf{u}_2|\ldots|\mathbf{u}_n)$, respectively. The multiplicity of eigenvalue 0 is equal to the number of connected components in the graph [16]. For undirected graphs, all of these square matrices are symmetric.

Another way to represent graphs is via the edge incidence matrix $\mathsf{B} = (\mathbf{b}_1|\mathbf{b}_2|\ldots|\mathbf{b}_m)^\top \in \mathbb{R}^{m\times n}$ where $\mathbf{b}_k \in \mathbb{R}^n$ is a vector associated to the k-th edge (i, j). The only nonzero entries of $\mathbf{b}_k$ are $\mathbf{b}_k(i) = \pm\sqrt{w(i, j)}$ and $\mathbf{b}_k(j) = \mp\sqrt{w(i, j)}$. In directed graphs, the sign is set by considering the edge orientations. For example, if the k-th edge starts from i and ends in j, then, $\mathbf{b}_k(i) < 0$ and $\mathbf{b}_k(j) > 0$. In undirected graphs, the signs of the non-zero entries of $\mathbf{b}_k$ can be arbitrarily chosen as long as they are opposite. Although this matrix seems less natural than the others, it often appears in graph theory. One example is the well known identity $\mathsf{L} = \mathsf{B}^\top\mathsf{B}$.

B. Random spanning trees and Wilson's algorithm

Let us recall the definition of random spanning trees (RSTs). Consider a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, w)$. A subgraph of $\mathcal{G}$ is a graph whose vertex and edge sets are subsets of $\mathcal{V}$ and $\mathcal{E}$, respectively, and its edge weights are valued by w. A subgraph contains a cycle whenever there exists a pair of vertices (u, v) that are connected via (strictly) more than one path. If there exists no such pair, the subgraph is called a tree. A spanning tree $\tau = (\mathcal{V}_\tau, \mathcal{E}_\tau, w_\tau)$ is a tree whose vertex set $\mathcal{V}_\tau$ is equal to $\mathcal{V}$. A rooted spanning tree τ_r is a directed spanning tree where all edges are directed towards a node called the root. See Fig. 1 for illustrations.

Our work is related to a particular distribution on spanning trees called random spanning trees (RSTs). An RST T is a randomly generated spanning tree from the following distribution:

$$\mathbb{P}(T = \tau) \propto \prod_{(i,j)\in\tau} w(i, j) \tag{3}$$

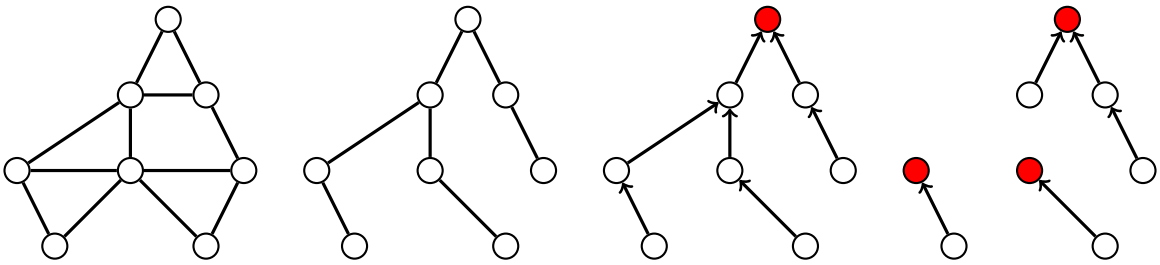


Fig. 1: From left to right, a graph $\mathcal{G}$, a spanning tree on $\mathcal{G}$, a rooted spanning tree on $\mathcal{G}$ and a rooted spanning forest on $\mathcal{G}$ (roots are colored in red)

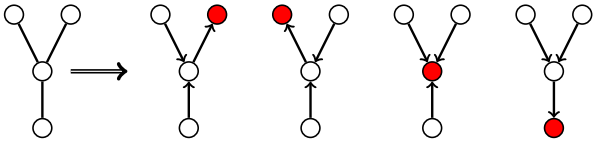


Fig. 2: All possible rooted spanning trees associated with a given undirected spanning tree. For four vertices, four different rooted trees exist.

Note that this distribution becomes uniform over all possible spanning trees whenever the given graph is unweighted *i.e.* $\forall i,j \in \mathcal{V}$, $w(i,j) \in \{0,1\}$:

$$\mathbb{P}(T=\tau) = \frac{1}{|\mathcal{T}|} \tag{4}$$

where $\mathcal{T}$ is the set of all spanning trees. In this particular case, the random tree T is also known as a uniform spanning tree (UST) in the literature.

In his celebrated work [23], Wilson proposes an algorithm, called `RandomTreeWithRoot`, that samples a random spanning tree from the set of all spanning trees rooted in node r. Wilson also shows that, in the case of undirected graphs, sampling an unrooted RST amounts to: i/ choosing uniformly a root, ii/ running `RandomTreeWithRoot`, and iii/ erasing the orientation.

C. Random spanning forests

A forest is a set of disjoint trees. When all the trees in a forest are rooted, it is called a rooted forest. A rooted spanning forest, generically denoted by ϕ, reaches all the vertices in the graph. Let ρ be the function that maps any rooted spanning forests to its set of roots. The number of roots $|\rho(\phi)|$ is between 1 and n. For $|\rho(\phi)|=1$, ϕ corresponds to a rooted spanning tree. See Fig. 1 for illustrations.

Random Spanning Forests. Let $\mathcal{F}$ be the set of all rooted spanning forests. A random spanning forest (RSF) is a random variable whose outcome space is $\mathcal{F}$. Among many possible options, we focus on the following parametric distribution for RSFs. For a fixed parameter $q>0$, Φ_q is a random variable in $\mathcal{F}$ verifying:

$$\forall \phi \in \mathcal{F}, \quad \mathbb{P}(\Phi_q = \phi) \propto q^{|\rho(\phi)|} \prod_{(i,j)\in\phi} w(i,j). \tag{5}$$

An algorithm [22] to sample from this distribution is derived from `RandomTreeWithRoot`. This algorithm:

1) extends the graph $\mathcal{G}=(\mathcal{V},\mathcal{E},w)$ by adding a node called Γ.
2) connects each node i in $\mathcal{V}$ to Γ with an edge of weight $w(i,\Gamma)=q$.
3) runs `RandomTreeWithRoot` by setting Γ as the root to obtain a spanning tree rooted in Γ in the extended graph.
4) deletes the edges incident to Γ in the obtained tree to yield a forest in the original graph.

The result is a rooted spanning forest whose root set is formed by the nodes which were neighbors of Γ. For every distinct spanning tree rooted at Γ, a distinct spanning forest is obtained after removing the root and its incident edges. Using this one-to-one relation ensures that this algorithm indeed samples from the distribution in (5):

$$\begin{aligned}\mathbb{P}(T_\Gamma=\tau_\Gamma) = \mathbb{P}(\Phi_q=\phi) &\propto \prod_{(i,j)\in\tau_\Gamma} w(i,j) \\ &\propto \prod_{(i,\Gamma)\in\tau_\Gamma} q \prod_{\substack{(i,j)\in\tau_\Gamma \\ i,j\neq\Gamma}} w(i,j) \\ &\propto q^{|\rho(\phi)|} \prod_{(i,j)\in\phi} w(i,j)\end{aligned} \tag{6}$$

An implementation of this algorithm is detailed in Algorithm 1. In the algorithm, `rand` (line 7) returns a uniform random value between 0 and 1 and `RandomSuccessor` (line 11) returns a random node i from $\mathcal{N}(u)$ with probability $\frac{w(u,i)}{\sum_{j\in\mathcal{N}(u)} w(u,j)}$. At termination, the array Next contains all the necessary information to build the sampled spanning forest.

The expected run time of `RandomForest` is the expected number of calls of `RandomSuccessor` before termination. For `RandomTreeWithRoot`, the number of calls equals to the mean commute time, *i.e.*, the expected length of a random walk going from node i to j and back (see theorem 2 in [23]). In proposition 1 of [25], Marchal rewrites this commute time in terms of graph matrices. Adapting his result to the current setting, the expected run time of `RandomForest` can be shown to equal the trace of $\big((\mathsf{L}+q\mathsf{I})^{-1}(\mathsf{D}+q\mathsf{I})\big)$, a rough upper-bound of which is $n+2|\mathcal{E}|/q$, which is linear with the number of edges.

Varying q over nodes. The original graph can also be extended by setting $w(i,\Gamma) \leftarrow q_i > 0, \forall i \in \mathcal{V}$, that is, by connecting the added node Γ with links of unequal weights.

Algorithm 1 `RandomForest`

Inputs:
  $\mathcal{G} = (\mathcal{V}, \mathcal{E}, w)$
  $q \in \mathbb{R}^+$
Initialize:
  # Initially, the forest is empty
  $\forall i \in \mathcal{V}$, InForest$[i] \leftarrow$ `false`
  $\forall i \in \mathcal{V}$, Next[i] $\leftarrow -1$
  $\forall i \in \mathcal{V}$, d$[i] \leftarrow \sum_{j \in \mathcal{N}(i)} w(i,j)$ *# Degrees*
for $i \leftarrow 1$ to $|\mathcal{V}|$ **do**
  $u \leftarrow i$
  # Start a random walk to create a forest branch
  while not InForest$[u]$ **do** *# Stop if u is in the forest*
    if `rand` $\leq \frac{q}{q+\text{d}[u]}$ **then** *#If true, u becomes a root*
      InForest$[u] \leftarrow$ `true` *# Add u to the forest*
      Next$[u] \leftarrow -1$ *# Set next of u to null*
    else *# If false, continue the random walk*
      Next[u] $\leftarrow$ `RandomSuccessor`$(u, \mathcal{G})$
      $u \leftarrow$ Next$[u]$
    end if
  end while
  $u \leftarrow i$ *# Go back to the initial node*
  # Add the newly created branch to the forest
  while not InForest$[u]$ **do**
    InForest$[u] \leftarrow$ `true`
    $u \leftarrow$ Next$[u]$
  end while
end for
`return` Next

In this case, the distribution of sampled forests becomes:

$$\mathbb{P}(\Phi_Q = \phi) \propto \prod_{i \in \rho(\phi)} q_i \prod_{(i,j) \in \phi} w(i,j), \quad \phi \in \mathcal{F} \tag{7}$$

where $Q = \{q_1, q_2, \ldots, q_n\}$ is the collection of parameters. Algorithm 1 can easily be adapted by modifying the scalar input q to $Q = \{q_1, q_2, \ldots, q_n\}$ and $\frac{q}{q+d[u]}$ to $\frac{q_u}{q_u+d[u]}$ at the step of root selection (line 7). In addition, the average run time in this case becomes $\text{tr}\left((\mathsf{L}+\mathsf{Q})^{-1}(\mathsf{D}+\mathsf{Q})\right)$, with $\mathsf{Q} = \text{diag}(q_1, \ldots, q_n)$.

Random partitions. A partition of $\mathcal{V}$, denoted by $\mathcal{P}$, is a set of disjoint subsets whose union equals $\mathcal{V}$. The trees of Φ_q give a random partition of $\mathcal{V}$ by splitting it into $|\rho(\Phi_q)|$ disjoint subsets. Let us enumerate the trees from 1 to $|\rho(\Phi_q)|$ and denote the vertex set of the k-th tree as $\mathcal{V}_k \subset \mathcal{V}$. Let π be a function that outputs the partition for a given spanning forest. Then, the random partition of $\mathcal{V}$ derived from Φ_q is $\pi(\Phi_q) = (\mathcal{V}_1, \ldots, \mathcal{V}_{|\rho(\Phi_q)|})$ with $|\pi(\Phi_q)|$ subsets. Note that this function is a many-to-one mapping because different spanning forests may correspond to the same partition (see Figure 3).

D. Useful properties of Φ_q

Recent studies in [21], [22] have established some theoretical properties of Φ_q and we reproduce here a few results.

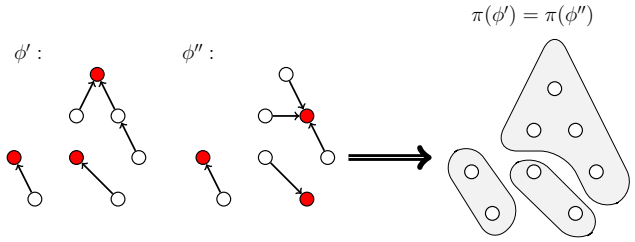


Fig. 3: Two different rooted spanning forests (on the left) with the same corresponding partition (on the right)

The root process. To start with, Proposition 2.2 in [21] states that $\rho(\Phi_q)$ is sampled from a determinantal point process (DPP) [26] with marginal kernel:

$$\mathsf{K} = (q\mathsf{I} + \mathsf{L})^{-1} q\mathsf{I} \tag{8}$$

This means that the inclusion probabilities verify:

$$\forall \mathcal{S} \subset \mathcal{V}, \quad \mathbb{P}(\mathcal{S} \in \rho(\Phi_q)) = \det \mathsf{K}_{\mathcal{S}}$$

where $\mathsf{K}_{\mathcal{S}} = [\mathsf{K}_{i,j} | (i,j) \in \mathcal{S} \times \mathcal{S}]$ is the submatrix of K reduced to the rows and columns indexed by $\mathcal{S}$.

Cardinality of $\rho(\Phi_q)$. As a consequence of $\rho(\Phi_q)$ being a DPP, the first two moments of $|\rho(\Phi_q)|$ verify [26]:

$$\begin{aligned} \mathbb{E}[|\rho(\Phi_q)|] &= \text{tr}(\mathsf{K}) = \sum_i \frac{q}{q+\lambda_i} \\ \text{Var}(|\rho(\Phi_q)|) &= \text{tr}(\mathsf{K} - \mathsf{K}^2) = \sum_i \frac{\lambda_i q}{(q+\lambda_i)^2} \end{aligned} \tag{9}$$

where the λ_i's are the eigenvalues of L.

The root probability distribution. Given any rooted spanning forest ϕ, define the *root function* $r_\phi : \mathcal{V} \to \rho(\phi)$ which takes as input any node i and outputs the root of the tree which i belongs to. In [21], [22], the authors show that the probability, for any node pair (i,j), that i is rooted in j reads:

$$\forall i, j \in \mathcal{V} \quad \mathbb{P}(r_{\Phi_q}(i) = j) = \mathsf{K}_{ij} \tag{10}$$

Conditioning on a partition. Let $t : \mathcal{V} \to \{1, 2, \ldots, |\rho(\Phi_q)|\}$ be a random mapping between any node and its tree number in Φ_q. (*e.g.*, $t(i) = k$ if $i \in \mathcal{V}_k \in \pi(\Phi_q)$). By conditioning the root probability over a fixed partition $\mathcal{P}$, one obtains (see Proposition 2.2 in [22]):

$$\forall i, j \in \mathcal{V} \quad \mathbb{P}(r_{\Phi_q}(i) = j | \pi(\Phi_q) = \mathcal{P}) = \frac{\mathbb{I}(j \in \mathcal{V}_{t(i)})}{|\mathcal{V}_{t(i)}|} \tag{11}$$

where $\mathbb{I}$ is the indicator function (*i.e.*, it outputs 1 if the input statement is true and 0 otherwise). In other words, given a fixed partition $\mathcal{P}$, the root probability within each subset $\mathcal{V}_k$ is uniform over the nodes in $\mathcal{V}_k$.

Extending to non-constant q. All of these properties are adaptable to the case of q varying over nodes (with some changes). The root process $\rho(\Phi_Q)$ is also a DPP. However, the associated marginal kernel becomes:

$$\mathsf{K} = (\mathsf{L}+\mathsf{Q})^{-1}\mathsf{Q} \quad \text{with} \quad \mathsf{Q} = \text{diag}(q_1, \ldots, q_n). \tag{12}$$

Notice that this kernel is not co-diagonalizable with the graph Laplacian L. Thus, the expected number of roots $\mathbb{E}[|\rho(\Phi_q)|]$ is not writable in terms of λ_i's, but it is still equal to $\mathrm{tr}(\mathsf{K})$. Similarly, the root probability $\mathbb{P}(r_{\Phi_Q}(i) = j)$ remains $\mathsf{K}_{i,j}$ whereas the conditional probability in (11) becomes:

$$\forall i, j \in \mathcal{V} \quad \mathbb{P}(r_{\Phi_Q}(i) = j | \pi(\Phi_Q) = \mathcal{P}) = \frac{q_j \mathbb{I}(j \in \mathcal{V}_{t(i)})}{\sum_{k \in \mathcal{V}_{t(i)}} q_k} \tag{13}$$

III. RSF BASED ESTIMATORS

In this section, we present our main results. We first recall the graph Tikhonov regularization and interpolation problems. Then, we describe the RSF-based methods to solve them. We also provide some theoretical analysis of the performance of the methods. Finally, we show how to tune hyperparameters for the proposed estimators.

Graph Tikhonov regularization. For a given graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, w)$ and measurements $\mathbf{y} = (y_1, y_2, \dots, y_n)^\top$ on the $|\mathcal{V}| = n$ vertices, the Tikhonov regularization of $\mathbf{y}$ reads:

$$\hat{\mathbf{x}} = \operatorname*{argmin}_{\mathbf{z} \in \mathbb{R}^n} \mu ||\mathbf{y} - \mathbf{z}||^2 + \mathbf{z}^\top \mathsf{L} \mathbf{z} \tag{14}$$

where $\mathsf{L} \in \mathbb{R}^{n \times n}$ is the graph Laplacian of $\mathcal{G}$. The solution of this minimization problem is:

$$\hat{\mathbf{x}} = \mathsf{K}\mathbf{y} \text{ with } \mathsf{K} = (\mathsf{L} + \mu \mathsf{I})^{-1} \mu \mathsf{I}.$$

Interestingly, the matrix in this solution also appears in (8) as the marginal kernel of the root process. This correspondence plays a significant role for the proposed methods by connecting RSFs to the Tikhonov regularization problem.

In some important cases, instead of $(\mathsf{L} + \mu \mathsf{I})^{-1} \mu \mathsf{I}\, \mathbf{y}$, the generalized solution $(\mathsf{L} + \mathsf{Q})^{-1} \mathsf{Q}\, \mathbf{y}$ is required where Q is an entry-wise non-negative diagonal matrix. For example, if we write the Tikhonov regularization of (14) with another graph Laplacian such as the random walk Laplacian $\mathsf{L}_{rw} = \mathsf{D}^{-1}\mathsf{L}$, then the solution reads $\hat{\mathbf{x}} = (\mathsf{L} + \mathsf{Q})^{-1} \mathsf{Q}\, \mathbf{y}$ where $\mathsf{Q} = \mu \mathsf{D}$. Another example occurs when the noise variance is known to be non-constant over vertices, *i.e.* heteroscedastic noise. The measurements may be less reliable at some vertices compared to others, meaning that there are different noise variances $\sigma_1, \dots, \sigma_n$. This implies that $q_1 \dots, q_n$ should be set proportional to $\frac{1}{\sigma_1}, \dots, \frac{1}{\sigma_n}$ in the estimation of $\hat{\mathbf{x}}$. This again corresponds to the generalized formulation.

Graph interpolation. Given a connected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, w)$, a parameter $\mu \geq 0$, and $\ell \subset \mathcal{V}$ a set of nodes where a signal $\mathbf{x}$ is known, the interpolated signal reads:

$$\begin{aligned} \hat{\mathbf{x}} &= \arg\min_{z \in \mathbb{R}^n} \mathbf{z}^\top (\mathsf{L} + \mu \mathsf{I}) \mathbf{z} \\ &\text{subject to } \forall i \in \ell,\ z_i = x_i \end{aligned} \tag{15}$$

Define $u = \mathcal{V} \backslash \ell$ the set of nodes for which $\mathbf{x}$ is not known and write L in block form:

$$\mathsf{L} = \begin{bmatrix} \mathsf{L}_{\ell|\ell} & \mathsf{L}_{\ell|u} \\ \mathsf{L}_{u|\ell} & \mathsf{L}_{u|u} \end{bmatrix}$$

where $\mathsf{L}_{\ell|u}$ is the Laplacian reduced to its rows and columns indexed by ℓ and u, respectively. The solution of (15) reads:

$$\hat{\mathbf{x}} = \begin{cases} x_i & \text{if } i \in \ell \\ \left(-(\mathsf{L}_{u|u} + \mu \mathsf{I})^{-1} \mathsf{L}_{u|l}\, \mathbf{x}_\ell\right)_i & \text{otherwise} \end{cases} \tag{16}$$

where $\mathbf{x}_\ell \in \mathbb{R}^{|\ell|}$ is the signal $\mathbf{x}$ reduced to its entries in ℓ. This solution can almost always[2] be rewritten as:

$$\hat{\mathbf{x}}_u = \mathsf{K}\mathbf{y} \quad \text{with } \begin{cases} \mathsf{K} = (\mathsf{L}_{\mathcal{G}\backslash\ell} + \mathsf{Q})^{-1}\mathsf{Q} \\ \mathbf{y} = -\mathsf{Q}^{-1}\mathsf{L}_{u|\ell}\mathbf{x}_\ell, \end{cases} \tag{17}$$

where $\mathsf{L}_{\mathcal{G}\backslash\ell}$ is the Laplacian of the reduced graph obtained by removing the vertices (and the incident edges) in ℓ, $\mathsf{Q} \in \mathbb{R}^{|u| \times |u|}$ is a diagonal matrix with $\mathsf{Q}_{i,i} = \mu + \sum_{j \in \ell} w(i,j)$. Similarly to graph TR, the RSF-based estimator for interpolation proposed in this paper draws upon the connection between Eqs. (17) and (12).

Parameter selection. The solution to graph TR tends to the constant vector (equal to the average of $\mathbf{y}$) as $\mu \to 0$, and to $\mathbf{y}$ for $\mu \to \infty$, where it suffers from underfitting and overfitting, respectively. In the interpolation problem, as $\mu \to \infty$, no prior information gets propagated through the other vertices, and $\hat{\mathbf{x}}_u$ tends to the zero vector. The case of $\mu = 0$ corresponds to solving the Dirichlet problem [9] which does not necessarily give the closest inference to the original signal $\mathbf{x}$. Due to these reasons, μ needs to be set at a value that gives the best approximation to the original signal. In both problems, choosing μ is a classical hyperparameter selection problem which can be approached in several ways for the proposed estimators.

In the following, we first present the proposed estimators for approximating $\hat{\mathbf{x}}$ for a fixed value of μ in Section III-A. Then, we outline in Section III-B several methods that select an appropriate μ automatically. Combining the hyperparameter selection and the RSF based estimators forms an RSF-based framework to approximate the solutions of graph Tikhonov regularization and graph interpolation.

A. RSF based estimation of $\hat{\mathbf{x}} = \mathsf{K}\mathbf{y}$

We propose two novel Monte Carlo estimators to approximate $\hat{\mathbf{x}} = \mathsf{K}\mathbf{y}$ with $\mathsf{K} = (\mathsf{L} + \mathsf{Q})^{-1}\mathsf{Q}$. These estimators leverage the probability distribution of the root process on RSFs presented in (10).

The first estimator, denoted by $\tilde{\mathbf{x}}$, is defined as follows:

$$\forall i \in \mathcal{V} \quad \tilde{x}(i) = y(r_{\Phi_Q}(i)) \tag{18}$$

In practice, a realization of Φ_Q is considered. Then, in each tree, the measurement of the root is propagated through the nodes of the tree. (See top-right in Fig. 4).

Proposition 1. *$\tilde{\mathbf{x}}$ is an unbiased estimator of $\hat{\mathbf{x}}$:*

$$\mathbb{E}[\tilde{\mathbf{x}}] = \hat{\mathbf{x}}.$$

[2] Q, as defined in the paragraph following (17), needs to be invertible for $\mathbf{y}$ to be well-defined. This is always the case if $\mu > 0$. When $\mu = 0$, it may not be the case. We will see in Section IV-A what can be done in this scenario.

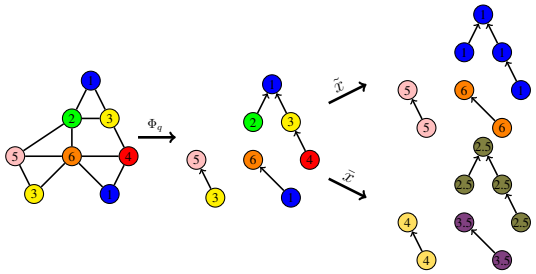


Fig. 4: An illustration for the estimators where q is constant over all nodes. In the left, the graph signal is interpreted by both colors and numbers. In the middle, a realization of Φ_q, a forest, is illustrated. On this forest, the estimators $\tilde{\mathbf{x}}$ and $\bar{\mathbf{x}}$ are illustrated in top-right and bottom right, respectively.

Moreover, the weighted expected error of $\tilde{\mathbf{x}}$ is:

$$\mathbb{E}\big(||\hat{\mathbf{x}}-\tilde{\mathbf{x}}||^2_{\mathsf{Q}}\big)=\sum_{i\in\mathcal{V}} q_i \operatorname{Var}(\tilde{x}(i)) = \mathbf{y}^\top(\mathsf{Q}-\mathsf{K}^\top\mathsf{Q}\mathsf{K})\mathbf{y}$$

where $||\mathbf{x}||^2_{\mathsf{Q}}=\mathbf{x}^\top\mathsf{Q}\mathbf{x}$.

Proof. For every node i, $\tilde{x}(i)$ is an unbiased estimator of $\hat{x}(i)$ thanks to the following:

$$\begin{aligned}\mathbb{E}\left[\tilde{x}(i)\right]=\mathbb{E}\left[y(r_{\Phi_Q}(i))\right]&=\sum_j \mathbb{P}(r_{\Phi_Q}(i)=j)y(j)\\&=\sum_j \mathsf{K}_{ij}y(j)=\boldsymbol{\delta}_i^\top\mathsf{K}\mathbf{y}=\hat{x}(i)\end{aligned}$$

where δ_i is the Kronecker delta (*i.e.* $\delta_i(i)=1$ and 0 otherwise). This result is prominently due to the root probability of RSFs given in (10). Also, the variance of $\tilde{x}(i)$ reads:

$$\operatorname{Var}(\tilde{x}(i))=\mathbb{E}\left[\tilde{x}(i)^2\right]-\mathbb{E}\left[\tilde{x}(i)\right]^2=\boldsymbol{\delta}_i^\top\mathsf{K}\mathbf{y}^{(2)}-(\boldsymbol{\delta}_i^\top\mathsf{K}\mathbf{y})^2.$$

where $y^{(2)}(k)=y(k)^2$, $\forall k\in\mathcal{V}$. Then, the weighted sum reads:

$$\sum_{i\in\mathcal{V}} q_i\operatorname{Var}(\tilde{x}(i))=\mathbf{1}^\top\mathsf{Q}\mathsf{K}\mathbf{y}^{(2)}-\mathbf{y}^\top\mathsf{K}^\top\mathsf{Q}\mathsf{K}\mathbf{y}$$

where $\mathbf{1}$ denotes the all-ones vector. Note that $\mathbf{1}^\top\mathsf{Q}\mathsf{K}=\mathbf{1}^\top\mathsf{K}^\top\mathsf{Q}$. Moreover, $\mathbf{1}$ is a left eigenvector of $\mathsf{K}^\top$ with corresponding eigenvalue 1. Then, the first term becomes $\mathbf{1}^\top\mathsf{K}^\top\mathsf{Q}\mathbf{y}^{(2)}=\mathbf{1}^\top\mathsf{Q}\mathbf{y}^{(2)}=\mathbf{y}^\top\mathsf{Q}\mathbf{y}$, and, one obtains:

$$\sum_{i\in\mathcal{V}} q_i\operatorname{Var}(\tilde{x}(i))=\mathbf{y}^\top(\mathsf{Q}-\mathsf{K}^\top\mathsf{Q}\mathsf{K})\mathbf{y} \tag{19}$$

□

The second estimator, denoted by $\bar{\mathbf{x}}$, is the expectation of $\tilde{\mathbf{x}}$ conditioned on the partition induced by Φ_Q:

$$\bar{x}(i)=\mathbb{E}[\tilde{x}(i)|\pi(\Phi_Q)=\mathcal{P}]=\frac{\sum\limits_{j\in\mathcal{V}_{t(i)}} y(j)q_j}{\sum\limits_{j\in\mathcal{V}_{t(i)}} q_j} \tag{20}$$

Due to the law of iterated expectations, this estimator is also unbiased, moreover it has a reduced variance compared to $\tilde{x}(i)$ due to the law of total variance:

$$\operatorname{Var}(\tilde{x}(i))=\mathbb{E}[\operatorname{Var}(\tilde{x}(i)|\pi(\Phi_Q)=\mathcal{P})]+\operatorname{Var}(\bar{x}(i))$$

which implies $\operatorname{Var}(\tilde{x}(i))\geq\operatorname{Var}(\bar{x}(i))$. This idea of improving an estimator is often called Rao-Blackwellization [27], [28].

In practice, we again take a realization of Φ_Q and consider the corresponding partition $\pi(\Phi_Q)$. Then, we compute the weighted average of the measurements in each subset of $\pi(\Phi_Q)$. Then, we finally propagate these averages in each subset (see Fig. 4).

Proposition 2. *$\bar{\mathbf{x}}$ is an unbiased estimator of $\hat{\mathbf{x}}$:*

$$\mathbb{E}[\bar{\mathbf{x}}]=\hat{\mathbf{x}}$$

Moreover, the weighted expected error reads:

$$\mathbb{E}\left(||\hat{\mathbf{x}}-\bar{\mathbf{x}}||^2_{\mathsf{Q}}\right)=\sum_{i\in\mathcal{V}} q_i\operatorname{Var}(\bar{x}(i))=\mathbf{y}^\top(\mathsf{Q}\mathsf{K}-\mathsf{K}^\top\mathsf{Q}\mathsf{K})\mathbf{y}.$$

Proof. Let $\mathsf{S}=\left[\frac{q_j\mathbb{I}(j\in\mathcal{V}_{t(i)})}{\sum_{k\in\mathcal{V}_{t(i)}}q_k}\right]_{i,j}$ be a symmetric random matrix associated to the random partition $\pi(\Phi_Q)$. A simple matrix product shows that $\mathsf{S}^\top\mathsf{Q}\mathsf{S}=\mathsf{Q}\mathsf{S}$ by definition of S. Moreover, $\bar{x}(i)=\boldsymbol{\delta}_i^\top\mathsf{S}\mathbf{y}$ and the expectation of $\mathsf{S}_{i,j}$ over all possible partitions derived from Φ_Q is:

$$\begin{aligned}\mathbb{E}\left[\mathsf{S}_{i,j}\right]&=\sum_{\mathcal{P}\in\pi(\mathcal{F})}\frac{q_j\mathbb{I}(j\in\mathcal{V}_{t(i)})}{\sum_{k\in\mathcal{V}_{t(i)}}q_k}\mathbb{P}(\pi(\Phi_Q)=\mathcal{P})\\&=\sum_{\mathcal{P}\in\pi(\mathcal{F})}\mathbb{P}(r_{\Phi_Q}(i)=j|\pi(\Phi_Q)=\mathcal{P})\mathbb{P}(\pi(\Phi_Q)=\mathcal{P})\\&=\mathbb{P}(r_{\Phi_Q}(i)=j)=\mathsf{K}_{i,j}\end{aligned} \tag{21}$$

Similarly, the expectation of $\bar{x}(i)$ reads:

$$\mathbb{E}[\bar{x}(i)]=\mathbb{E}[\boldsymbol{\delta}_i^\top\mathsf{S}\mathbf{y}]=\boldsymbol{\delta}_i^\top\mathbb{E}[\mathsf{S}]\mathbf{y}=\boldsymbol{\delta}_i^\top\mathsf{K}\mathbf{y}=\hat{x}(i) \tag{22}$$

Thus, $\bar{\mathbf{x}}$ is unbiased. The expected error is also computed in a similar way:

$$\begin{aligned}\mathbb{E}\left(||\hat{\mathbf{x}}-\bar{\mathbf{x}}||^2_{\mathsf{Q}}\right)&=\sum_{i\in\mathcal{V}} q_i\operatorname{Var}(\bar{x}(i))\\&=\sum_{i\in\mathcal{V}} q_i\left(\mathbb{E}[(\boldsymbol{\delta}_i^\top\mathsf{S}\mathbf{y})^2]-\mathbb{E}[(\boldsymbol{\delta}_i^\top\mathsf{S}\mathbf{y})]^2\right)\\&=\sum_{i\in\mathcal{V}}\mathbf{y}^\top\mathbb{E}\left[q_i\mathsf{S}^\top\boldsymbol{\delta}_i\boldsymbol{\delta}_i^\top\mathsf{S}\right]\mathbf{y}-q_i\mathbf{y}^\top(\boldsymbol{\delta}_i^\top\mathbb{E}[\mathsf{S}])^2\mathbf{y}\\&=\mathbf{y}^\top\mathbb{E}\left[\mathsf{S}^\top\mathsf{Q}\mathsf{S}\right]\mathbf{y}-\mathbf{y}^\top(\mathbb{E}[\mathsf{S}]^\top\mathsf{Q}\mathbb{E}[\mathsf{S}])\mathbf{y}\\&=\mathbf{y}^\top(\mathbb{E}\left[\mathsf{S}^\top\mathsf{Q}\mathsf{S}\right]-\mathsf{K}^\top\mathsf{Q}\mathsf{K})\mathbf{y}\end{aligned} \tag{23}$$

Finally, rewriting $\mathsf{S}^\top\mathsf{Q}\mathsf{S}=\mathsf{Q}\mathsf{S}$, one has:

$$\mathbb{E}\left(||\hat{\mathbf{x}}-\bar{\mathbf{x}}||^2_{\mathsf{Q}}\right)=\mathbf{y}^\top(\mathbb{E}\left[\mathsf{Q}\mathsf{S}\right]-\mathsf{K}^\top\mathsf{Q}\mathsf{K})\mathbf{y}=\mathbf{y}^\top(\mathsf{Q}\mathsf{K}-\mathsf{K}^\top\mathsf{Q}\mathsf{K})\mathbf{y}$$

□

Sample Mean. The sample mean of an unbiased Monte Carlo estimator over different realizations has a reduced variance, and so, gives a better estimator. Thus, in the rest, we use the sample means $\frac{1}{N}\sum_{k=1}^N\tilde{\mathbf{x}}_{\Phi_Q^{(k)}}$ and $\frac{1}{N}\sum_{k=1}^N\bar{\mathbf{x}}_{\Phi_Q^{(k)}}$ over N forest realizations as the outputs of the RSF based methods.

A remark. Reducing these results to the constant q case ($\mathsf{Q}=q\mathsf{I}$), one recovers the preliminary results presented in [24].

B. Parameter selection for the RSF estimators

The proposed estimators are efficient tools to approximate $\hat{\mathbf{x}}$ in both graph TR and interpolation problems for a fixed value of μ. However, as usual in these problems, a difficult question is the tuning of the hyper-parameter: the choice of μ that yields the best performance. For linear smoothers such as the one we have at hand ($\hat{\mathbf{x}} = \mathsf{K}\mathbf{y}$), many methods such as AIC, BIC, Marlow's Cp, leave-one-out cross validation (LOOCV), generalized cross validation (GCV) or Stein's unbiased risk estimator (SURE) are readily available for this tuning step (for more details and motivations, we refer the reader to [29]).

All of these methods need to compute a quantity called the effective number of parameters or the degree of freedom [29], which equals $\mathrm{tr}(\mathsf{K})$ for linear smoothers of the form $\mathsf{K}\mathbf{y}$. Computing exactly this trace requires the matrix inversion we wish to avoid from the start. A classical estimator of this quantity is Girard's estimator [30] (also known as Hutchinson's estimator [31]). We showed in [32] that RSFs can also be used to efficiently estimate $\mathrm{tr}(\mathsf{K})$. In this section, we build upon these preliminary results to show how the SURE and LOOCV methods can be adapted to the proposed estimators in order to select a good value of μ. Other methods are adaptable in a similar fashion.

Stein's Unbiased Risk Estimator. Given independent noisy measurements $\mathbf{y} = \mathbf{x} + \epsilon \in \mathbb{R}^n$ with a Gaussian noise $\epsilon_i \propto \mathcal{N}(0, \sigma^2)$, let $\theta(\mathbf{y})$ be an estimate for the unknown quantity $\mathbf{x}$. $\texttt{SURE}(\mathbf{y}, \theta(\mathbf{y}))$ provides an unbiased estimate of the expected error $\mathbb{E}_\epsilon[||\theta(\mathbf{y}) - \mathbf{x}||_2^2]$. For the linear smoother $\theta(\mathbf{y}) = \mathsf{K}\mathbf{y}$ with $\mathsf{K} = (\mathsf{Q} + \mathsf{L})^{-1}\mathsf{Q}$, the generic formula of SURE in [33] can be adapted as:

$$\texttt{SURE}(\mathbf{y}, \theta(\mathbf{y})) = -n\sigma^2 + ||\mathbf{y} - \theta(\mathbf{y})||_2^2 + 2\sigma^2\,\mathrm{tr}(\mathsf{K}) \quad (24)$$

where the degree of freedom term is replaced with $\mathrm{tr}(\mathsf{K})$. The theory behind relies on Stein's lemma on multivariate Gaussians [34]. Note that this method requires prior knowledge on the noise variance σ^2 and it outputs an unbiased estimation of the error. Then, this error needs to be evaluated for different values of μ and select the value yielding the smallest error.

SURE for RSF estimators. Similar to $\hat{\mathbf{x}}$, the RSF estimators too need the value of μ that gives the best performance. For this purpose, SURE can be used. In the following derivations, we present the adapted SURE formula for $\tilde{\mathbf{x}}$ and $\bar{\mathbf{x}}$. Moreover, these derivations show that numerically computing this formula is trivial after sampling N spanning forests.

Consider two random matrices $\tilde{\mathsf{S}} = \left[\mathbb{I}(r_{\Phi_Q}(i) = j)\right]_{i,j}$ and $\bar{\mathsf{S}} = \left[\frac{q_j \mathbb{I}(j \in \mathcal{V}_{t(i)})}{\sum_{k \in \mathcal{V}_{t(i)}} q_k}\right]_{i,j}$ (previously defined as S in the proof of Prop. 2). With these definitions, notice that $\tilde{\mathbf{x}} = \tilde{\mathsf{S}}\mathbf{y}$ and $\bar{\mathbf{x}} = \bar{\mathsf{S}}\mathbf{y}$. Moreover, the proposed estimators can be written in the form of linear smoothers:

$$\begin{aligned} \tilde{\theta}(\mathbf{y}) &= \frac{1}{N}\sum_{k=1}^{N} \tilde{\mathbf{x}}_{\Phi_Q^{(k)}} = \frac{1}{N}\sum_{k=1}^{N} \tilde{\mathsf{S}}^{(k)}\mathbf{y} \\ \bar{\theta}(\mathbf{y}) &= \frac{1}{N}\sum_{k=1}^{N} \bar{\mathbf{x}}_{\Phi_Q^{(k)}} = \frac{1}{N}\sum_{k=1}^{N} \bar{\mathsf{S}}^{(k)}\mathbf{y} \end{aligned} \quad (25)$$

where superscript (k) denotes k-th realization of $\tilde{\mathsf{S}}$ or $\bar{\mathsf{S}}$.

Then, one can also evaluate the formula in (24) for $\tilde{\theta}(\mathbf{y})$ and $\bar{\theta}(\mathbf{y})$. For instance, we get for $\tilde{\theta}(\mathbf{y})$:

$$\texttt{SURE}(\mathbf{y}, \tilde{\theta}(\mathbf{y})) = -n\sigma^2 + ||\mathbf{y} - \tilde{\theta}(\mathbf{y})||_2^2 + 2\sigma^2\,\mathrm{tr}\left(\frac{1}{N}\sum_{k=1}^{N}\tilde{\mathsf{S}}^{(k)}\right) \quad (26)$$

The residual error is trivial to compute after sampling N spanning forests. Moreover, this is also the case for the degree of freedom term. A closer look at the trace shows:

$$\mathrm{tr}\left(\frac{1}{N}\sum_{k=1}^{N}\tilde{\mathsf{S}}^{(k)}\right) = \mathrm{tr}\left(\frac{1}{N}\sum_{k=1}^{N}\bar{\mathsf{S}}^{(k)}\right) = \frac{1}{N}\sum_{k=1}^{N}|\rho(\Phi_Q^{(k)})|$$

This result yields that the trace term can be replaced with the average number of roots in the computation of $\texttt{SURE}(\mathbf{y}, \tilde{\theta}(\mathbf{y}))$ or $\texttt{SURE}(\mathbf{y}, \bar{\theta}(\mathbf{y}))$. Thus, the SURE scores of both estimators are trivial to numerically compute after sampling N spanning forests.

Note that neither $\texttt{SURE}(\mathbf{y}, \tilde{\theta}(\mathbf{y}))$ nor $\texttt{SURE}(\mathbf{y}, \bar{\theta}(\mathbf{y}))$ is an unbiased estimator for $\texttt{SURE}(\mathbf{y}, \theta(\mathbf{y}))$. Moreover, the estimation errors read:

$$\begin{aligned} \mathbb{E}\left[\texttt{SURE}(\mathbf{y}, \tilde{\theta}(\mathbf{y}))\right] - \texttt{SURE}(\mathbf{y}, \theta(\mathbf{y})) &= \mathrm{Var}(\tilde{\theta}(\mathbf{y})) \geq 0 \\ \mathbb{E}\left[\texttt{SURE}(\mathbf{y}, \bar{\theta}(\mathbf{y}))\right] - \texttt{SURE}(\mathbf{y}, \theta(\mathbf{y})) &= \mathrm{Var}(\bar{\theta}(\mathbf{y})) \geq 0 \end{aligned} \quad (27)$$

Thus, $\texttt{SURE}(\mathbf{y}, \tilde{\theta}(\mathbf{y}))$ and $\texttt{SURE}(\mathbf{y}, \bar{\theta}(\mathbf{y}))$ are (with high probability) upper-bounds for $\texttt{SURE}(\mathbf{y}, \theta(\mathbf{y}))$. For large graphs, in which computing $\texttt{SURE}(\mathbf{y}, \theta(\mathbf{y}))$ is prohibitive, these upper bounds also might be useful since they can be obtained cheaply.

Leave-One-Out cross validation. LOOCV is a very simple method to select μ for interpolation problems. Let $\mathbf{x}_\ell \in \mathbb{R}^{|\ell|}$ be the known part of the original signal $\mathbf{x}$ over the vertices in $\ell \subset \mathcal{V}$ and θ be an estimator for interpolation. LOOCV computes the following score (See Chapter 5.5.1 in [29]):

$$\texttt{LOOCV}(\mathbf{x}_\ell, \theta(\mathbf{x}_\ell)) = \frac{1}{|\ell|}\sum_{i \in \ell}(\theta^{-i}(\mathbf{x}_\ell)_i - x_i)^2$$

where $\theta^{-i}(\mathbf{x}_\ell)$ is the estimation without using the i-th measurement. This method leaves x_i out at the estimation stage, and calculates the error on it. The overall score is the average error over the vertices in ℓ. Note that this method needs to compute the estimation $\theta^{-i}(\mathbf{x}_\ell)$ for each individual i which might not be computationally feasible. Fortunately, this formula simplifies to the following for linear estimators in the form of $\theta(\mathbf{y}) = \mathsf{K}\mathbf{y}$ [29]:

$$\texttt{LOOCV}(\mathbf{x}_\ell, \theta(\mathbf{y})) = \frac{1}{|\ell|}\sum_{i \in \ell}\left(\frac{\theta(\mathbf{x}_\ell)_i - x_i}{1 - \mathsf{K}_{i,i}}\right)^2 \quad (28)$$

which avoids re-computation.

LOOCV for RSF estimators. Similar to SURE, this score can be adapted for the RSF based estimators. For example, in case of $\tilde{\theta}(\mathbf{x}_\ell)$, it becomes:

$$\texttt{LOOCV}(\mathbf{x}_\ell, \tilde{\theta}(\mathbf{x}_\ell)) = \frac{1}{|\ell|}\sum_{i \in \ell}\left(\frac{\tilde{\theta}(\mathbf{x}_\ell)_i - x_i}{1 - \frac{1}{N}\sum_{k=1}^{N}\tilde{\mathsf{S}}_{i,i}^{(k)}}\right)^2 \quad (29)$$

and for $\bar{\theta}$, it can be derived in the same way. Notice that every element in this expression is numerically available after sampling N spanning forests. Thus, this score can be easily computed for both estimators.

IV. RSF estimators for other graph problems

In this section, we explore a few graph problems in which the RSF based estimators presented can replace expensive exact computations.

A. Node Classification in semi-supervised learning

Consider a dataset consisting of elements one wishes to classify. In the semi-supervised learning context, the class label of a few elements are supposed to be known *a priori*, along with a graph structure encoding some measure of affinity between the different elements: the larger the weight of the edge connecting two elements, the closer they are according to some application-dependent metric, the more likely these two elements belong to the same class. The goal is then to infer all the labels given this prior information.

Among many options to solve this problem, label propagation [8] and generalized SSL framework [11] are two well-known baseline approaches. In this section, we deploy $\tilde{\mathbf{x}}$ and $\bar{\mathbf{x}}$ to approximate the solutions given by these approaches.

Problem definition. Let us denote the labeled vertices by $\ell \subset \mathcal{V}$ (typically $|\ell| \ll |\mathcal{V}|$ and the unlabeled ones by $u = \mathcal{V} \backslash \ell$. Assume C distinct label classes and define the following binary encoding of the prior knowledge for the c-th class:

$$\forall i \in \mathcal{V}, \mathbf{y}_c(i) = \begin{cases} 1 \text{ if } i \text{ is known to belong to the } c\text{-th class} \\ 0 \text{ otherwise} \end{cases} \tag{30}$$

The matrix $\mathsf{Y} = [\mathbf{y}_1 | \dots | \mathbf{y}_C] \in \mathbb{R}^{n \times C}$ thus encodes the prior knowledge. Many approaches to SSL formulate the problem as follows. First, for each class c, compute the so-called "classification function", defined as:

$$\mathbf{f}_c = \arg \min_{\mathbf{z}_c \in \mathbb{R}^n} \mu \sum_{i \in \mathcal{V}} q'_i (y_c(i) - z_c(i))^2 + \mathbf{z}_c^\top \mathsf{L} \mathbf{z}_c \tag{31}$$

where μ and q'_i's are regularization parameters: μ sets the global regularization level, and each q'_i acts entry-wise (when q'_i is high, the corresponding entry in f_c is close to the measurement $y_c(i)$). Eq. (31) has the following explicit solution:

$$\mathbf{f}_c = (\mathsf{L} + \mathsf{Q})^{-1} \mathsf{Q} \, \mathbf{y}_c \tag{32}$$

where $\mathsf{Q} = \operatorname{diag}(q_1, \dots, q_n)$ and $q_i = \mu q'_i$. Thus, each classification function $\mathbf{f}_c$ can be viewed as a smoothed version of the prior knowledge encoded in $\mathbf{y}_c$. As such, if $f_c(i)$ is large, it implies that labels of class c are relatively dense around node i. The last step in these SSL algorithms is to assign each node i to the class $\arg\max_c f_c(i)$.

Label propagation and the generalized SSL framework are two algorithms that adapt this solution in different ways. In particular, by using different set of q'_i's in (31), label propagation is in fact a graph interpolation and the generalized SSL framework may be understood as a graph TR. Thus, the RSF estimators can be used to approximate the solution for both algorithms. In the following, we discuss the corresponding parameter settings for these algorithms along with their RSF versions.

Label Propagation. The label propagation algorithm [8] solves the Dirichlet problem for each class c, that is:

$$\begin{aligned} \forall i \in \mathcal{V}, \quad & \mathsf{L}\mathbf{f}_c(i) = 0 \\ \text{s. t. } \forall i \in \ell, \quad & \mathbf{f}_c(i) = \mathbf{y}_c(i) \end{aligned} \tag{33}$$

which is equivalent to (15) for $\mu = 0$. Defining the classification matrix $\mathsf{F} = [\mathbf{f}_1 | \dots | \mathbf{f}_C] \in \mathbb{R}^{n \times C}$, one thus has:

$$\mathsf{F}_{i,c} = \begin{cases} \mathsf{Y}_{i,c}, & \text{if } i \in \ell \\ (-(\mathsf{L}_{u|u})^{-1} \mathsf{L}_{u|\ell} \mathsf{Y}_{\ell|:})_{i,c}, & \text{otherwise} \end{cases} \tag{34}$$

where $\mathsf{Y}_{\ell|:}$ is the matrix Y restricted to rows in ℓ. Note in passing that F corresponds to a special set of functions for graphs called *harmonic functions*. Besides being the solution of Dirichlet boundary problem, they have interesting connections with electrical networks and random walks [8].

Zhu *et al.* [8] provide a simple algorithm to compute F without computing the inverse matrix. Starting from an arbitrary initial $\mathsf{F}^{(0)}$, at each iteration k, the algorithm updates $\mathsf{F}^{(k)} \leftarrow \mathsf{D}^{-1}\mathsf{W}\mathsf{F}^{(k-1)}$. The iteration is completed by setting the known labels $\mathsf{F}^{(k)}_{\ell|:}$ to $\mathsf{Y}_{\ell|:}$. They prove that the output of this iteration converges to F as $k \to \infty$ (see Section 2.3 in [35]).

Here, we provide an RSF-based estimator to approximate F. Two scenarii are possible. The first (unlikely) scenario is when any node in u is connected to at least one node in ℓ. In this case, one can rewrite (34) as:

$$\mathsf{F} = -\mathsf{K} \, \mathsf{Q}^{-1} \mathsf{L}_{u|\ell} \mathsf{Y} \quad \text{with } \mathsf{K} = (\mathsf{L}_{\mathcal{G} \backslash \ell} + \mathsf{Q})^{-1} \mathsf{Q} \tag{35}$$

where $\mathsf{L}_{\mathcal{G} \backslash \ell}$ is the Laplacian of the reduced graph obtained by removing the vertices (and the incident edges) in ℓ, $\mathsf{Q} \in \mathbb{R}^{|u| \times |u|}$ is a diagonal matrix with $\mathsf{Q}_{i,i} = \sum_{j \in \ell} w(i,j)$. The condition of this first scenario ensures that Q is indeed invertible; and RSFs on the reduced graph $\mathcal{G} \setminus \ell$ can thus estimate the columns of F.

However, when there exists at least one node in u that is not connected to ℓ, Q is no longer invertible and another approach is needed. In this second scenario, the parameters are defined over all vertices and set to $q_i = \alpha > 0$ for $i \in \ell$ and $q_i = 0$ for $i \in u$. The following proposition guarantees that as $\alpha \to \infty$, the RSF estimator $\tilde{\mathbf{x}}$ with this setting approximates the solution given in Eq (34).

Proposition 3. *Given the parameters $q_i = \alpha > 0$ for $i \in \ell$ and $q_i = 0$ for $i \in u$, as well as the input vector $\mathbf{y}_c \in \mathbb{R}^n$ for the RSF estimator $\tilde{\mathbf{x}}$, the following is verified:*

$$\lim_{\alpha \to \infty} \mathbb{E}\left[\tilde{\mathbf{x}}\right] = \mathbf{f}_c.$$

Proof. See the supplementary material for the detailed proof. □

From the random forest point-of-view, setting q_i to infinity for all nodes i in ℓ, and to 0 otherwise, implies that all possible realizations of Φ_Q have exactly the same root set: ℓ. Thus, when using the estimator $\tilde{\mathbf{x}}$, the measurements in ℓ are not altered and are simply propagated to other vertices via the

sampled random trees. In addition, the estimator $\bar{\mathbf{x}}$ boils down to $\tilde{\mathbf{x}}$ in this very specific case.

Generalized SSL framework. The generalized SSL framework proposed in [11] can be seen as a graph TR. It defines the classification function as follows:

$$\mathbf{f}_c = \frac{\mu}{\mu+2}\left(\mathsf{I} - \frac{2}{\mu+2}\mathsf{D}^{-\sigma}\mathsf{W}\mathsf{D}^{\sigma-1}\right)^{-1}\mathbf{y}_c$$

where $\mu > 0$ is the regularization parameter and σ determines which graph Laplacian is used: $\sigma = 0, 0.5, 1$ respectively correspond to the combinatorial, normalized and random walk graph Laplacian. This formula can also be written as:

$$\mathbf{f}_c = \mathsf{D}^{1-\sigma}\mathsf{K}\mathsf{D}^{\sigma-1}\mathbf{y}_c \text{ with } \mathsf{K} = (\mathsf{L}+\mathsf{Q})^{-1}\mathsf{Q}$$

where $\mathsf{Q} = \frac{\mu}{2}\mathsf{D}$. Notice that the cumbersome part in this formula is to compute $\mathsf{K}\mathsf{D}^{\sigma-1}\mathbf{y}_c$ and it can be approximated by the proposed estimators on the input vector $\mathbf{y} = \mathsf{D}^{\sigma-1}\mathbf{y}_c$. Then, $\mathbf{f}_c$ is obtained by left-multiplying the result by $\mathsf{D}^{1-\sigma}$.

Both solutions, label propagation and the generalized SSL, can be considered as two different versions of a more generic optimization problem. Label propagation puts a very high confidence on the prior. $\tilde{\mathbf{x}}$, for example, only propagates the measurements of the labeled vertices (from the second scenario's perspective). Whereas, in generalized SSL, lower confidence over the prior information is assumed, and thus, the propagation of other measurements, which are all set to 0 in this encoding, is authorized. The success of both methods depends on the correctness of these assumptions on the data. Section V provides empirical comparisons on benchmark datasets.

B. Non-quadratic convex functions and Newton's method

Consider the following generalized optimization problem:

$$\hat{\mathbf{x}} = \arg\min_{z\in\mathbb{R}^n} \mu f(\mathbf{z}) + \frac{1}{2}\mathbf{z}^\top\mathsf{L}\mathbf{z} \tag{36}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is a generic twice-differentiable function for the data fidelity term (*e.g.* previously $f(\mathbf{z}) = ||\mathbf{y}-\mathbf{z}||_2^2$). A common occurence of the generalised form above is when the function f above is a log-likelihood, i.e. $f(\mathbf{z}) = \sum_{i=1}^n \log p(y_i|z_i)$. This is used when the assumption that the observations are Gaussian (given the signal) is inappropriate, for instance when the observations are discrete. In such cases f is not a quadratic function and there is typically no closed-form solution for 36.

In these cases, iterative approaches are often deployed. These approaches draw an iteration scheme to minimize the objective or the loss function. One popular approach among them is Newton's method. Let $L(\mathbf{z})$ denote the loss function, then Newton's method draws the following iteration scheme:

$$\mathbf{z}_{k+1} = \mathbf{z}_k - \alpha(\mathsf{H}L(\mathbf{z}_k))^{-1}\nabla L(\mathbf{z}_k) \text{ with } \alpha\in[0,1],\ k = 1,2,\dots \tag{37}$$

H and ∇ are the Hessian and gradient operators, respectively and $\mathbf{z}_k$ denotes the estimation at iteration k. Note that, by definition of the Hessian operator, this method requires twice-differentiability for the loss function.

Given this scheme, the methods proposed here may become useful for approximating the inversion throughout the iterations. We illustrate this usage in the following setup:

Assume an independent Poisson distribution for each likelihood at i:

$$P(y_i|\lambda = z_i) = \frac{z_i^{y_i}\exp(-z_i)}{y_i!}$$

where λ is the distribution parameter. This assumption is often made in image processing applications to eliminate shot noise [36]. Also, consider the following slightly modified loss function:

$$L'(\mathbf{t}) = -\mu\sum_{i=1}^n \log P(y_i|\lambda = \exp(t_i)) + \frac{1}{2}\mathbf{t}^\top\mathsf{L}\mathbf{t} \tag{38}$$

where $\exp(t_i) = z_i$. The gradient $\nabla L'(\mathbf{t})$ reads:

$$\nabla L'(\mathbf{t}) = \mu\exp(\mathbf{t}) - \mu\mathbf{y} + \mathsf{L}\mathbf{t}$$

where $\exp$ operates entry-wise on vectors. Then, the Hessian matrix becomes:

$$\mathsf{H}L'(\mathbf{t}) = \mu\,\mathrm{diag}(\exp(\mathbf{t})) + \mathsf{L}$$

With these two ingredients, a Newton iteration scheme becomes:

$$\mathbf{t}_{k+1} = \mathbf{t}_k - \alpha\left[\mu\,\mathrm{diag}(\exp(\mathbf{t}_k)) + \mathsf{L}\right]^{-1}(\mu\exp(\mathbf{t}_k) - \mu\mathbf{y} + \mathsf{L}\mathbf{t}_k)$$

The update term, which requires an inverse operation, can be approximated by our RSF estimators. This approximation is achieved by setting $\mathsf{Q} = \mu\,\mathrm{diag}(\exp(\mathbf{t}_k))$ and the graph measurements $\mathbf{y}'$ to $\mu^{-1}\,\mathrm{diag}(\exp(-\mathbf{t}_k))(\mu\exp(\mathbf{t}_k) - \mu\mathbf{y} + \mathsf{L}\mathbf{t}_k)$. This particular case yields the following computation:

$$\begin{aligned}\mathsf{K}\mathbf{y}' &= (\mathsf{Q}+\mathsf{L})^{-1}\mathsf{Q}\mathbf{y}' \\ &= \left[\mu\,\mathrm{diag}(\exp(\mathbf{t}_k)) + \mathsf{L}\right]^{-1}(\mu\exp(\mathbf{t}_k) - \mu\mathbf{y} + \mathsf{L}\mathbf{t}_k)\end{aligned}$$

which equals to the update in Newton's method. Thus, the RSF estimators can be easily used to compute each update step with a cheap cost.

In the classical Newton's method *i.e.* $\alpha = 1$, convergence of the result is not guaranteed. It might diverge or stuck in a loop depending on the closeness of the initial point $\mathbf{t}_0$ to the solution. Guessing a good initial point is not an easy task and may require expensive computations in high dimensions. Instead, modifying α is a more applicable option to ensure convergence. Thus, Newton's method is often combined with an additional step at each iteration in which α is reset accordingly. Line search algorithms [12] are simple and well-understood methods for this purpose. At each iteration, if needed, they damp the applied update by shrinking α. These methods provide convergence, however, they may require more iteration steps w.r.t. the pure Newton's method with a good initial point. In our case, the updates are stochastic and exact convergence cannot be expected.

C. l_1-Regularization and iteratively reweighted least squares

As with the data fidelity term, many alternatives for the regularization term are also available. Among them, l_1-regularization [37] is often deployed to obtain sparser solutions. In [38], Sharpnack *et al.* adapts l_1-regularization for graphs as follows:

$$\hat{\mathbf{x}} = \underset{\mathbf{z}\in\mathbb{R}^n}{\operatorname{argmin}}\, \mu||\mathbf{y}-\mathbf{z}||^2 + ||\mathsf{B}\mathbf{z}||_1 \tag{39}$$

where B is the edge incidence matrix and $||\mathsf{B}\mathbf{z}||_1 = \sum_{(i,j)\in\mathcal{E}} w(i,j)^{1/2}|z_i - z_j|$. In contrast to the l_2 regularization, a closed form solution for l_1 case is not available. Thus, iterative optimization schemes are usually utilized to compute the solution. Iterative reweighted least square [13] (IRLS) method is one that can be easily adapted for our RSF based estimators. In this section, we derive the iteration scheme of IRLS for the given problem and show that each iteration step can be approximated by $\tilde{\mathbf{x}}$ and $\bar{\mathbf{x}}$.

Let $\mathsf{M} = \operatorname{diag}(\operatorname{abs}(\mathsf{B}\mathbf{z}))^{-1}$ where abs is the entry-wise absolute value operator. An iteration scheme can be derived by using this equality:

$$||\mathsf{B}\mathbf{z}||_1 = \operatorname{abs}(\mathbf{z}^\top\mathsf{B}^\top)\mathbf{1} = \operatorname{abs}(\mathbf{z}^\top\mathsf{B}^\top)\mathsf{M}\operatorname{abs}(\mathsf{B}\mathbf{z}) = \mathbf{z}^\top\mathsf{B}^\top\mathsf{M}\mathsf{B}\mathbf{z}$$

where $\mathbf{1}\in\mathbb{R}^m$ is the all-ones vector. Then, the problem can also be written as:

$$\hat{\mathbf{x}} = \underset{\mathbf{z}\in\mathbb{R}^n}{\operatorname{argmin}}\, \mu||\mathbf{y}-\mathbf{z}||^2 + \mathbf{z}^\top\mathsf{B}^\top\mathsf{M}\mathsf{B}\mathbf{z}^\top \tag{40}$$

which can be iteratively solved by the following scheme:

$$\mathbf{z}_{k+1} = (\mu\mathsf{I} + \mathsf{B}^\top\mathsf{M}_k\mathsf{B})^{-1}\mu\mathsf{I}\mathbf{y} \text{ with } \mathsf{M}_k = \operatorname{diag}(\operatorname{abs}(\mathsf{B}\mathbf{z}_k))^{-1}$$

A more detailed derivation and the convergence analysis of this scheme can be found in [13]. Notice that, by definition, $\mathsf{B}^\top\mathsf{M}_k\mathsf{B}$ equals to a reweighted graph Laplacian L_k. Then, computing the update at each iteration step immediately reduces to solve a graph Tikhonov regularization. Thus, $\tilde{\mathbf{x}}$ or $\bar{\mathbf{x}}$ can be used for estimating the update.

V. Experiments

We provide in this section several illustrations and run time analysis of the proposed methods. In the first illustration, the RSF based methods are run on two image denoising setups. In these, we consider

- An image with a Gaussian noise: solution provided by the RSF based Tikhonov regularization parameterized by SURE.
- An image with a Poisson noise: solution provided by the RSF based Newton's method coupled with line search.

In both setups, the underlying graph is assumed to be a 2-dimensional (2D) grid graph.

In the second illustration, the SSL node classification problem is considered on three benchmark datasets. We examine the classification performances of Tikhonov regularization, label propagation and the RSF versions of these algorithms. In these experiments, the parameter selection for the Tikhonov regularization is done by the RSF based leave-one-out cross validation.

Finally, we briefly examine the computational time for sampling a spanning forest which is the building block operation for the proposed estimators. Then, we compare this quantity with the computational time of computing $\mathsf{L}\mathbf{y}$ which is the building block for state-of-the-art approaches, polynomial approximations or iterative approaches.

A. Image Denoising

A 2D grid graph is a natural underlying structure for images: every pixel corresponds to a node and each pixel is connected to its four direct neighbors with equal weights. Other structures could be used (such as an 8-neighbour version of the grid graph) and performances will depend on the chosen structure. However, for the purpose of illustration, we will only consider the simplest 4-neighbour grid graph.

In the first setup, noisy (with additive Gaussian noise) measurements $\mathbf{y}$ of the original image $\mathbf{x}$ are given:

$$\mathbf{y} = \mathbf{x} + \epsilon \text{ with } \epsilon \propto \mathcal{N}(\mathbf{0}, \sigma^2\mathsf{I})$$

To recover $\mathbf{x}$, Tikhonov regularization is applied. Fig. 5 compares the exact result $\hat{\mathbf{x}} = \mathsf{K}\mathbf{y} = (\mathsf{L}+\mu\mathsf{I})^{-1}\mu\mathbf{y}$, to its forest-based approximations $\tilde{\mathbf{x}}$ and $\bar{\mathbf{x}}$. The SURE method to estimate the best value of μ is also illustrated for all three estimations of the original image $\mathbf{x}$ (and we remark in passing that they are consistent). The results confirms that $\bar{\mathbf{x}}$ produces a better estimate for $\hat{\mathbf{x}}$ than $\tilde{\mathbf{x}}$. Also, in Fig. 5d, the scores computed for the RSF estimators are observed as upper bounds of the scores for $\hat{\mathbf{x}}$ which is expected from the results in Eq (27).

In the second setup, each pixel value is assumed to be sampled from a Poisson distribution whose mean is the true value of the pixel:

$$\mathbf{y} \propto \text{Poisson}(\mathbf{x})$$

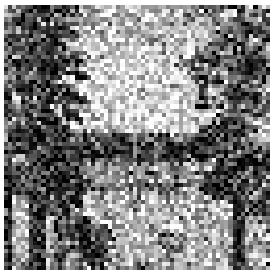
To reconstruct $\mathbf{x}$, Newton's method is applied, as explained in Section IV-B. The line search algorithm is used for picking the value of α at each iteration to ensure convergence. Both qualitative and quantitative results in Fig. 6 show that all methods converge to the same solution. Fig. 6c shows that, even though all three methods converge in less than 100 iterations, $\hat{\mathbf{x}}$ and $\bar{\mathbf{x}}$ minimize the loss function in a similar way whereas $\tilde{\mathbf{x}}$ diverges from this pattern and requires more iterations for the convergence (due to a larger approximation error at each step).

B. Node Classification

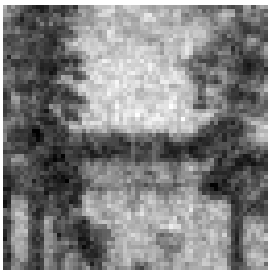
In this illustration, we run our methods to solve the node classification problem discussed in Section IV-A. For the generalized SSL framework, σ is set to 0, and μ is set by RSF based cross-validation.

The experiments are done on three standard benchmark datasets [39], namely Cora, Citeseer and Pubmed. In the first two, the underlying graphs are disconnected, thus we use the largest connected components. Also, in all datasets, the orientations of the edges are omitted to operate on undirected graphs. The general statistics of these datasets after the pre-processing are summarized in Table I. Note that in these three

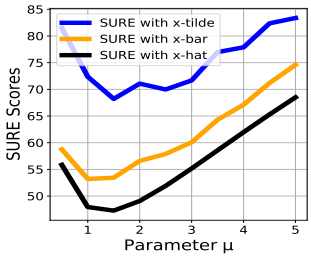
(a) Original Image $\mathbf{x}$



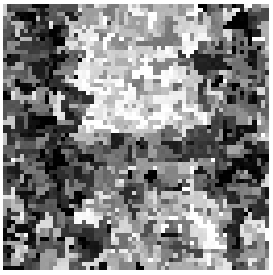
(b) Noisy Measurments $\mathbf{y}$, PSNR = 14.2



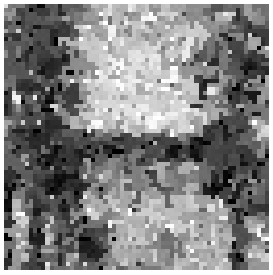
(c) Exact solution $\hat{\mathbf{x}}$, PSNR = 19.4



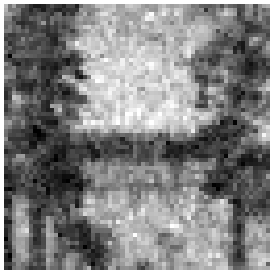
(d) SURE scores vs μ



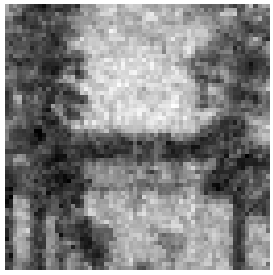
(e) $\tilde{\mathbf{x}}$ with $N = 1$



(f) $\bar{\mathbf{x}}$ with $N = 1$



(g) $\tilde{\mathbf{x}}$ with $N = 20$, PSNR = 18.6

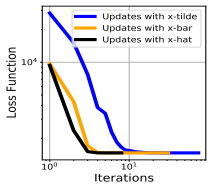


(h) $\bar{\mathbf{x}}$ with $N = 20$, PSNR = 19.1

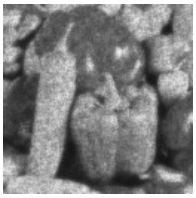
Fig. 5: An image denoising experiment with additive Gaussian noise. a) the original image. b) a noisy version $\mathbf{y} = \mathbf{x} + \epsilon$ with $\epsilon\ \mathcal{N}(\mathbf{0}, \sigma)$. c) the exact graph TR $\hat{\mathbf{x}} = \mathsf{K}\mathbf{y}$. Figure d) summarizes the SURE scores of $\hat{\mathbf{x}}$,$\tilde{\mathbf{x}}$ and $\bar{\mathbf{x}}$ for different μ's. In each, the value of μ that minimizes the SURE score is selected. e-f) the two RSF estimates $\tilde{\mathbf{x}}$ and $\bar{\mathbf{x}}$ based on only one sampled forest. g-h) same as e-f) but averaged over $N = 20$ sampled forests.

(a) Original image $\mathbf{x}$

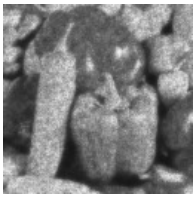
(b) Noisy image $\mathbf{y}$, PSNR=18.0



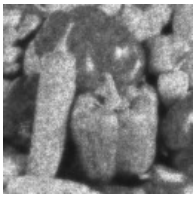
(c) Loss function through iterations



(d) Denoised image with $\tilde{\mathbf{x}}$ with $N = 40$, PSNR=23.1



(e) Denoised image with $\bar{\mathbf{x}}$ with $N = 40$, PSNR=23.1



(f) Denoised image with $\hat{\mathbf{x}}$, PSNR=23.1

Fig. 6: An image denoising experiment with Poisson noise. a) original image $\mathbf{x}$. b) a noisy version $\mathbf{y} \propto$ Poisson$(\mathbf{x})$. Newton's method is deployed to recover $\mathbf{x}$ by minimizing the loss function in 38. For three update options, namely $\hat{\mathbf{x}}$, $\tilde{\mathbf{x}}$ and $\bar{\mathbf{x}}$, Newton's method yields the results shown on the bottom line. Figure c) shows the loss function through the iterations for the three update options.

datasets, the class of each node is known. This will enable us to test the different SSL frameworks (a small arbitrary fraction of nodes will serve as pre-labeled nodes, and one tests whether or not this is sufficient to infer the class of all nodes). More precisely, we use the following procedure:

TABLE I: SSL dataset statistics after preprocessing

Dataset	#Nodes	#Edges	#Classes
Citeseer	2110	3668	6
Cora	2485	5069	7
Pubmed	19717	44324	3

- m vertices are selected at random per class as the labeled nodes,
- the parameter μ is set by LOOCV separately for $\hat{\mathbf{x}}$, $\tilde{\mathbf{x}}$ and $\bar{\mathbf{x}}$.
- the classification functions $\mathbf{f}_c$ for each class c are computed by the generalized SSL framework, label propagation and their RSF versions averaged over N repetitions,
- for each vertex i, we assign $\arg\max_c \mathsf{F}_{i,c}$ as its class and calculate the classification accuracy as the ratio of correctly predicted labels to the total number of predictions.

In Fig. 7, the classification accuracy is reported as m and N vary. The results are averaged over 50 realizations of the m labeled vertices for all datasets.

In all experiments, $\bar{\mathbf{x}}$ performs better than $\tilde{\mathbf{x}}$ as expected. Moreover, for the first two datasets, Cora and Citeseer, $\bar{\mathbf{x}}$ has a comparable performance with the exact solution. However, in Pubmed, $\bar{\mathbf{x}}$ fails to perform as good as the $\hat{\mathbf{x}}$ for gSSL due to larger approximation errors in both the parameter selection and the estimation steps.

The empirical results yield that the proposed methods need much less forest realizations to reach the exact solution of LP rather than the generalized SSL. However, sampling a forest for LP often takes more time if n is large and m is relatively small. For example, in the Pubmed graph, for

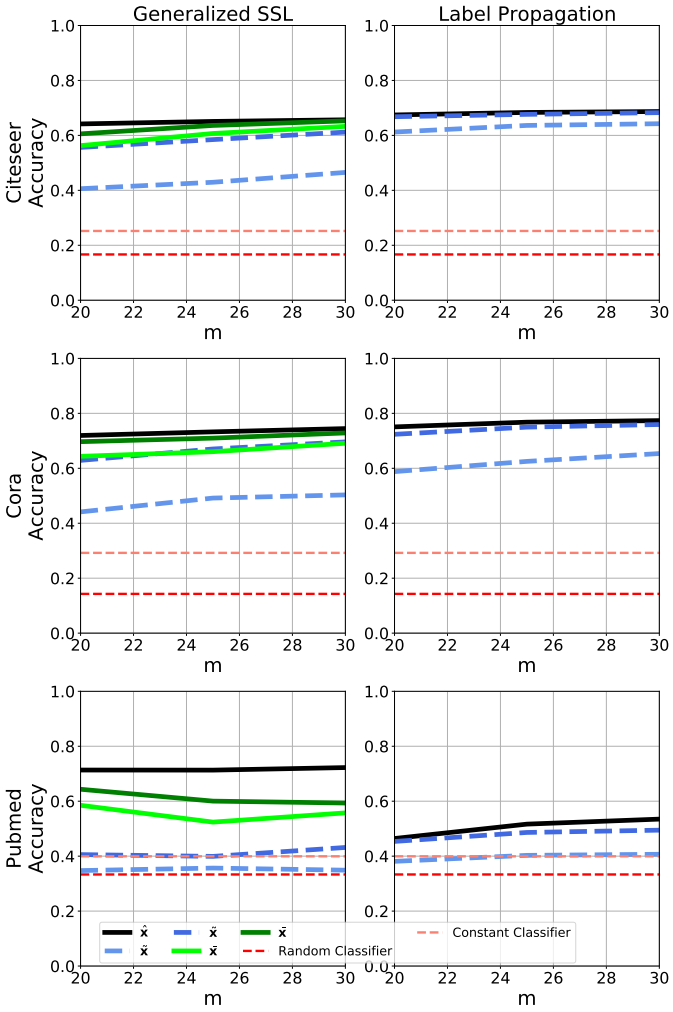


Fig. 7: The classification accuracy of the generalized SSL, LP and their RSF variants are presented on the datasets Citeseer, Cora and Pubmed. The RSF methods for the generalized SSL are illustrated for $N = 50,\ 100,\ 500$ forest realizations, whereas, these numbers for LP are $N = 5,\ 10,\ 50$. In these plots, the random classifier denotes the accuracy of inferring classes at random and the constant classifier is the accuracy of assigning the most occurring class to all unlabeled vertices. The results for Citeseer, Cora and Pubmed datasets are averaged over respectively 50 different set of labeled vertices.

$m = 20$, sampling a single forest for LP (resp. the generalized SSL) takes 6.3×10^{-2} (resp. 1.4×10^{-3}) seconds averaged over 100 repetitions in a single threaded run time of a laptop. Note that these figures strongly depend on m and the given network. Thus, one needs to examine this trade-off with the given dataset to adjust the total run time.

C. Run-time Analysis

Sampling a spanning forest is the repeated core operation in the proposed estimators. Similarly, the matrix product $\mathsf{L}\mathbf{y}$ is the building block operation for the state-of-the-art, polynomial approximations and iterative methods. In this section, we compare the times needed for these operations to give an order of magnitude for the run-time of the proposed methods.

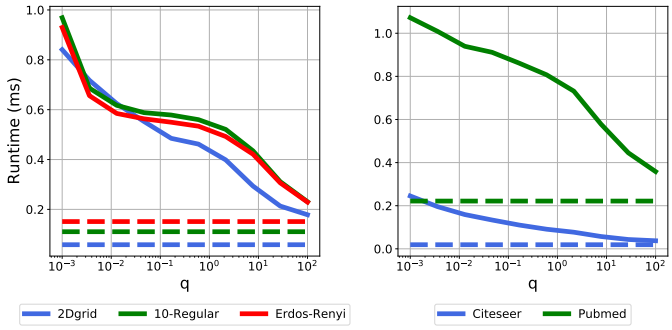


Fig. 8: Average computational times for the RSF sampling (solid lines) and the sparse matrix multiplication $\mathsf{L}\mathbf{y}$ (dashed lines) are compared. The entries of $\mathbf{y}$ are chosen uniformly between 0 and 1. On the left plot, the underlying graphs are 2D grid with 10^4 nodes, 10-regular graph (every node has exactly 10 neighbors) with 10^4 nodes, Erdos-Renyi graph with 10^4 nodes (average degree is 10). On the right, the graphs given by Citeseer and Pubmed datasets are used. The experiments are run on a single thread on a laptop.

In Fig. 8, the average time needed for sampling a spanning forest on different graphs are plotted while the value of q varies. As shown in the plots, the time to sample a spanning forest via our naive Julia implementation remains comparable with the time to compute the sparse matrix product $\mathsf{L}\mathbf{y}$ through various graphs. Our implementation can be improved in several ways by going beyond the scope of this work. Nevertheless, the empirical results indicate that the times to sample a spanning forest and the product $\mathsf{L}\mathbf{y}$ are within the same order of magnitude.

VI. Conclusion

The Monte Carlo estimators proposed have a comparable computational cost with state-of-the-art methods, can be used as building blocks in various problems, and are amenable to theoretical analysis. As we have shown, the proposed methods are adaptable to various graph-based optimization algorithms including the generalized SSL framework, label propagation, Newton's method and IRLS. Moreover, their use can be more general than the problems involving graph Laplacians. In particular, optimization problems with symmetric, diagonally dominant regularisers can be reduced to graph Tikhonov regularisation problems, along the lines of [40]. Future work will continue to leverage the links between RSFs and graph-related algebra to develop efficient estimators of graph-related quantities *e.g.* effective resistances or $\mathrm{tr}(\mathsf{L}^\dagger)$ where $\mathsf{L}^\dagger$ is the Moore-Penrose inverse of L.

Acknowledgment

This work was partly funded by the the French National Research Agency in the framework of the "Investissements d'avenir" program (ANR-15-IDEX-02), the LabEx PERSYVAL (ANR-11-LABX-0025-01), the ANR GraVa (ANR-18-CE40-0005) the ANR GenGP (ANR-16-CE23-0008) the Grenoble Data Institute (ANR-15-IDEX-02) the MIAI@Grenoble Alpes chair "LargeDATA at UGA." (ANR-19-P3IA-0003).

REFERENCES

[1] Y. Li, R. Yu, C. Shahabi, and Y. Liu, "Diffusion convolutional recurrent neural network: Data-driven traffic forecasting," *6th Int. Conf. Learn. Represent. ICLR 2018 - Conf. Track Proc.*, jul 2018.

[2] W. Huang, T. A. Bolton, J. D. Medaglia, D. S. Bassett, A. Ribeiro, and D. Van De Ville, "A Graph Signal Processing Perspective on Functional Brain Imaging," *Proc. IEEE*, vol. 106, no. 5, pp. 868–885, may 2018.

[3] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst, "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains," *IEEE Signal Process. Mag.*, vol. 30, no. 3, pp. 83–98, oct 2013.

[4] A. Sandryhaila and J. M. Moura, "Big data analysis with signal processing on graphs: Representation and processing of massive data sets with irregular structure," *IEEE Signal Process. Mag.*, vol. 31, no. 5, pp. 80–90, 2014.

[5] M. Belkin, P. Niyogi, and V. Sindhwani, "Manifold regularization: A geometric framework for learning from labeled and unlabeled examples," *J. Mach. Learn. Res.*, vol. 7, pp. 2399–2434, 2006.

[6] I. Pesenson, "Variational Splines and Paley-Wiener Spaces on Combinatorial Graphs," *Constr. Approx.*, vol. 29, no. 1, pp. 1–21, nov 2009.

[7] L. Grady and E. Schwartz, "Anisotropic interpolation on graphs : The combinatorial dirichlet problem Anisotropic Interpolation on Graphs : The Combinatorial Dirichlet Problem," *Bost. Univ.*, 2003.

[8] X. Zhu and Z. Ghahramani, "Learning from labeled and unlabeled data with label propagation," 2002.

[9] L. J. Grady and E. L. Schwartz, *Anisotropic interpolation on graphs: The combinatorial Dirichlet problem*. Citeseer, 2003.

[10] D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Schölkopf, "Learning with local and global consistency," in *Adv. Neural Inf. Process. Syst.*, 2004, pp. 321–328.

[11] K. Avrachenkov, A. Mishenin, P. Gonçalves, and M. Sokol, "Generalized optimization framework for graph-based semi-supervised learning," in *Proc. 2012 SIAM Int. Conf. Data Min.* SIAM, 2012, pp. 966–974.

[12] S. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.

[13] C. S. Burrus, J. A. Barreto, and I. W. Selesnick, "Iterative Reweighted Least-Squares Design of FIR Filters," *IEEE Trans. Signal Process.*, vol. 42, no. 11, pp. 2926–2936, 1994.

[14] Y. Saad, *Iterative Methods for Sparse Linear Systems*, 2003.

[15] D. I. Shuman, P. Vandergheynst, and P. Frossard, "Chebyshev Polynomial Approximation for Distributed Signal Processing," may 2011.

[16] F. R. K. Chung and F. C. Graham, *Spectral graph theory*. American Mathematical Soc., 1997, no. 92.

[17] G. Grimmett, *Probability on graphs: Random processes on graphs and lattices: Second edition*. Cambridge University Press, 2018, vol. 8.

[18] X. M. Wu, Z. Li, A. M. C. So, J. Wright, and S. F. Chang, "Learning with partially absorbing random walks," in *Adv. Neural Inf. Process. Syst.*, vol. 4, 2012, pp. 3077–3085.

[19] X. Wu, "Learning on Graphs with Partially Absorbing Random Walks: Theory and Practice," Ph.D. dissertation, 2016.

[20] K. Avrachenkov, P. Chebotarev, and A. Mishenin, "Semi-supervised learning with regularized Laplacian," *Optim. Methods Softw.*, vol. 32, no. 2, pp. 222–236, 2017.

[21] L. Avena and A. Gaudillière, "Random spanning forests, Markov matrix spectra and well distributed points," oct 2013.

[22] L. Avena and A. Gaudillière, "Two applications of random spanning forests," *Journal of Theoretical Probability*, vol. 31, no. 4, pp. 1975–2004, 2018.

[23] D. B. Wilson, "Generating random spanning trees more quickly than the cover time," in *Proc. Annu. ACM Symp. Theory Comput.*, vol. Part F1294. Association for Computing Machinery, jul 1996, pp. 296–303.

[24] Y. Y. Pilavci, P. O. Amblard, S. Barthelmé, and N. Tremblay, "Smoothing graph signals via random spanning forests," in *ICASSP, IEEE Int. Conf. Acoust. Speech Signal Process. - Proc.*, vol. 2020-May. Institute of Electrical and Electronics Engineers Inc., may 2020, pp. 5630–5634.

[25] P. Marchal, "Loop-erased random walks, spanning trees and hamiltonian cycles," *Elect. Comm. in Probab*, vol. 5, pp. 39–50, 2000.

[26] A. Kulesza and B. Taskar, "Determinantal point processes for machine learning," *Found. Trends Mach. Learn.*, vol. 5, no. 2-3, pp. 123–286, jul 2012.

[27] D. Blackwell, "Conditional Expectation and Unbiased Sequential Estimation," *Ann. Math. Stat.*, vol. 18, no. 1, pp. 105–110, mar 1947.

[28] C. R. Rao, "Information and the Accuracy Attainable in the Estimation of Statistical Parameters," 1992, pp. 235–247.

[29] T. Hastie, R. Tibshirani, J. Friedman, and J. Franklin, "The elements of statistical learning: data mining, inference and prediction," *Math. Intell.*, vol. 27, no. 2, pp. 83–85, 2005.

[30] D. Girard, "Un algorithme simple et rapide pour la validation croisée généralisée sur des problèmes de grande taille," Tech. Rep., 1987.

[31] M. F. Hutchinson, "A stochastic estimator of the trace of the influence matrix for laplacian smoothing splines," *Commun. Stat. - Simul. Comput.*, vol. 19, no. 2, pp. 433–450, 1990.

[32] S. Barthelme, N. Tremblay, A. Gaudilliere, L. Avena, and P.-O. Amblard, "Estimating the inverse trace using random forests on graphs," in *XXVIIème Colloq. GRETSI (GRETSI 2019)*, Lille, France, aug 2019.

[33] R. Tibshirani and L. Wasserman, "Stein's unbiased risk estimate," *Course notes from "Statistical Mach. Learn.*, pp. 1–12, 2015.

[34] C. M. Stein, "Estimation of the Mean of a Multivariate Normal Distribution," *Ann. Stat.*, vol. 9, no. 6, pp. 1135–1151, nov 1981.

[35] X. Zhu, "Semi-Supervised Learning with Graphs," Ph.D. dissertation, 2005.

[36] M. Lebrun, M. Colom, A. Buades, and J. M. Morel, "Secrets of image denoising cuisine," *Acta Numer.*, vol. 21, pp. 475–576, may 2012.

[37] R. J. Tibshirani, J. Taylor *et al.*, "The solution path of the generalized lasso," *The Annals of Statistics*, vol. 39, no. 3, pp. 1335–1371, 2011.

[38] J. Sharpnack, A. Singh, and A. Rinaldo, "Sparsistency of the edge lasso over graphs," in *Artificial Intelligence and Statistics*, 2012, pp. 1028–1036.

[39] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Galligher, and T. Eliassi-Rad, "Collective classification in network data," *AI Mag.*, vol. 29, no. 3, p. 93, 2008.

[40] J. A. Kelner, L. Orecchia, A. Sidford, and Z. A. Zhu, "A simple, combinatorial algorithm for solving SDD systems in nearly-linear time," in *Proc. forty-fifth Annu. ACM Symp. Theory Comput.*, 2013, pp. 911–920.