Efficient Graph Learning From Noisy and Incomplete Data

Peter Berger, Gabor Hannak, and Gerald Matz, *Senior Member, IEEE*

***Abstract*—We consider the problem of learning a graph from a given set of smooth graph signals. Our graph learning approach is formulated as a constrained quadratic program in the edge weights. We provide an implicit characterization of the optimal solution and propose a tailored ADMM algorithm to solve this problem efficiently. Several nearest neighbor and smoothness based graph learning methods are shown to be special cases of our approach. Specifically, our algorithm yields an efficient but extremely accurate approximation to b-matched graphs. We then propose a generalization of our scheme that can deal with noisy and incomplete data via joint graph learning and signal inpainting. We compare the performance of our approach with state-of-the art methods on synthetic data and on real-world data from the Austrian National Council.**

***Index Terms*—Graph signal processing, topology identification, nearest neighbor graphs, signal inpainting, ADMM.**

I. Introduction

A. Motivation and Related Work

GRAPH signal models are a powerful tool to deal with huge data sets since they are flexible, intuitive, efficient, and scalable [2]–[4]. We consider the problem of learning a graph (i.e., its edge topology) from—possibly noisy and incomplete—training data (graph signals). Finding the most suitable graph model for a particular application is key for numerous graph signal processing algorithms to achieve satisfactory performance (e.g., [5]–[10]).

Many graph learning schemes use kernels to quantify similarity of data points, followed by graph sparsification and, possibly, by edge reweighting. Among the most frequently used constructions here are nearest-neighbor graphs such as kNN [9], b-matched graphs [6], and fixed-radius nearest neighbor graphs [11].

Recently, a lot of attention was paid to graph learning with graph signal smoothness constraints [12]–[15]. In [16], [17] the graph topology is learned from stationary graph signals (a special case of which are graph diffusion signals) and [18] imposes that the available graph signals be band-limited on the learned graph. The graph constructed in [7] expresses each data point as a linear combination of the data from neighbor nodes. In [19], the graph topology is learned by minimizing a Laplacian quadratic objective function and lower bounds on the node degrees. A connectedness constraint on the learned graph is enforced in [20].

A different approach interprets the graph Laplacian as the inverse covariance matrix of a multivariate normal distribution and adopts sparse inverse covariance estimation for learning Laplacians [21]–[24].

B. Contributions

We next summarize the key contributions of our paper:

- We provide a novel formulation of the graph learning problem as a constrained quadratic optimization problem in the graph's edge weights. Our formulation can manage complexity by incorporating prior information about admissible edges. Contrary to existing work, we can explicitly control the weighted node degrees and the maximal value of the edge weights, which is advantageous for irregular graphs. We are the first to investigate total variation as a smoothness metric for graph learning, but our method also works with the Laplacian form (used in [12]–[15]).
- To solve the optimization problem underlying our graph learning approach we propose to use ADMM. By exploiting the problem structure, we derive closed-form solutions for the ADMM iterations such that the per-iteration complexity of the resulting algorithm scales linearly with the number of admissible edges, which is crucial for large-scale problems.
- We provide an implicit closed-form analytic characterization of the optimal edge weights learned with our approach. This characterization resembles the waterfilling principle from communications and gives insights as to how to choose the learning parameters in particular problem instances. We further show how existing graph learning algorithms are re-obtained as special cases of our framework, thereby underpinning its generality.
- To deal with the practically important case of noisy and incomplete data, we extend our scheme by formulating a (non-convex) optimization problem for joint graph learning and graph signal inpainting. We solve this problem via an algorithm that alternates between two ADMM iterations, both of which scale linearly with the number of

Manuscript received April 2, 2019; revised September 24, 2019; accepted December 4, 2019. Date of publication January 6, 2020; date of current version January 27, 2020. This work was supported by WWTF under Grant ICT15-119. This article was presented in part at the IEEE International Conference on Acoustics, Speech and Signal Processing, Calgary, AB, Canada, April 2018 [1]. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Gonzalo Mateos. *(Corresponding author: Gerald Matz.)*

P. Berger is with the MED-EL, 6020 Innsbruck, Austria (e-mail: peter.berger@nt.tuwien.ac.at).

G. Hannak is with the Nokia Bell Labs, 1092 Budapest, Hungary (e-mail: ghannak@gmail.com).

G. Matz is with the Institute of Telecommunications, Vienna University of Technology, 1040 Vienna, Austria (e-mail: gerald.matz@tuwien.ac.at).

Digital Object Identifier 10.1109/TSIPN.2020.2964249

admissible graph edges. We further show how to initialize the discrepancy matrix required in the learning step when data is missing.

- We describe numerical experiments on synthetic data and on a real-world data set comprising votes from the Austrian Parliament. The results of these experiments confirm that our learning method is superior to state-of-the-art methods with regard to several performance indicators.

C. Paper Organization

Section II summarizes the required background on graph learning, providing a link between nearest neighbor graphs and graph learning from smooth signals. In Section III-A we formulate the optimization problem for graph learning algorithm and we provide an implicit analytical characterization of the solution that lends itself to an intuitive interpretation. An ADMM algorithm for solving the optimization problem is derived in Section III-B. In Section III-C we show how to extend our framework for learning symmetric weight matrices. In Section IV we discuss the connections of our method to existing graph learning approaches, several of which can be obtained as special cases. Section V presents a method for joint graph learning and inpainting, suitable for the case of noisy and incomplete data. Numerical experiments are described in Section VI and conclusions are provided in Section VII.

II. Preliminaries

Our goal is to learn the topology of a graph that characterizes the relations between N objects (labeled $1,\ldots,N$). For each object $i\in\{1,\ldots,N\}$, we are given an M-dimensional data point (measurement, observation) $\mathbf{x}_i=(X_{i1}\ldots X_{iM})^{\mathrm{T}}\in\mathbb{R}^M$ (superscript $^{\mathrm{T}}$ denotes transposition). The case of noisy and incomplete data will be treated later in Section V. The vertex set of the graph $\mathcal{G}$ to be constructed is given by $\mathcal{V}=\{1,\ldots,N\}$ and we want to use the data $\mathbf{x}_1,\ldots,\mathbf{x}_N$ to determine the edge set $\mathcal{E}\subset\mathcal{V}\times\mathcal{V}$ or, more generally, the weighted adjacency matrix $\mathbf{W}$ of the graph (an edge between nodes i and j exists if and only if $W_{ij}>0$ and W_{ij} characterizes the strength of the connection). For convenience, we arrange the data points into a data matrix $\mathbf{X}=(\mathbf{x}_1,\ldots,\mathbf{x}_N)^{\mathrm{T}}\in\mathbb{R}^{N\times M}$. The columns of this matrix can be interpreted as M graph signals $\tilde{\mathbf{x}}_m=(X_{1\mathrm{m}}\ldots X_{Nm})^{\mathrm{T}}$, $m=1,\ldots,M$.

The general idea of most graph learning methods is to place an edge between two nodes i and j whenever the associated measurements $\mathbf{x}_i$ and $\mathbf{x}_j$ are similar in a certain sense. Two metrics used to quantify similarity are the Manhattan (taxicab, city block) distance (based on the ℓ_1-norm),

$$D_{ij}\triangleq\|\mathbf{x}_i-\mathbf{x}_j\|_1=\sum_{m=1}^{M}|X_{im}-X_{jm}|,\tag{1}$$

and the squared Euclidean distance (based on the ℓ_2-norm),

$$D_{ij}\triangleq\|\mathbf{x}_i-\mathbf{x}_j\|_2^2=\sum_{m=1}^{M}(X_{im}-X_{jm})^2.\tag{2}$$

Among the most popular approaches are nearest neighbor (NN) learning schemes, where nodes that are close to each other (in the sense of having small distance D_{ij}) are connected by an edge. With the radius nearest neighbor approach (see [11]), node i is connected to all nodes within a given distance r, i.e., $(i,j)\in\mathcal{E}$ whenever $D_{ij}\le r$. This construction is very simple but the resulting graph may be disconnected and have isolated nodes. With the kNN scheme, edges are placed between a node and the k nodes that are closest to it (cf. [9]); this guarantees that each node has at least k neighbors.

More recent approaches proposed in the context of graph signal processing impose that the observed graph signals $\tilde{\mathbf{x}}_m$, $m=1,\ldots,M$, are smooth with respect to the graph topology [12]–[15]. Two common metrics for signal smoothness are the (anisotropic) graph total variation (also termed nonlocal total variation in image processing)

$$\|\tilde{\mathbf{x}}_m\|_{\mathrm{TV}}\triangleq\sum_{i=1}^{N}\sum_{j=1}^{N}W_{ij}\,|X_{im}-X_{jm}|\tag{3}$$

and the quadratic form

$$2\tilde{\mathbf{x}}_m^{\mathrm{T}}\mathbf{L}\tilde{\mathbf{x}}_m=\sum_{i=1}^{N}\sum_{j=1}^{N}W_{ij}(X_{im}-X_{jm})^2.\tag{4}$$

Here $\mathbf{L}\triangleq\mathrm{Diag}(\mathbf{W1})-\mathbf{W}$ is the Laplacian matrix associated with the graph ($\mathbf{1}$ is the all-ones vector). The average smoothness ($\sum_{m=1}^{M}\|\tilde{\mathbf{x}}_m\|_{\mathrm{TV}}$ or $2\sum_{m=1}^{M}\tilde{\mathbf{x}}_m^{\mathrm{T}}\mathbf{L}\tilde{\mathbf{x}}_m$) of all given graph signals can be written as

$$\mathrm{tr}\{\mathbf{W}\mathbf{D}^{\mathrm{T}}\}=\sum_{i=1}^{N}\sum_{j=1}^{N}W_{ij}D_{ij},\tag{5}$$

where $\mathrm{tr}\{\cdot\}$ denotes the matrix trace and the elements D_{ij} of the discrepancy matrix $\mathbf{D}$ are given by (1) for the case of total variation and by (2) for the case of the Laplacian quadratic form (we emphasize that the graph learning algorithms in [12]–[15] all use the latter).

III. Graph Learning

The general idea is to construct a sparse weight matrix $\mathbf{W}$ that renders (5) small. With this approach, nodes i and j are connected only if their discrepancy D_{ij} is small whereas large D_{ij} promotes $W_{ij}=0$. Therefore, graph learning under Laplacian smoothness (4) or total variation smoothness (3) constraints is closely related to nearest neighbor graph constructions with Euclidean or Manhattan distance. This and other connections will be explored further in Section IV, revealing that the various existing graph learning schemes differ mainly in how penalty terms and constraints are used to control the graph's sparsity and structure.

A. Proposed Approach

1) The Optimization Problem: Our approach to graph learning aims for determining the graph's edge weight matrix $\mathbf{W}$. Approaches that attempt to learn the graph's Laplacian have been observed to be inherently more difficult to handle (cf. [12]). We next state the convex (quadratic) optimization problem that we propose, assuming we are given a discrepancy matrix $\mathbf{D}$ computed from a data matrix $\mathbf{X}$ (note that this problem imposes no symmetry constraint on the weight matrix $\mathbf{W}$).

$$\min_{\mathbf{W}} \ \operatorname{tr}\{\mathbf{W}\mathbf{D}^{\mathrm{T}}\} + \frac{\mu}{2}\|\mathbf{W}\|_{\mathrm{F}}^2, \tag{6a}$$

$$\text{s.t. } \mathbf{0} \preceq \mathbf{W} \preceq \mathbf{C}, \tag{6b}$$

$$\mathbf{a} \preceq \mathbf{W}\mathbf{1} \preceq \mathbf{b}, \tag{6c}$$

$$\mathbf{1}^{\mathrm{T}}\mathbf{W}\mathbf{1} = 2N. \tag{6d}$$

Here, $\|\mathbf{W}\|_{\mathrm{F}}^2 = \operatorname{tr}\{\mathbf{W}\mathbf{W}^{\mathrm{T}}\}$ is the (squared) Frobenius norm, the inequalities $\preceq$ are to be understood in an element-wise sense, and $\mathbf{a}, \mathbf{b} \in \mathbb{R}^N$, $\mathbf{C} \in \mathbb{R}^{N \times N}$, and $\mu > 0$ are tuning parameters. Let us briefly discuss the various parts of the quadratic program (6). The objective (6a) is a regularized version of the smoothness metric (5), with the parameter μ controlling the amount of connectivity in the graph (a larger μ entails more edges, cf. [12]–[15]). The inequality constraints (6b) on the one hand ensure non-negative edge weights and on the other hand upper bound the weights by the entries of the matrix $\mathbf{C}$. The upper bounds C_{ij} help to avoid excessive imbalances in the weight values and are instrumental for reobtaining known nearest neighbor approaches (cf. Section IV). Choosing $C_{ij} = 0$ for a certain node pair (i, j) entails $W_{ij} = 0$ and hence enforces that $(i, j) \notin \mathcal{E}$. In particular, we henceforth impose $C_{ii} = 0$, which precludes self-loops in the graph. The zero entries of the matrix $\mathbf{C}$ thus amount to a support constraint on the edges that is useful to limit the dimensionality of the problem. The constraints (6c) can be used to independently and flexibly control the (weighted) node degrees via the vectors $\mathbf{a}$ and $\mathbf{b}$, which is an advantage over existing methods that either explicitly aim for regular graphs (e.g., [6], [9]) or use the same degree penalty parameter for all nodes (e.g., [12]–[14]). Finally, (6d) is a simple normalization constraint. The actual choice of the normalization constant is immaterial in practice since replacing $2N$ in (6d) by $2Nc$ is equivalent to solving (6) with the replacements $\mathbf{a} \to \mathbf{a}/c$, $\mathbf{b} \to \mathbf{b}/c$, $\mathbf{C} \to \mathbf{C}/c$, $\mu \to c\mu$, and multiplying the resulting weights by c. Furthermore, due to the normalization constraint (6d), an affine transformation $c_0\mathbf{D} + c_1$ of the discrepancy matrix is equivalent to solving (6) with the replacement $\mu \to \mu/c_0$. We henceforth assume $\mathbf{0} \preceq \mathbf{a} \preceq \mathbf{b}$, $\mathbf{1}^{\mathrm{T}}\mathbf{a} \leq 2N \leq \mathbf{1}^{\mathrm{T}}\mathbf{b}$, $\mathbf{C} \succeq \mathbf{0}$, $\mathbf{C}\mathbf{1} \succeq \mathbf{a}$, and $\mathbf{1}^{\mathrm{T}}\mathbf{C}\mathbf{1} \geq 2N$; all of these conditions are necessary for (6) to be feasible. Note that the degree and weight constraints become inactive whenever $a_i = 0$, $b_i \geq 2N$, or $C_{ij} \geq 2N$, respectively.

2) Characterization and Interpretation of Optimal Solution: We next characterize the solution of (6). To obtain implicit equations for the solution of (6), we use the element-wise clipping function $[\mathbf{x}]_+^c \triangleq \max\{\mathbf{0}, \min\{\mathbf{x}, c\mathbf{1}\}\}$. We use the conventions $[\mathbf{x}]_+ \triangleq [\mathbf{x}]_+^\infty$ and $[\mathbf{x}]_- \triangleq -[-\mathbf{x}]_+$. The proof of the following theorem is given in Appendix A.

Theorem 1: Let $s_i = \frac{1}{\mu}\sum_{j=1}^N \big[r - D_{ij}\big]_+^{\mu C_{ij}}$. The optimal edge weights W_{ij} for (6) are then characterized by

$$W_{ij} = \begin{cases} \frac{1}{\mu}\big[r - D_{ij} + \alpha_i\big]_+^{\mu C_{ij}}, & s_i < a_i, \\ \frac{1}{\mu}\big[r - D_{ij}\big]_+^{\mu C_{ij}}, & a_i \leq s_i \leq b_i, \\ \frac{1}{\mu}\big[r - D_{ij} - \beta_i\big]_+^{\mu C_{ij}}, & s_i > b_i, \end{cases} \tag{7}$$

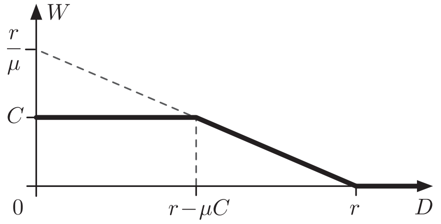


Fig. 1. Illustration of the baseline solution $W = f(D)$ for the optimal edge weights W. Note that for small enough μ, $f(D)$ effectively becomes a step function.

with α_i, β_i, and r determined by the equations

$$\frac{1}{\mu}\sum_{j=1}^N \big[r - D_{ij} + \alpha_i\big]_+^{\mu C_{ij}} = a_i,$$

$$\frac{1}{\mu}\sum_{j=1}^N \big[r - D_{ij} - \beta_i\big]_+^{\mu C_{ij}} = b_i,$$

$$\sum_{i=1}^N\sum_{j=1}^N W_{ij} = 2\,N.$$

Theorem 1 provides an intuitive interpretation for the optimal edge weights and provides a basis for comparison with other learning schemes (see Section IV). The baseline solution $W = f(D) \triangleq \frac{1}{\mu}\big[r - D\big]_+^{\mu C}$ amounts to a (clipped) water-filling solution for the weights (see Figure 1); here, r plays the role of the water level that has to be chosen to meet the normalization constraint (6d). The optimal water level is determined by the normalization parameter μ. The baseline solution can also be thought of as a generalization of the fixed-radius nearest neighbor graph that admits arbitrary edge weights, with r serving as the neighborhood radius. If the baseline solution is not consistent with the degree constraints a_i and b_i, the waterlevel is adjusted correspondingly by either adding α_i to r (thus increasing the weights and thereby the degree) or by subtracting β_i from r (thus decreasing the weights and the associated degree).

When there are no constraints on the weighted degrees and the weights themselves, the optimal edge weights can be explicitly computed via the expression

$$W_{ij} = \frac{1}{\mu}\big[r - D_{ij}\big]_+,$$

with the radius/water level chosen such that

$$\omega(r) \triangleq \sum_{i=1}^N\sum_{j=1}^N \big[r - D_{ij}\big]_+ = 2N\mu.$$

The function $\omega(r)$ is piecewise linear (with bends at $r = D_{ij}$), continuous, and monotonically increasing. The optimal water level can thus be found as $r = \omega^{-1}(2N\mu)$. Choosing r between the lth smallest and the $(l+1)$th smallest discrepancies yields a graph with exactly l edges (see [1] and Figure 2).

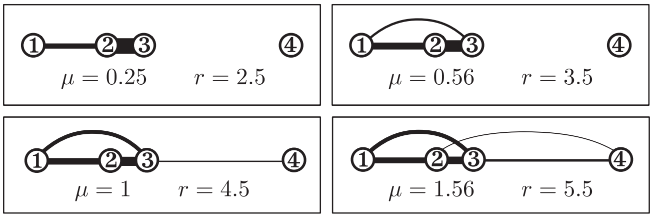


Fig. 2. Learning a four-node graph with discrepancies $D_{12} = 2$, $D_{13} = 3$, $D_{14} = 7$, $D_{23} = 1$, $D_{24} = 5$, $D_{34} = 4$ and various μ (edge thickness is proportional to weight, node distance is proportional to discrepancy).

B. Learning Algorithm

1) Problem Reformulation: We next derive an efficient ADMM algorithm [25] for finding the optimal edge weights in the general case. We first reduce the problem dimension be exploiting the fact that $W_{ij} = 0$ whenever $C_{ij} = 0$. Incorporating such prior knowledge on the support of the edge set is key for handling massive data sets [21]. A simple way to obtain such prior information is, e.g., by constructing an initial approximate nearest neighbor graph [14].

Define the set of indices for the potentially non-zero weights incident at node i as $\mathcal{C}_i \triangleq \{j : C_{ij} > 0\}$ and its cardinality $k_i \triangleq |\mathcal{C}_i|$. Furthermore, let the set of indices for admissible edges be defined as $\mathcal{C} \triangleq \{(i,j) : C_{ij} > 0\}$. The total number of weights to optimize is then $K \triangleq |\mathcal{C}| = \sum_{i=1}^{N} k_i \leq N^2$. We stack the rows of the weight matrix $\mathbf{W}$ into a column vector and drop all elements for which $C_{ij} = 0$. This results in a K-dimensional vector $\mathbf{w} = (\mathbf{w}_1^{\mathrm{T}}, \ldots, \mathbf{w}_N^{\mathrm{T}})^{\mathrm{T}}$ where each subvector $\mathbf{w}_i$ has length k_i. The same re-arrangement of the discrepancy matrix $\mathbf{D}$ and of the weight-bound matrix $\mathbf{C}$ yields K-dimensional vectors $\mathbf{d}$ and $\mathbf{c}$, respectively. To reformulate and combine the affine inequality constraints we define ($\mathbf{I}$ denotes the identity matrix)

$$\mathbf{H} \triangleq \Big(-\mathbf{I} \quad \mathbf{I} \quad -\mathbf{B}^{\mathrm{T}} \quad \mathbf{B}^{\mathrm{T}}\Big)^{\mathrm{T}} \in \{0,1\}^{(2K+2N)\times K},$$

$$\mathbf{h} \triangleq \Big(\mathbf{0}^{\mathrm{T}} \quad \mathbf{c}^{\mathrm{T}} \quad -\mathbf{a}^{\mathrm{T}} \quad \mathbf{b}^{\mathrm{T}}\Big)^{\mathrm{T}} \in \mathbb{R}^{(2N+2\,K)}.$$

Here, the matrix $\mathbf{B}$ reads

$$\mathbf{B} \triangleq (\mathbf{b}_1, \ldots, \mathbf{b}_N)^{\mathrm{T}} \in \{0,1\}^{N\times K}, \tag{8}$$

with

$$\mathbf{b}_i \triangleq (\underbrace{0, 0, \ldots, 0}_{\sum_{j=1}^{i-1} k_j}, \underbrace{1, 1, \ldots, 1}_{k_i}, \underbrace{0, 0, \ldots, 0}_{\sum_{j=i+1}^{N} k_j})^{\mathrm{T}}.$$

With the above definitions, our graph learning problem (6) can be rewritten in standard form as

$$\min_{\mathbf{w}} \ \mathbf{d}^{\mathrm{T}}\mathbf{w} + \frac{\mu}{2}\mathbf{w}^{\mathrm{T}}\mathbf{w}, \tag{9a}$$

$$\text{s.t. } \mathbf{H}\mathbf{w} \preceq \mathbf{h}, \tag{9b}$$

$$\mathbf{1}^{\mathrm{T}}\mathbf{w} = 2N. \tag{9c}$$

2) ADMM Updates: The quadratic program (9) can be solved using standard tools like interior-point methods or ADMM. We advocate ADMM since it turns out to allow for closed-form solutions to the intermediate steps. The general ADMM update equations for the linearly constrained quadratic program (9) read (cf. [25])

$$\begin{aligned}
\mathbf{s}^{(\ell+1)} &= \mathbf{d} + \rho\mathbf{H}^{\mathrm{T}}\big(\mathbf{z}^{(\ell)} + \mathbf{u}^{(\ell)} - \mathbf{h}\big) + \rho\mathbf{1}\big(\upsilon^{(\ell)} - 2N\big),\\
\mathbf{w}^{(\ell+1)} &= -(\mu\mathbf{I} + \rho\mathbf{H}^{\mathrm{T}}\mathbf{H} + \rho\mathbf{J})^{-1}\,\mathbf{s}^{(\ell+1)},\\
\mathbf{z}^{(\ell+1)} &= \Big[\mathbf{h} - \mathbf{H}\mathbf{w}^{(\ell+1)} - \mathbf{u}^{(\ell)}\Big]_{+}\\
\mathbf{u}^{(\ell+1)} &= \mathbf{u}^{(\ell)} + \mathbf{H}\mathbf{w}^{(\ell+1)} - \mathbf{h} + \mathbf{z}^{(\ell+1)},\\
\upsilon^{(\ell+1)} &= \upsilon^{(\ell)} + \mathbf{1}^{\mathrm{T}}\mathbf{w}^{(\ell+1)} - 2N.
\end{aligned} \tag{10}$$

Here, $\mathbf{J} = \mathbf{1}\mathbf{1}^{\mathrm{T}}$ is the all-ones matrix. These ADMM updates are guaranteed to converge to an optimal point for any $\rho > 0$ [25]. All steps except (10) are simple additions (all entries of $\mathbf{H}$ are either 0 or 1). We next show that (10) amounts to a simple additive update, too. We use the partition $\mathbf{w} = (\mathbf{w}_1^{\mathrm{T}}, \ldots, \mathbf{w}_N^{\mathrm{T}})^{\mathrm{T}}$ where each subvector $\mathbf{w}_i$ has length k_i, and similarly for $\mathbf{s}$. Furthermore, we define $\eta \triangleq \mu + 2\rho$, $\gamma_i \triangleq 1/(\eta + 2\rho k_i)$, and $\bar{\gamma} \triangleq \sum_{i=1}^{N} \gamma_i k_i$. The proof of Lemma 2 is given in Appendix B.

Lemma 2: The update (10) is equivalent to

$$\mathbf{w}_i^{(\ell+1)} = -\frac{1}{\eta}\mathbf{s}_i^{(\ell+1)} + \left(\frac{2\rho\gamma_i\delta_i^{(\ell+1)}}{\eta} + \frac{\rho\gamma_i\bar{\delta}^{(\ell+1)}}{1+\rho\bar{\gamma}}\right)\mathbf{1}_{k_i},$$

where $\delta_i^{(\ell+1)} \triangleq \mathbf{1}_{k_i}^{\mathrm{T}}\mathbf{s}_i^{(\ell+1)}$ and $\bar{\delta}^{(\ell+1)} \triangleq \sum_{i=1}^{N} \gamma_i \delta_i^{(\ell+1)}$.

Using Lemma 2, all ADMM steps decouple into simple scalar operations. We next reformulate the ADMM updates. We exploit the fact that with $\mathbf{m} \triangleq \mathbf{h} - \mathbf{H}\mathbf{w} - \mathbf{u}$ we have $\mathbf{z} = [\mathbf{m}]_+$, $\mathbf{u} = -\mathbf{m} + \mathbf{z} = -[\mathbf{m}]_-$ and hence $\mathbf{z} + \mathbf{u} = [\mathbf{m}]_+ - [\mathbf{m}]_- = |\mathbf{m}|$. We thus perform the iterations on the elements of the vector $\mathbf{m} = (\mathbf{q}^{\mathrm{T}}, \bar{\mathbf{q}}^{\mathrm{T}}, \mathbf{y}^{\mathrm{T}}, \bar{\mathbf{y}}^{\mathrm{T}})^{\mathrm{T}}$. Writing out the iterations in an element by element fashion results in Algorithm 1. In this algorithm, it is implicitly understood that the updates of double-index quantities is performed for all $(i,j) \in \mathcal{C}$ and the updates of single-index quantities is performed for $i = 1, \ldots, N$. We note that the convergence of Algorithm 1 can be further accelerated along the lines of [26]. Furthermore, for special problem instances some of the constraints become inactive and the corresponding updates can be simplified.

3) Stopping Criterion and Complexity: To halt the ADMM iterations, we use the stopping criterion based on the norms of the primal and dual residual (see [25, Sec. 3.3.1]), which in our problem are given by

$$r^{(\ell+1)} = \sqrt{\big\|\mathbf{H}\mathbf{w}^{(\ell+1)} + \mathbf{z}^{(\ell+1)} - \mathbf{h}\big\|_2^2 + (\mathbf{1}^{\mathrm{T}}\mathbf{w}^{(\ell+1)} - 2N)^2},$$

$$R^{(\ell+1)} = \rho\big\|\mathbf{H}^{\mathrm{T}}(\mathbf{z}^{(\ell+1)} - \mathbf{z}^{(\ell)})\big\|_2.$$

Let us further define

$$\Delta^{(\ell)} \triangleq \sqrt{\max\{\|\mathbf{H}\mathbf{w}^{(\ell)}\|_2^2 + (\mathbf{1}^{\mathrm{T}}\mathbf{w}^{(\ell)})^2, \|\mathbf{h}\|_2^2 + 4\,N^2\}}.$$

Algorithm 1 will be stopped as soon as the following two conditions are simultaneously satisfied:

$$r^{(\ell)} \leq \epsilon_{\mathrm{abs}}\sqrt{2(K+N)+1} + \epsilon_{\mathrm{rel}}\max\{\|\mathbf{z}^{(\ell)}\|_2, \Delta^{(\ell)}\}, \tag{11}$$

$$R^{(\ell)} \leq \epsilon_{\mathrm{abs}}\sqrt{K} + \epsilon_{\mathrm{rel}}\,\rho\|\mathbf{H}^{\mathrm{T}}\mathbf{u}^{(\ell)}\|_2. \tag{12}$$

Algorithm 1. — learn( )

```
input: $\mathbf{D}$, $\mu$, $\mathbf{a}$, $\mathbf{b}$, $\mathbf{C}$, $\rho$
$\mathcal{C}_i = \{j : C_{ij} > 0\}$
$k_i = |\mathcal{C}_i|$
$\nu = 1/(\mu + 2\rho)$
$\xi_i = 2\nu\rho/(\mu + 2\rho + 2\rho k_i)$
$\zeta_i = \xi_i / \big(2\nu + 2\nu\rho \sum_{j=1}^{N} \frac{k_j}{\mu+2\rho+2\rho k_j}\big)$
$q_{ij}^{(0)} = \bar{q}_{ij}^{(0)} = 0$
$y_i^{(0)} = \bar{y}_i^{(0)} = 0$
$\upsilon^{(0)} = 0$
$\ell = 0$
repeat
    $\hat{\upsilon}_{ij}^{(\ell)} = |\bar{q}_{ij}^{(\ell)}| - |q_{ij}^{(\ell)}| - C_{ij}$
    $\hat{y}_i^{(\ell)} = |\bar{y}_i^{(\ell)}| - |y_i^{(\ell)}| + a_i - b_i$
    $S_{ij}^{(\ell+1)} = D_{ij} + \rho\big(\hat{\upsilon}_{ij}^{(\ell)} + \hat{y}_i^{(\ell)} - 2N + \upsilon^{(\ell)}\big)$
    $\delta_i^{(\ell+1)} = \sum_{j \in \mathcal{C}_i} S_{ij}^{(\ell+1)}$
    $\bar{\delta}^{(\ell+1)} = \frac{1}{2\nu\rho} \sum_{i=1}^{N} \delta_i^{(\ell+1)} \xi_i$
    $W_{ij}^{(\ell+1)} = -\nu S_{ij}^{(\ell+1)} + \xi_i \delta_i^{(\ell+1)} + \zeta_i \bar{\delta}^{(\ell+1)}$
    $g_i^{(\ell+1)} = \sum_{j \in \mathcal{C}_i} W_{ij}^{(\ell+1)}$
    $q_{ij}^{(\ell+1)} = W_{ij}^{(\ell+1)} + [q_{ij}^{(\ell)}]_-$
    $\bar{q}_{ij}^{(\ell+1)} = C_{ij} - W_{ij}^{(\ell+1)} + [\bar{q}_{ij}^{(\ell)}]_-$
    $y_i^{(\ell+1)} = g_i^{(\ell+1)} - a_i + [y_i^{(\ell)}]_-$
    $\bar{y}_i^{(\ell+1)} = b_i - g_i^{(\ell+1)} + [\bar{y}_i^{(\ell)}]_-$
    $\upsilon^{(\ell+1)} = \upsilon^{(\ell)} + \sum_{i=1}^{N} g_i^{(\ell+1)} - 2N$
    $\ell = \ell + 1$
until stopping criterion is satisfied
output: $W_{ij}^{(\ell-1)}$
```

The constants ϵ_{abs} and ϵ_{rel} are the absolute and relative tolerance, respectively [25]. In our numerical experiments we used $\epsilon_{\mathrm{abs}} = 10^{-10}$ and $\epsilon_{\mathrm{rel}} = 10^{-5}$.

All steps in Algorithm 1 are seen to be additive scalar operations on the set $\mathcal{C}$ or on the sets $\mathcal{C}_i$, $i = 1, \ldots, N$. Since $|\mathcal{C}| = \sum_i |\mathcal{C}_i| = K$, it follows that the overall complexity of each ADMM iteration scales linearly in the number K of admissible edge weights. The worst case is $K = N^2$, which happens when no prior knowledge about the edge support is available. In many problems there is structure, though, that leads to $K \ll N^2$. In the absence of prior knowledge we can enforce and control sparse edge support by using a nearest neighbor graph as a prior (see Section VI-A and [14]).

C. Symmetric Graph Learning

Sometimes symmetric weight matrices are advantageous, e.g., clustering and semi-supervised learning perform better with b-matched graphs than with kNN [6], [10], [27]. However, learning symmetric graphs may be more expensive [28].

We next discuss how to adapt the above framework to learn symmetric weight matrices. In particular we consider the optimization problem (6) augmented with the symmetry constraint $\mathbf{W} = \mathbf{W}^{\mathrm{T}}$. Since $\mathbf{D}$ and $\mathbf{W}$ are both symmetric, we further assume $\mathbf{C} = \mathbf{C}^{\mathrm{T}}$. If there are no constraints on the node degrees (equivalently, $a_i = 0, b_i \geq 2N$), the solution of (6) is symmetric (cf. Theorem 1) and the constraint $\mathbf{W} = \mathbf{W}^{\mathrm{T}}$ is implicitly satisfied. Thus, modifications are necessary only when the contraints on the node degrees are active. Since $\mathbf{D}$, $\mathbf{W}$, and $\mathbf{C}$ are all symmetric, the optimization problem can be reformulated in terms of their (strictly) upper triangular parts. Removing all elements of $\mathbf{D}$, $\mathbf{W}$, and $\mathbf{C}$ for which $C_{ij} = 0$ (and thus $W_{ij} = 0$) results in vectors $\mathbf{d}$, $\mathbf{w}$, $\mathbf{c}$. With these vectors, we reobtain the constrained quadratic optimization problem stated in (9), however, with a different 0/1-valued matrix $\mathbf{B}$ (which is defined via $\mathbf{B}\mathbf{w} = \mathbf{W}\mathbf{1}$). As in the non-symmetric case, this optimization problem can be solved via ADMM. However, due to the different structure of $\mathbf{B}$, the matrix inversion in (10) admits no closed form and hence interior-point methods are suitable alternatives.

IV. Relation to State-of-the-Art

In this section we explain the differences and similarities between our approach and existing methods, some of which can be interpreted as special cases of our method. Our discussion reveals a unifying perspective, i.e., all existing methods in a sense can be viewed as some kind of thresholding procedure on the data discrepancies.

A. Nearest Neighbor Graphs

With radius nearest neighbor, there are edges with identical weights between node pairs $(i, j) \in \mathcal{E}_r$ where $\mathcal{E}_r \triangleq \{(i, j) : D_{ij} < r\}$. Let $K = |\mathcal{E}_r|$, and $D' \triangleq \max_{(i,j) \in \mathcal{E}_r} D_{ij}$. It then follows that by setting $C_{ij} = C \triangleq 2N/K$, $\mu = (r - D')/C$, and by de-activating the degree constraints ($a_i = 0, b_i = 2N$), the baseline solution for our problem (cf. Theorem 1) simplifies to $W_{ij} = C$ for $D_{ij} < r$, thereby recovering the radius nearest neighbor graph.

A kNN graph can be thought of as a radius nearest neighbor graph in which the radius for each node is different in order to accommodate exactly k neighbors. We assume that all D_{ij} are distinct (otherwise the kNN graph is not unique). Denote by $\mathcal{D}_i$ the set containing the k smallest discrepancies associated to node i and let $D'_i \triangleq \max\{\mathcal{D}_i\}$. Furthermore, set $C_{ij} = C \triangleq 2/k$ for $i \neq j$ and $a_i = b_i = 2$, $i = 1, \ldots, N$. With these choices, Theorem 1 implies that $W_{ij} = \frac{1}{\mu}[\theta_i - D_{ij}]_+^{\mu C}$, with θ_i chosen to meet the degree constraint. The kNN solution is recovered with $\mu \leq \min\{D''_i - D'_i\}/C$ where $D''_i = \min\{D_{ij} : D_{ij} \notin \mathcal{D}_i\}$, leading to $\theta_i = D''_i$ and $W_{ij} = C = 2/k$ for $D_{ij} \in \mathcal{D}_i$.

The b-matched graph constructed in [6] is the symmetric version of the kNN graph. One might thus speculate that the symmetric version of our approach (cf. Section III-C) yields a b-matched graph (assuming $a_i = b_i = 2$, $C_{ij} = 2/k$ for $i \neq j$, and $\mu \leq \min\{D''_i - D'_i\}/C$ as with the kNN graph). However, we will see in our numerical experiments (cf. Section VI-A) that this conjecture is only approximately true.

B. Learning via Smoothness

In [15], a sparse unweighted graph is learned by minimizing the Laplacian smoothness metric subject to the constraint $\|\mathbf{w}\|_0 = K$. The solution is a graph with edges between those node pairs that correspond to the K smallest discrepancies; this in fact is a radius nearest neighbor graph with a radius that equals the $(K+1)$th smallest discrepancy and hence amounts to a special case of our method.

The Laplacian learning method in [12] amounts to minimizing the Laplacian smoothness objective using the squared Frobenius norm of the Laplacian as regularization. Since $\frac{\beta}{2}\|\mathbf{L}\|_{\mathrm{F}}^2 = \frac{\beta}{2}\|\mathbf{W}\|_{\mathrm{F}}^2 + \frac{\beta}{2}\|\mathbf{W}\mathbf{1}\|_2^2$, this approach penalizes large node degrees implicitly via the term $\frac{\beta}{2}\|\mathbf{W}\mathbf{1}\|_2^2 = \frac{\beta}{2}\sum_i(\sum_j W_{ij})^2$ (there is no penalty for small node degrees, though). Furthermore, the penalty acts globally and is uniform over all nodes (e.g., a 50 node star graph incurs the same degree penalty as a 7-regular graph of the same size). In [13], [14], the objective function is similar to ours but comprises the additional term $-\alpha\mathbf{1}^{\mathrm{T}}\log(\mathbf{W}\mathbf{1})$, which implicitly penalizes small weighted node degrees (the penalty is global/uniform over all nodes and there is no control for large node degrees, though). Contrary to these two approaches, our scheme prevents too large and too small node degrees explicitly via the constraints $a_i \le \sum_{j=1}^N W_{ij} \le b_i$, $i = 1,\ldots,N$, which allow us to control the degree for each node individually.

We remark that the implicit one-node solution stated in [14, Theorem 2] is structurally similar to that obtained with our method in the absence of upper bounds on the weights and degrees. However, the per-node solutions in [14, Theorem 2] are decoupled whereas in our case the waterlevel r for all edges is determined by the joint normalization constraint (6d).

V. Graph Learning From Noisy, Incomplete Data

A. Problem Formulation and Basic Approach

We next extend our graph learning algorithm to the case of incomplete and noisy data. Assume we are given noisy and subsampled observations of the actual data X_{im},

$$Y_{im} = \begin{cases} X_{im} + E_{im}, & \text{for } i \in \mathcal{S}_m, \\ 0, & \text{for } i \notin \mathcal{S}_m. \end{cases}$$

The error term E_{im} summarizes all modeling and measurement imperfections. Note that for each graph signal $\mathbf{x}_m = (X_{1m},\ldots,X_{Nm})^{\mathrm{T}}$ the sampling set $\mathcal{S}_m \subset \{1,\ldots,N\}$ may (and preferably should) be distinct. Our goal is to learn an unknown graph $\mathbf{W}$ and to reconstruct the unknown data $\mathbf{X}$ such that the data is maximally smooth on the graph while simultaneously being maximally consistent with the measurements. To this end, we propose to determine the weight matrix $\mathbf{W}$ and the graph signals $\mathbf{X} = (\tilde{\mathbf{x}}_1,\ldots,\tilde{\mathbf{x}}_M) = (\mathbf{x}_1,\ldots,\mathbf{x}_N)^{\mathrm{T}}$ via the joint graph learning and inpainting problem

$$\min_{\mathbf{W},\mathbf{X}}\ \mathrm{tr}\{\mathbf{W}\mathbf{D}(\mathbf{X})^{\mathrm{T}}\} + \frac{\mu}{2}\|\mathbf{W}\|_{\mathrm{F}}^2, \tag{13a}$$

$$\text{s.t. } \mathbf{0} \preceq \mathbf{W} \preceq \mathbf{C}, \tag{13b}$$

$$\mathbf{a} \preceq \mathbf{W}\mathbf{1} \preceq \mathbf{b}, \tag{13c}$$

$$\mathbf{1}^{\mathrm{T}}\mathbf{W}\mathbf{1} = 2N, \tag{13d}$$

$$\sum_{i\in\mathcal{S}_m}(X_{im} - Y_{im})^2 \le \varepsilon_m^2, \quad m = 1,\ldots,M. \tag{13e}$$

This is essentially the original graph learning problem (6), augmented with the minimization over the unknown data matrix $\mathbf{X}$ and with the empirical error constraint (13e) (we allow for different accuracies of the various measured graph signals via distinct ε_m). To learn a symmetric graph one can impose the additional constraint $\mathbf{W} = \mathbf{W}^{\mathrm{T}}$ (cf. Section III-C). The dependence of the discrepancy matrix on the (unknown) data is made explicit by writing $\mathbf{D}(\mathbf{X})$. The smoothness term

$$\mathrm{tr}\{\mathbf{W}\mathbf{D}(\mathbf{X})^{\mathrm{T}}\} = \sum_{i=1}^N\sum_{j=1}^N W_{ij}\,\|\mathbf{x}_i - \mathbf{x}_j\|_p^p, \quad p = 1 \text{ or } 2,$$

is linear in $\mathbf{W}$ and convex in $\mathbf{X}$, however, it is not *jointly* convex in $\mathbf{W}$ and $\mathbf{X}$. Thus, the joint graph learning and signal inpainting problem (13) is not convex.

We propose to find a (generally local) minimum of (13) by iteratively optimizing with respect to $\mathbf{W}$ and $\mathbf{X}$ in an alternating manner while keeping the other variable fixed.[1] This yields a sequence of weight matrices $\mathbf{W}^{(q)}$ and graph signals $\mathbf{X}^{(q)}$, where $\mathbf{W}^{(q)}$ is the solution of (6) with $\mathbf{D} = \mathbf{D}(\mathbf{X}^{(q)})$ and $\mathbf{X}^{(q+1)}$ is the solution of

$$\min_{\mathbf{X}}\ \mathrm{tr}\{\mathbf{W}^{(q)}\mathbf{D}(\mathbf{X})^{\mathrm{T}}\} = \sum_{i=1}^N\sum_{j=1}^N W_{ij}^{(q)}\|\mathbf{x}_i - \mathbf{x}_j\|_p^p,$$
$$\text{s.t. } \sum_{i\in\mathcal{S}_m}(X_{im} - Y_{im})^2 \le \varepsilon_m^2, \quad m = 1,\ldots,M.$$

Note that the penalty term and constraints for the graph weights can be omitted in this noisy signal reconstruction (or inpainting) problem. Furthermore, with (2) the smoothness term for the ℓ_2 case decouples as

$$\mathrm{tr}\{\mathbf{W}^{(q)}\mathbf{D}(\mathbf{X})^{\mathrm{T}}\} = 2\sum_{m=1}^M \tilde{\mathbf{x}}_m^{\mathrm{T}}\mathbf{L}^{(q)}\tilde{\mathbf{x}}_m,$$

and, using (1), for the total variation (ℓ_1 case) as

$$\mathrm{tr}\{\mathbf{W}^{(q)}\mathbf{D}(\mathbf{X})^{\mathrm{T}}\} = \sum_{m=1}^M \|\tilde{\mathbf{x}}_m\|_{\mathrm{TV}}$$

(where the weight matrix $\mathbf{W}^{(q)}$ is used in the total variation). Since the empirical error constraint is imposed on the individual graph signals $\tilde{\mathbf{x}}_m$ as well, the graph signal inpainting step can be performed for each graph signal $\tilde{\mathbf{x}}_m$ separately.

The ℓ_2 discrepancy is a quadratic differentiable function of the data; hence, the inpainting problem here is a constrained quadratic program which can be solved via ADMM [25] or interior-point methods [29].

B. Total Variation Inpainting

Inpainting based on total variation has been addressed in our previous work [30], where we devised a primal-dual algorithm [31]–[33] to solve the non-differentiable convex problem (in this subsection we drop the indices m and q and the tilde on

[1] Such an approach has briefly been suggested in the introduction of [14] and was pursued for joint learning and denoising in [1], [12].

the graph signals $\tilde{\mathbf{x}}_m$)

$$\min_{\mathbf{x}} \|\mathbf{x}\|_{\mathrm{TV}} = \sum_{i=1}^{N}\sum_{j=1}^{N} W_{ij}\,|x_i - x_j|$$
$$\text{s.t.} \sum_{i\in\mathcal{S}} (x_i - y_i)^2 \le \varepsilon^2. \tag{14}$$

In this paper, we derive an alternative augmented ADMM algorithm (cf. [34]), which is closely related to the primal-dual algorithm [34, Theorem 2]. This augmented ADMM method is more consistent with the ADMM-based graph learning algorithm and admits for a varying penalty strategy and a practical stopping criterion [25].

To derive the augmented ADMM iterations, we need the graph gradient $\nabla\colon \mathbb{R}^N \to \mathbb{R}^{N\times N}$, defined via

$$(\nabla\mathbf{x})_{ij} \triangleq (x_j - x_i)W_{ij}. \tag{15}$$

We can then rewrite the total variation as

$$\|\mathbf{x}\|_{\mathrm{TV}} = \sum_{i=1}^{N}\sum_{j=1}^{N} W_{ij}\,|x_i - x_j| = \|\nabla\mathbf{x}\|_1.$$

Furthermore, let us denote the constraint set in (14) by

$$\mathcal{Q} \triangleq \left\{\mathbf{x}\in\mathbb{R}^N\colon \sum_{i\in\mathcal{S}}(x_i - y_i)^2 \le \varepsilon^2\right\},$$

and define the characteristic function

$$\chi_{\mathcal{Q}}(\mathbf{x}) \triangleq \begin{cases} 0, & \mathbf{x}\in\mathcal{Q},\\ +\infty, & \mathbf{x}\notin\mathcal{Q}. \end{cases}$$

The optimization problem (14) can now be reformulated as

$$\min_{\mathbf{x},\mathbf{Z}} \|\mathbf{Z}\|_1 + \chi_{\mathcal{Q}}(\mathbf{x}), \quad \text{s.t. } \nabla\mathbf{x} = \mathbf{Z}.$$

This is exactly the form required for the augmented ADMM [34, Eq. (3)]. To formulate the ADMM updates, we need the graph divergence, which is the negative adjoint of the graph gradient, $\operatorname{div} = -\nabla^* : \mathbb{R}^{N\times N} \to \mathbb{R}^N$, defined as [35]

$$(\operatorname{div}\mathbf{Z})_i \triangleq \sum_{j=1}^{N}(W_{ij}Z_{ij} - W_{ji}Z_{ji}). \tag{16}$$

In analogy to [30] we then obtain the ADMM updates

$$\hat{\mathbf{z}}^{(\ell)} = \operatorname{div}\bigl(2\mathbf{Z}^{(\ell)} - \mathbf{Z}^{(\ell-1)}\bigr), \tag{17}$$
$$\mathbf{x}^{(\ell+1)} = \arg\min_{\mathbf{x}\in\mathcal{Q}} \frac{\tau}{2}(\mathbf{x}-\mathbf{x}^{(\ell)})^{\mathrm{T}}\boldsymbol{\Psi}(\mathbf{x}-\mathbf{x}^{(\ell)}) - \mathbf{x}^{\mathrm{T}}\hat{\mathbf{z}}^{(\ell)}, \tag{18}$$
$$\mathbf{Z}^{(\ell+1)} = \pi_{\mathcal{P}}(\mathbf{Z}^{(\ell)} + \tau\nabla\mathbf{x}^{(\ell+1)}). \tag{19}$$

In the last step, $\pi_{\mathcal{P}}(\mathbf{Z})$ denotes the orthogonal projection of $\mathbf{Z}$ onto the convex set

$$\mathcal{P} \triangleq \{\mathbf{P} : |P_{ij}| \le 1,\ 1 \le i,j \le N\}.$$

Furthermore, $\boldsymbol{\Psi}$ in (18) can be any matrix that satisfies $\mathbf{x}^{\mathrm{T}}\boldsymbol{\Psi}\mathbf{x} \ge -\mathbf{x}^{\mathrm{T}}\operatorname{div}\nabla\mathbf{x}$ for all $\mathbf{x}$. Since $\operatorname{div} = -\nabla^*$, we have

$$\begin{aligned} -\mathbf{x}^{\mathrm{T}}\operatorname{div}\nabla\mathbf{x} &= \operatorname{tr}\{\mathbf{x}^{\mathrm{T}}\nabla^*\nabla\mathbf{x}\} = \|\nabla\mathbf{x}\|_{\mathrm{F}}^2 \\ &= \sum_{i=1}^{N}\sum_{j=1}^{N}(x_j - x_i)^2\,W_{ij}^2 \\ &\le \sum_{i=1}^{N}\sum_{j=1}^{N} 2\bigl(x_j^2 + x_i^2\bigr)W_{ij}^2 \\ &= \sum_{i=1}^{N} x_i^2\, 2\sum_{j=1}^{N}\bigl(W_{ij}^2 + W_{ji}^2\bigr) \\ &\le \sum_{i=1}^{N} x_i^2 \max_i\left\{2\sum_{j=1}^{N}\bigl(W_{ij}^2 + W_{ji}^2\bigr)\right\}. \end{aligned}$$

Algorithm 2. — inpaint( ):

```
input: $\mathbf{W}$, $\mathbf{y}$, $\mathcal{S}$, $\varepsilon^2$, $\mathbf{x}^{(0)}$, $\tau$
$Z_{ij}^{(0)} = Z_{ij}^{(-1)} = 0$
$\psi_i = 2\sum_{j=1}^{N}(W_{ij}^2 + W_{ji}^2)$
    if $\varepsilon^2 > 0$ then
    $\psi_i = \max\{\psi_1,\dots,\psi_N\}$
    end if
    $\ell = 0$
    repeat
    $\tilde{Z}_{ij}^{(\ell)} = 2Z_{ij}^{(\ell)} - Z_{ij}^{(\ell-1)}$
    $\tilde{z}_i^{(\ell)} = \sum_{j=1}^{N}\bigl(W_{ij}\tilde{Z}_{ij}^{(\ell)} - W_{ji}\tilde{Z}_{ji}^{(\ell)}\bigr)$
    $\hat{x}_i^{(\ell)} = x_i^{(\ell)} + \frac{1}{\tau\psi_i}\tilde{z}_i^{(\ell)}$
    $\eta^{(\ell)} = 1 - \sqrt{\varepsilon^2 / \sum_{i\in\mathcal{S}}\bigl(y_i - \hat{x}_i^{(\ell)}\bigr)^2}$
    if $i\in\mathcal{S}$ and $\eta^{(\ell)} > 0$ then
        $x_i^{(\ell+1)} = \hat{x}_i^{(\ell)} + \eta^{(\ell)}\bigl(y_i - \hat{x}_i^{(\ell)}\bigr)$
    else
        $x_i^{(\ell+1)} = \hat{x}_i^{(\ell)}$
    end if
    $\tilde{Z}_{ij}^{(\ell+1)} = Z_{ij}^{(\ell)} + \tau W_{ij}\bigl(x_j^{(\ell+1)} - x_i^{(\ell+1)}\bigr)$
    $Z_{ij}^{(\ell+1)} = [\tilde{Z}_{ij}^{(\ell+1)}]_{-1}^{+1}$
    $\ell = \ell + 1$
until stopping criterion is satisfied
output: $x_i^{(\ell)}$
```

Thus, a suitable choice for the matrix $\boldsymbol{\Psi}$ is

$$\boldsymbol{\Psi} = \operatorname{Diag}(\psi_1,\dots,\psi_N), \qquad \psi_i = 2\sum_{j=1}^{N}\bigl(W_{ij}^2 + W_{ji}^2\bigr).$$

With this choice, (18) has a closed-form solution only if there is no noise ($\varepsilon^2 = 0$). Otherwise, we resort to choosing $\boldsymbol{\Psi} = \max\{\psi_1,\dots,\psi_N\}\mathbf{I}$. The overall augmented ADMM update steps for solving (14) are summarized in Algorithm 2. For any penalty parameter $\tau > 0$, this algorithm converges to an optimal point of (14) [34, Theorem 1]. Since all steps are scalar and the graph gradient (Z_{ij} updates) needs to be computed only for $(i,j)\in\mathcal{E}$, the per-iteration complexity of the inpainting algorithm scales linearly with the number $K = |\mathcal{E}|$ of graph edges.

As a stopping criterion for Algorithm 2, we use the primal and dual residual (cf. [25, Section 3.3.1]), which in our case are

Algorithm 3. — joint.learn.inpaint( ):

```
input: $\mathbf{Y} = (\mathbf{y}_1, \ldots, \mathbf{y}_M)$, $\mathcal{S}_1, \ldots, \mathcal{S}_M$, $\varepsilon_1^2, \ldots, \varepsilon_M^2$, $\tau$,
    $\mathbf{D}^{(0)}$, $\mu$, $\mathbf{a}$, $\mathbf{b}$, $\mathbf{C}$, $\rho$
 1:  $\mathbf{X}^{(0)} = \mathbf{Y}$
 2:  $q = 0$
 3:  repeat
 4:      $\mathbf{W}^{(q+1)} = \textbf{learn}(\mathbf{D}^{(q)}, \mu, \mathbf{a}, \mathbf{b}, \mathbf{C}, \rho)$
 5:      for $m = 1, \ldots, M$ do
 6:          $\tilde{\mathbf{x}}_m^{(q+1)} =$
                 $\textbf{inpaint}(\mathbf{W}^{(q+1)}, \mathbf{y}_m, \mathcal{S}_m, \varepsilon_m^2, \tilde{\mathbf{x}}_m^{(q)}, \tau)$
 7:      end for
 8:      $\mathbf{X}^{(q+1)} = \big(\tilde{\mathbf{x}}_1^{(q+1)}, \ldots, \tilde{\mathbf{x}}_M^{(q+1)}\big)$
 9:      compute $\mathbf{D}^{(q+1)} = \mathbf{D}(\mathbf{X}^{(q+1)})$ via (1)
10:      $q = q + 1$
11:  until stopping criterion is satisfied
output: $\mathbf{W}^{(q)}$, $\mathbf{X}^{(q)}$
```

respectively given by

$$\begin{aligned}\mathbf{r}^{(\ell+1)} &= -\tau \operatorname{div} \nabla\big(\mathbf{x}^{(\ell+1)} - \mathbf{x}^{(\ell)}\big)\\ &\quad - \operatorname{div}\Big(2\mathbf{Z}^{(\ell)} - \mathbf{Z}^{(\ell-1)} - \mathbf{Z}^{(\ell+1)}\Big),\\ \mathbf{R}^{(\ell+1)} &= \frac{1}{\tau}\Big(\mathbf{Z}^{(\ell+1)} - \mathbf{Z}^{(\ell)}\Big). \end{aligned} \tag{20}$$

Algorithm 2 is stop as soon as the following two conditions are satisfied:

$$\begin{aligned}\|\mathbf{r}^{(\ell)}\|_2 &\leq \epsilon_{\mathrm{abs}}\sqrt{N} + \epsilon_{\mathrm{rel}}\,\|\operatorname{div}\mathbf{Z}^{(\ell)}\|_2,\\ \|\mathbf{R}^{(\ell)}\|_{\mathrm{F}} &\leq \epsilon_{\mathrm{abs}}\sqrt{K} + \epsilon_{\mathrm{rel}}\,\|\nabla \mathbf{x}^{(\ell)}\|_{\mathrm{F}}. \end{aligned} \tag{21}$$

In our experiments we used $\epsilon_{\mathrm{abs}} = 10^{-10}$ and $\epsilon_{\mathrm{rel}} = 10^{-2}$.

C. Summary of Overall Method

The joint graph learning and signal inpainting algorithm for incomplete noisy data is summarized in Algorithm 3. With I_W iterations, the learning step (line 4) has a complexity scaling with KI_W (recall that K is the the number of admissible edges); similarly, the complexity of the inpainting step for the M graph signals (line 5–7) is proportional to KMI_X, where I_X is the number of inpainting iterations; finally, the complexity of (re-)computing the K relevant entries of the discrepancy matrix (line 9) is proportional to KM. Assuming there are I_{o} iterations of the repeat loop (lines 3–11), the overall complexity of the joint learning-inpainting algorithm scales as $I_{\mathrm{o}}K(M + c_1 I_W + c_2 M I_X)$. Since the inpainting step of Algorithm 3 (line 6) uses the solution $\tilde{\mathbf{x}}_m^{(q)}$ from the previous iteration as initialization, the number I_X of inpainting iterations is typically small. Furthermore, we observed in our numerical experiments that about $I_{\mathrm{o}} = 10$ outer iterations are sufficient.

In order to detect convergence of Algorithm 3, we check the relative change in the learned graph weights and the inpainted signal, leading to the stopping criterion

$$\max\left\{\frac{\|\mathbf{W}^{(q)} - \mathbf{W}^{(q-1)}\|_{\mathrm{F}}}{\|\mathbf{W}^{(q)}\|_{\mathrm{F}}}, \frac{\|\mathbf{X}^{(q)} - \mathbf{X}^{(q-1)}\|_{\mathrm{F}}}{\|\mathbf{X}^{(q)}\|_{\mathrm{F}}}\right\} < \delta. \tag{22}$$

In our simulations, we used a relative accuracy of $\delta = 0.01$.

Except for the measurements, the weight constraints, and the empirical error constraints, Algorithm 3 requires an initial discrepancy matrix $\mathbf{D}^{(0)}$ as input (note that the initialization $\mathbf{D}^{(0)}$ matters since (13) is non-convex). In case there is no missing data, $\mathbf{D}^{(0)}$ can be computed directly using the measurements $\mathbf{X}^{(0)} = \mathbf{Y}$. However, in case of incomplete data we have $|Y_{im} - Y_{jm}| = 0$ whenever $Y_{im} = Y_{jm} = 0$, which results in possibly too small discrepancies for nodes with missing data and, as a consequence, too strong edges between such nodes. This problem needs to be dealt with in an application-specific manner, taking into account the range of values of X_{im}. Unless the graph signal values of both nodes are available, we essentially want a "non-informative" contribution to the discrepancy (e.g., the mean or median discrepancy) that is neither in favor nor against placing an edge. Define the set of graph signal indices for which an observation is available both for node i and node j as

$$\bar{\mathcal{S}}_{ij} = \{m \colon i \in \mathcal{S}_m \text{ and } j \in \mathcal{S}_m\}. \tag{23}$$

The partial discrepancy for these nodes then reads

$$\tilde{D}_{ij} = \sum_{m \in \bar{\mathcal{S}}_{ij}} |Y_{im} - Y_{jm}|, \tag{24}$$

and the average discrepancy is $\tilde{D}_{ij}/|\bar{\mathcal{S}}_{ij}|$. Replacing the $M - |\bar{\mathcal{S}}_{ij}|$ missing discrepancies with this average yields

$$\begin{aligned} D_{ij}^{(0)} &= (M - |\bar{\mathcal{S}}_{ij}|)\frac{\tilde{D}_{ij}}{|\bar{\mathcal{S}}_{ij}|} + \tilde{D}_{ij}\\ &= \frac{M\tilde{D}_{ij}}{|\bar{\mathcal{S}}_{ij}|} = \frac{M}{|\bar{\mathcal{S}}_{ij}|}\sum_{m \in \bar{\mathcal{S}}_{ij}} |Y_{im} - Y_{jm}|. \end{aligned} \tag{25}$$

This amounts to a linear extrapolation of the partial discrepancy to the whole data set.

VI. Numerical Experiments

We next illustrate the performance and versatility of our method via numerical experiments on synthetic and real-world data.

A. Approximation of a b-Matched Graph

We first consider a pure graph learning scenario where complete and exact observations are available (i.e., $Y_{im} = X_{im}$ for $i = 1, \ldots, N$, $m = 1, \ldots, M$). The synthetic data set consists of M-dimensional coordinates $\mathbf{x}_i$ of $N = 20000$ nodes that were independently sampled according to a uniform distribution on the M-dimensional hypersphere. As discrepancy we used the squared Euclidean distance $D_{ij} = \|\mathbf{x}_i - \mathbf{x}_j\|_2^2$. Due to the symmetry of the problem, it is reasonable to aim for a regular graph. The state of the art approach here is the b-matched graph construction [6] with a prescribed number b of neighbors; however, this construction amounts to a computationally expensive binary optimization problem. By contrast, the kNN method is computationally cheap but results in a large number of nodes with more than k neighbors.

As an alternative that is computationally efficient and yields an almost regular graph, we use our graph learning method

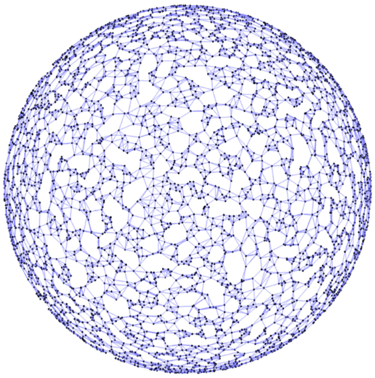


Fig. 3. An almost regular graph learned for $N = 20000$ nodes using Algorithm 1. The nodes are uniformly distributed on a sphere ($M = 3$, top view of northern hemisphere). Here, 92% of nodes have the target degree of $b = 6$.

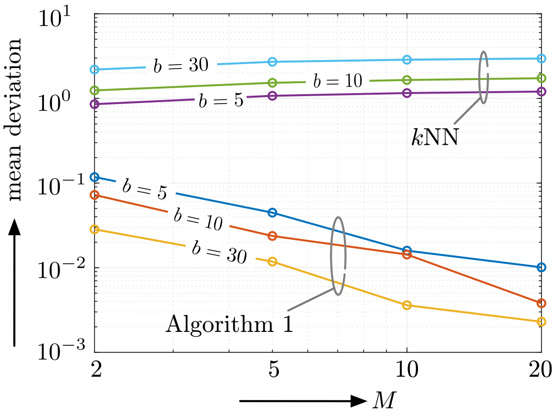


Fig. 4. Mean deviation of the node degrees from the target degree b versus dimension M for different b.

TABLE I
PERCENTAGE OF NODES WHOSE DEGREE EQUALS THE TARGET DEGREE b FOR GRAPHS LEARNED IN DIMENSION M

	Algorithm 1			kNN		
M	$b = 5$	$b = 15$	$b = 30$	$b = 5$	$b = 10$	$b = 30$
2	88.2	92.8	97.2	46.7	40.5	33.9
5	95.5	97.6	98.8	40.0	32.2	21.0
10	98.4	98.6	99.6	40.5	32.5	22.2
20	99.0	99.6	99.8	41.0	32.8	22.8

augmented with a symmetry constraint (cf. Section III-C) and with a kNN-based support constraint that is similar to the approach in [14] (the methods from [12]–[15] are not well suited to control for a regular graph, however). More specifically, since the adjacency matrix $\mathbf{W}_{k\text{NN}} \in \{0, 1\}^{N \times N}$ of the kNN graph has no edges between nodes with large discrepancy, we use the kNN graph with $k = 3b$ as a prior to reduce the dimensionality of the learning problem from $\frac{N(N-1)}{2}$ to $K = \mathbf{1}^{\mathrm{T}}\mathbf{W}_{k\text{NN}}\mathbf{1}/2 \approx 3bN$ by setting $\mathbf{C} = 2\mathbf{W}_{k\text{NN}}/b$. The remaining parameters were chosen as described in Subsection IV-A. An illustrative example of a graph learned in this way for $M = 3$ and $b = 6$ is shown in Figure 3.

The weight matrices delivered by Algorithm 1 were essentially binary-valued. To be able to compare our method to the kNN construction, all non-zero edge weights were set equal to 1 (effectively, a rescaling by $b/2$). We compare the results obtained by the proposed method (Algorithm 1) and by the kNN construction (with $k = b$) for $b \in \{5, 10, 30\}$ and $M \in \{2, 5, 10, 20\}$ using two metrics: Figure 4 shows the mean deviation of all $N = 20000$ node degrees from the target degree b; Table I specifies the percentage of nodes whose degree exactly equals the target degree b (we note that with Algorithm 1 all remaining nodes had degree $b + 1$, whereas with kNN substantially larger node degrees occurred). Both metrics show that Algorithm 1 by far outperforms kNN (the graphs obtained with the two methods have only 70%–90% of edges in common). Interestingly, the performance of our scheme improves with increasing M and increasing b whereas kNN deteriorates as M and b grow. We conclude that our graph learning scheme can be used as an efficient and accurate approximation to b-matched graph constructions.

B. Weight Recovery in Cluster Graphs

We next compare the edge weight recovery performance of Algorithm 1 with state-of-the-art graph learning methods from [12]–[14]. We consider a ground truth graph with four clusters consisting of 10, 20, 40, and 80 nodes, respectively, in two-dimensional space (thus, $N = 150$). The node positions $\mathbf{p}_i$ are randomly arranged inside of ℓ^1 balls of diameter 1 about the cluster centers $(\pm 1, 0)$, $(0, \pm 1)$. The edge weights of the graph are defined as $W_{ij} = (1.75 - \|\mathbf{p}_i - \mathbf{p}_j\|_1)_+$. This model amounts to a weighted version of a stochastic block model with intra-cluster edge probability of 1 and inter-cluster edge probability of 0.03. The observed data comprises 50 noisy measurements of the node positions.

We learned the graph edge weights from these data using Algorithm 1 (with Manhattan discrepancy). Due to the different cluster sizes, we expect different (weighted) node degrees and allow for deviations from the average degree of 2 by factors of $1/8$ and 2, leading to $\mathbf{a} = \frac{1}{4}\mathbf{1}$ and $\mathbf{b} = 4\mathbf{1}$. Furthermore, we control for an edge sparsity of about $1/3$ by setting $\mathbf{C} = \frac{2N}{N^2/3}\mathbf{1}\mathbf{1}^{\mathrm{T}} = \frac{1}{25}\mathbf{1}\mathbf{1}^{\mathrm{T}}$. The regularization parameter μ was then optimally tuned. We compare our algorithm to the Laplacian learning method from [12] with quadratic degree penalty (subsequently referred to as "ℓ^2-penalty"), and the graph learning scheme from [13], [14] with log penalty on the node degrees (referred to as "log-penalty"). Both reference methods use Euclidean discrepancy and were simulated using the GSP toolbox [36] implementations with optimally tuned regularization parameters.

The graph learning performance was assessed from 100 Monte Carlo trials in terms of averages of five metrics:

1) The F-score [21], denoted F, i.e., the harmonic mean of edge precision and recall. The F-score is between 0 and 1, with 1 indicating perfect edge recovery.
2) The Jaccard similarity [37], which is defined as $J = \frac{\sum_{ij}\min\{W_{ij}, \widehat{W}_{ij}\}}{\sum_{ij}\max\{W_{ij}, \widehat{W}_{ij}\}}$ and is a metric particularly well-suited for non-negative quantities.

TABLE II
F-SCORE F (IN PERCENT), JACCARD SIMILARITY J (IN PERCENT), ANGULAR DISTANCE ϕ (IN DEGREES), WEIGHT MSE ϵ_W^2 (IN DECIBEL), AND DEGREE MSE ϵ_d^2 (IN DECIBEL) FOR GRAPH LEARNING WITH ALGORITHM 1, LOG-PENALTY LEARNING, AND ℓ^2-PENALTY LEARNING

	F	J	ϕ	ϵ_W^2	ϵ_d^2
Algorithm 1	96.7	94.6	3.6	−12.0	−37.1
log-penalty	95.8	77.9	17.9	−5.2	−13.6
ℓ^2-penalty	55.8	23.7	52.4	−1.0	−7.5

3) The (scale-invariant) angular distance ϕ, computed via $\cos\phi = \mathrm{tr}\{\mathbf{W}\widehat{\mathbf{W}}^{\mathrm{T}}\}/(\|\mathbf{W}\|\,\|\widehat{\mathbf{W}}\|)$.
4) The normalized mean square error (MSE) $\epsilon_W^2 = \|\widehat{\mathbf{W}} - \mathbf{W}\|^2/\|\mathbf{W}\|^2$ between the learned weights $\widehat{\mathbf{W}}$ and the true weights $\mathbf{W}$.
5) The normalized MSE $\epsilon_d^2 = \|\widehat{\mathbf{W}}\mathbf{1} - \mathbf{W}\mathbf{1}\|^2/\|\mathbf{W}\mathbf{1}\|^2$ between the weighted node degrees of the learned graph and of the true graph.

The results are shown in Table II . Our Algorithm 1 performs best with regard to all five metrics. While the log-penalty method achieves an F-score that is only slightly worse than that of our approach, its performance gap is much larger for the other four metrics that take into account the actual weight values. The ℓ^2-penalty method performs worst by large margins, which is due to its inability to independently control the edge sparsity and the node degrees. The inferiority of the log-penalty and ℓ^2-penalty can be explained by the fact that these methods impose a uniform penalty on the node degrees and hence are not well-suited for strongly irregular graphs. This observation is supported by the far superior degree MSE of our method seen in the last column of Table II (in our example, the node degrees differ by factors of up to 8 across the four clusters).

C. Community Detection From Noisy and Incomplete Data

Next, we study our joint graph learning and inpainting scheme (Algorithm 3) on noisy and incomplete data.

We consider an unweighted graph with $R = 4$ disjoint communities $\mathcal{G}_n$, $n = 1, \ldots, R$, each consisting of 50 nodes (thus $N = 200$). There are no edges between nodes in distinct communities, whereas all nodes within a community are connected. Motivated by the fact that total variation promotes piecewise constant functions [30], [38], we consider graph signals that are constant within communities, i.e., $x_i = a_n$ for $i \in \mathcal{G}_n$, with the signal value in the nth community randomly chosen according to $a_n \sim \mathcal{N}(n, 1)$. The observed data is noisy and incomplete, i.e., $y_i = x_i + e_i$, $i \in \mathcal{S}$, where $e_i \sim \mathcal{N}(0, \sigma^2)$ and $\mathcal{S}$ is a randomly picked sampling set (the sampling rate equals $P = |\mathcal{S}|/N$ and the SNR is given by $\kappa = \mathrm{E}\{x_i^2\}/\sigma^2$).

Using this model, we created $M = 100$ observed graph signals $\tilde{\mathbf{y}}_m$, $m = 1, \ldots, M$, and tried to learn the underlying community graph using Algorithms 1 and 3. The empirical error bounds in (13e) were set to $\varepsilon_m^2 = |\mathcal{S}_m|\sigma^2$. The initial discrepancy matrix was computed according to (25). In order to prevent isolated nodes we used the lower bound $\mathbf{a} = \frac{1}{50}\mathbf{1}$ for the node degrees (no upper bound was imposed). The graph sparsity was controlled by choosing $\mu = 170$. By setting $C_{ij} = 0$ for nodes i and j that belong to two distinct communities whose average signal values differs by more than 1, we reduced the problem dimension by a factor of 5/8. Since the ground truth graph is unweighted, we used the F-score as performance metric.

TABLE III
F-SCORE (IN PERCENT) FOR GRAPHS LEARNED WITH ALGORITHMS 1 AND 3 AT VARIOUS SNRS κ AND SAMPLING RATES P

	Algorithm 3			Algorithm 1		
P	10 dB	15 dB	20 dB	10 dB	15 dB	20 dB
0.1	51.7	67.8	95.4	21.4	32.7	44.7
0.2	65.6	98.0	99.9	26.6	45.7	64.6
0.3	94.3	98.2	100.0	41.8	60.7	77.2
0.4	94.3	98.1	100.0	50.9	68.5	84.5
0.5	95.7	98.7	100.0	56.5	74.8	89.2

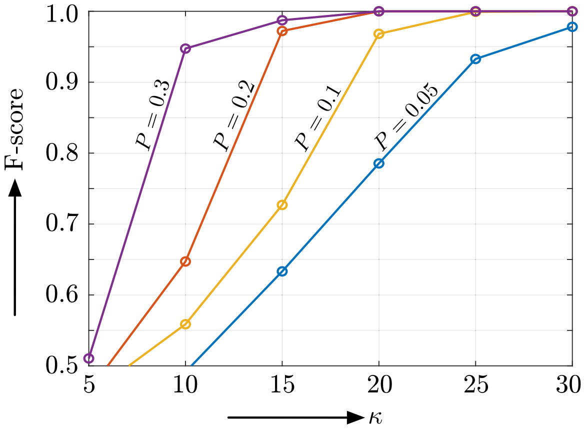


Fig. 5. F-score versus SNR κ for various sampling rates P.

Table III shows the results for sampling rates $P \in \{0.1, 0.2, 0.3, 0.4, 0.5\}$ and SNRs $\kappa \in \{10\,\mathrm{dB}, 15\,\mathrm{dB}, 20\,\mathrm{dB}\}$. It is evident that Algorithm 3 performs substantially better than Algorithm 1, specifically in the regime of low SNR or small sampling rate. In the high SNR regime, Algorithm 3 recovers the ground truth graph (almost) perfectly; even for low-to-medium SNRs the F-score is well above 0.9 for sampling rates larger than $P = 0.2$. Table III suggests that there is a noise threshold for successful recovery of the graph using Algorithm 3. This is confirmed by Figure 5, which shows F-score versus SNR κ for $P \in \{0.05, 0.1, 0.2\}$. Clearly, the smaller the sampling rate, the larger the SNR required to achieve satisfactory graph learning performance.

The F-score results demonstrate that graph learning with denoising/inpainting indeed works well. However, they do not make explicit whether inpainting (i.e., recovery of missing data) is successful. We thus also inspected the inpainted/denoised graph signals obtained by Algorithm 3. Figure 6 shows the normalized MSE of these graph signals versus the sampling rate P at three different SNRs κ. It is seen that, like with the F-score for graph learning, signal inpainting performs excellent. The normalized MSE decreases with increasing sampling rate and increasing SNR.

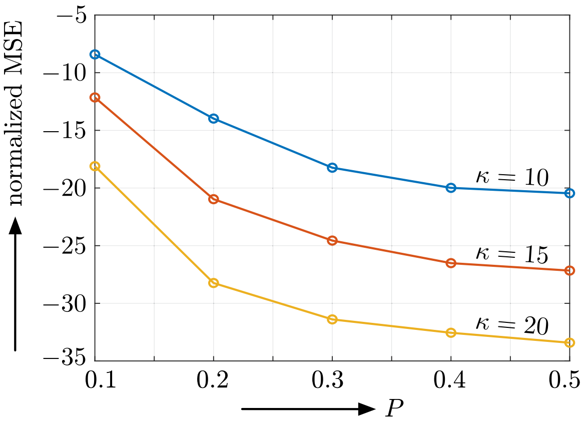


Fig. 6. Normalized MSE of the reconstructed data matrix versus sampling rate P at different SNRs κ.

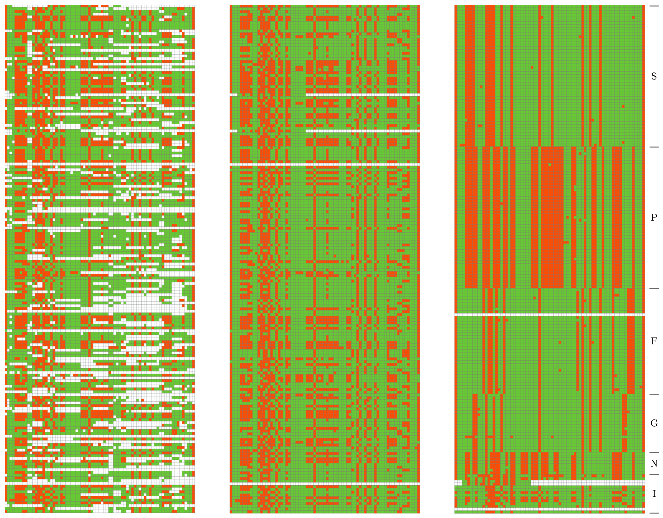


Fig. 7. Observed data matrix (left), data matrix inpainted by Algorithm 3 (middle), inpainted matrix with rows sorted according to party affiliation (right). Green corresponds to “yes,” red to “no,” white indicates missing vote.

D. Austrian Parliamentary Data

1) Context: We next consider real-world data consisting of votes cast during plenary sittings in the Austrian Parliament (courtesy of addendum, `https://politometer.addendum.org`). The Austrian National Council consists of $N = 183$ members (MPs). In the sitting on Oct. 12, 2017, $M = 75$ votes have been conducted, with the attending MPs voting “yes” ($Y_{im} = 1$) or “no” ($Y_{im} = -1$). Numerous MPs were absent during part of the votes ($Y_{im} = 0$), resulting in an incomplete data set (3806 votes, equivalently 28% of the data points, are missing). The corresponding 183 × 75 measurement matrix $\mathbf{Y}$ (with MPs sorted alphabetically) is shown in Figure 7 (left). In this data, there is no noise, $Y_{im} = X_{im}$.

Except for a few independents (I), the MPs are part of a parliamentary group. Each parliamentary group corresponds to an Austrian political party: Social Democratic Party of Austria (S), Austrian People’s Party (P), Freedom Party of Austria (F), NEOS - The New Austria and Liberal Forum (N), and The Greens (G). In any particular vote, MPs from different groups may cast identical votes and certain MPs may cast rebel votes dissenting from the majority of their group. However, we expect that on the long run, MPs from the same group vote coherently. Hence, the ideal ground truth graph connects MPs from the same party and has no edges between MPs from different parties (we emphasize that party affiliation is not part of the data set used for graph learning), i.e., all parties and all independent MPs constitute disjoint cliques.

There is only two different signal values ($Y_{im} = \pm 1$), 6 cliques of very different size, and no prior information. Thus, this data set is more challenging than the synthetic community graph considered before. Fortunately, there is no noise and the sampling rate $P = 0.72$ is rather high, so that we expect that the graph can still be recovered successfully.

2) Graph Learning: We view the MPs as nodes of a graph whose edges connect MPs with similar voting behaviour. Our aim is to learn the graph edges from the given dataset of yes/no votes. To this end, we compare four different approaches:

1) The noise-free Laplacian learning method from [12] with quadratic degree penalty (labeled “ℓ^2-penalty”), $\alpha = 1$ and tuning parameter β. We used the reformulation with discrepancy matrix from [13] and the actual implementation from the GSP toolbox.
2) The GSP toolbox [36] implementation of the graph learning scheme from [13], [14] with log penalty on the node degrees (labeled “log-penalty”). As parameters we used $\alpha = \beta = 1$ and the input matrix was $\mathbf{Z} = \theta \mathbf{D}$ with θ a tuning parameter.
3) Algorithm 1. We choose $\mathbf{a} = \frac{1}{100}\mathbf{1}$ to prevent the graph from having isolated nodes, $\mathbf{b}$ and $\mathbf{C}$ inactive ($b_i \geq 2N$ and $C_{ij} \geq 2N$ for $i \neq j$, $C_{ii} = 0$) and tuned the regularization parameter μ.
4) Algorithm 3. We used noise-free inpainting ($\varepsilon_m^2 = 0, m = 1, \ldots, M$) for the missing observations ($Y_{im} = 0$), $\mathbf{a} = \frac{1}{100}\mathbf{1}$, $\mathbf{b}$ and $\mathbf{C}$ inactive and tuned the parameter μ.

For all graphs obtained in this experiment, edge weights smaller than 0.1% of the largest edge weight were set to zero.

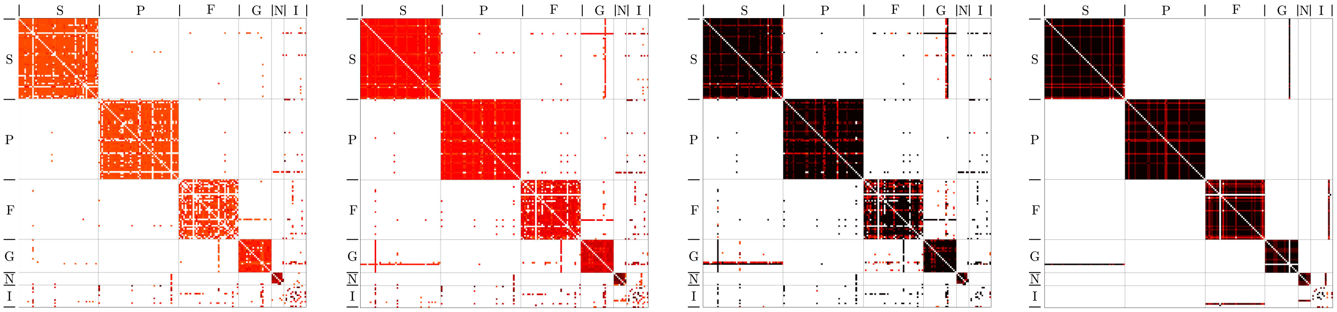
We note that all four algorithms require a discrepancy matrix as input. We provide all algorithms with two specific constructions for the discrepancy matrix. The first one (“extrapolate”) attempts to fill in for the missing data points by using the linear extrapolation (25) as described in Subsection V-C. For $|\bar{\mathcal{S}}_{ij}| = 0$ we set $D_{ij} = 2M$, the discrepancy obtained if representative i votes contrary to representative j in each of the M ballots. With the second construction (“neutral”), the discrepancy is computed as

$$\begin{aligned} D_{ij} &= (M - |\bar{\mathcal{S}}_{ij}|) \cdot 1 + \sum_{m \in \bar{\mathcal{S}}_{ij}} |Y_{im} - Y_{jm}| \\ &= \sum_{m=1}^{M} (1 - Y_{im}Y_{jm}) = M - \sum_{m=1}^{M} Y_{im}Y_{jm}. \end{aligned} \tag{26}$$

This definition builds on the observation that the discrepancies for identical votes is $|Y_{im} - Y_{jm}| = 0$ and for different votes it equals $|Y_{im} - Y_{jm}| = 2$. In both cases we have $|Y_{im} - Y_{jm}| =$

TABLE IV
F-SCORES (F, IN PERCENT) AND ANGULAR DISTANCE (ϕ, IN DEGREES) ACHIEVED BY THE GRAPH LEARNING ALGORITHMS

	ℓ^2-penalty		log-penalty		Algorithm 1		Algorithm 3	
discrepancy	F	ϕ	F	ϕ	F	ϕ	F	ϕ
extrapolate	90.7	48.3	94.2	29.0	94.1	22.3	96.6	21.8
neutral	69.1	54.8	82.1	42.1	75.5	45.3	97.6	16.6



ℓ^2-penalty

log-penalty

Algorithm 1

Algorithm 3

Fig. 8. Weight matrices learned from the parliamentary data (darker color indicates larger weights). Nodes/MPs are sorted according to party affiliation. The ℓ^2-penalty and log-penalty schemes as well as Algorithm 1 use extrapolated discrepancy, Algorithm 3 uses neutral discrepancy.

$1 - Y_{im}Y_{jm}$. If at least one vote is missing ($Y_{im} = 0$), we add the non-informative value $1 - Y_{im}Y_{jm} = 1$, thereby remaining neutral as to whether votes agree or disagree.

3) Discussion: The F-scores achieved by each method (with tuning parameters optimized by trial end error) are summarized in Table IV for both the extrapolating construction (25) and the neutral construction (26) of the discrepancy matrix. Since the F-score only accounts for the presence of an edge but not the associated edge weight, we further show the angular distance ϕ between the learned graph (weight matrix $\widehat{\mathbf{W}}$) and the ground-truth graph (weight matrix $\mathbf{W}_0$).

With the extrapolated discrepancy, ℓ^2-penalty learning performs worst, log-penalty learning and Algorithm 1 perform about the same (with Algorithm 1 achieving smaller angular distance), and Algorithm 3 clearly has the largest F-score and smallest angular distance. Algorithm 3 performs even better with the neutral discrepancy, achieving the largest F-score of 97.6% and the smallest angular distance of 16.6°. The performance of the other three schemes deteriorates with neutral discrepancy since they are not able to inpaint the missing data and hence rely strongly on extrapolation. The inpainted data matrix obtained by Algorithm 3 is shown in Figure 7 (middle) and—with MPs sorted according to party affiliation—in Figure 7 (right). The latter reveals that the MPs within a parliamentary group vote highly coherently, even though there are quite a few "rebel" votes.

Figure 8 depicts the weighted adjacency matrices learned by the four algorithms with optimally tuned parameters and nodes/MPs sorted according to party affiliation. These plots confirm the superior performance of Algorithm 3, which manages almost perfectly to split the MPs into disjoint cliques (parties). Interestingly, Algorithm 2 can indeed be interpreted as a clustering algorithm [39], [40]. It is insightful to examine the edges learned "incorrectly" by Algorithm 3. There is one Green MP who is connected to all Social Democrats; by inspecting the data matrix, it can be seen that this MP was absent precisely from those votes, where the opinions of the S and G groups differed. The graph learning method then assigned that MP to the larger group (in this case S). Furthermore, while most independents vote rather inconsistently, two MPs voted almost identically to the F and N groups. Finally, one MP from the F group and one from the independents form a clique of size 2 on their own; the data matrix reveals that these two MPs participated in none of the votes and hence clearly cannot be correctly assigned to their respective groups (inpainting is not successful either, see Figure 7 (right)).

4) Impact of Sampling Rate and Number of Snapshots: Next we study how the performance of Algorithm 3 on the parliamentary data set (with neutral discrepancy) depends on the number M of snapshots/graph signals. We performed 100 simulation runs in which we randomly selected $M \in \{20, 40, 60\}$ voting rounds (out of 75). The average F-scores we obtained were $F = 0.7$ ($M = 20$), $F = 0.87$ ($M = 40$), and $F = 0.95$ ($M = 60$), clearly confirming the intuition that performance deteriorates with decreasing M.

Finally, we examine the impact of the sampling rate P, i.e., the fraction of known votes. In the original data set, 28% percent of votes were missing, amounting to a sampling rate of 72%. We simulated smaller sampling rates by discarding randomly picked known votes (artificially making MPs absent) and then applied Algorithm 3. The resulting F-scores and the recovery rate (percentage of correctly recovered votes among those that

TABLE V
F-SCORES AND FRACTION OF CORRECTLY RECOVERED VOTES ACHIEVED BY ALGORITHM 3 FOR VARIOUS SAMPLING RATES P (ALL QUANTITIES IN PERCENT)

P	20	30	35	40	50	60
F-score	34.5	68.4	83.9	90.0	92.9	94.5
recovery rate	79.4	94.0	97.1	98.1	98.5	98.7

had been discarded) are shown in Table V. There is a threshold at about $P = 0.35$ below which graph learning deteriorates substantially. Interestingly, vote recovery works well even well below this threshold.

VII. CONCLUSION

In this paper, we formulated the problem of learning a graph from data as a constrained quadratic program. By specializing the ADMM iterations for this problem, we devised a powerful and flexible novel graph learning algorithm. The generality of our framework was demonstrated by showing that several existing graph learning methods can be re-obtained as special cases. Our approach is superior to existing methods in that it offers higher flexibility and more direct control of the graph structure and admits an analytical characterization and efficient computation of the optimal weights. For the practically highly important case of noisy and incomplete data we proposed a joint graph learning and signal inpainting scheme. The complexity of all our algorithms scales linearly with the number of admissible graph edges, which is of utmost importance for large-scale problems. Our numerical experiments on synthetic and real-world data revealed that both the pure graph learning scheme and the joint learning and inpainting method outperform state-of-the-art methods by substantial margins.

Some open problems for future work are (i) the analytic characterization of the thresholds for noise power and sampling rate such that accurate graph reconstruction is feasible, (ii) the automated tuning of the parameters μ, $\mathbf{a}$, $\mathbf{b}$, $\mathbf{C}$ in given application scenarios, and (iii) the exploration of alternative algorithmic approaches for solving the non-convex joint learning-inpainting scheme (e.g., using ideas from [41]).

APPENDIX A
PROOF OF THEOREM 1

The Lagrangian for (6) is

$$\mathcal{L}(\boldsymbol{\lambda},\boldsymbol{\alpha},\boldsymbol{\mu},\boldsymbol{\gamma},r) = \sum_{i=1}^{N}\sum_{j=1}^{N}\Big[W_{ij}D_{ij} + \frac{\mu}{2}W_{ij}^2 - \lambda_{ij}W_{ij} + \nu_{ij}(W_{ij}-C_{ij}) + r(2/N - W_{ij}) + \alpha_i(a_i/N - W_{ij}) + \beta_i(W_{ij} - b_i/N)\Big].$$

The KKT conditions [29] are

$$0 \le W_{ij} \le C_{ij}, \quad a_i \le \sum_{j=1}^{N} W_{ij} \le b_i, \quad \sum_{i=1}^{N}\sum_{j=1}^{N} W_{ij} = 2N,$$

$$\lambda_{ij} \ge 0, \quad \nu_{ij} \ge 0, \quad \alpha_i \ge 0, \quad \beta_i \ge 0,$$

$$\lambda_{ij}W_{ij} = 0, \quad \nu_{ij}(W_{ij} - C_{ij}) = 0,$$

$$\alpha_i\left(a_i - \sum_{j=1}^{N} W_{ij}\right) = 0, \quad \beta_i\left(\sum_{j=1}^{N} W_{ij} - b_i\right) = 0,$$

$$D_{ij} + \mu W_{ij} - \lambda_{ij} + \nu_{ij} - \alpha_i + \beta_i - r = 0.$$

Let us first inspect the case without constraints on on the weighted node degrees and without upper bounds on the weights. In this case the gradient conditions simplifies to

$$D_{ij} + \mu W_{ij} - \lambda_{ij} - r = 0.$$

Complementary slackness ($\lambda_{ij}W_{ij} = 0$) entails

$$\lambda_{ij} = 0 \quad \to \quad W_{ij} = \frac{1}{\mu}(r - D_{ij}),$$

$$W_{ij} = 0 \quad \to \quad \lambda_{ij} = D_{ij} - r \ge 0 \quad \to \quad D_{ij} \ge r.$$

This leads to the closed-form solution

$$W_{ij} = \frac{1}{\mu}[r - D_{ij}]_+,$$

which is similar to water-filling in communications [42, Sec. 9.4]. The waterlevel r is determined by the equation

$$\sum_{i=1}^{N}\sum_{j=1}^{N} W_{ij} = \frac{1}{\mu}\sum_{i=1}^{N}\sum_{j=1}^{N}[r - D_{ij}]_+ = 2N.$$

If we include the constraints $W_{ij} \le C_{ij}$, their duals $\nu_{ij} \ge 0$, and slackness $\nu_{ij}(W_{ij} - C_{ij}) = 0$, the gradient becomes

$$D_{ij} + \mu W_{ij} - \lambda_{ij} + \nu_{ij} - r = 0.$$

For $\nu_{ij} = \lambda_{ij} = 0$, we have $0 \le W_{ij} = \frac{1}{\mu}(r - D_{ij}) \le C_{ij}$; furthermore, when $\lambda_{ij} = 0$, $\nu_{ij} > 0$, necessarily $W_{ij} = C_{ij}$ and when $\lambda_{ij} > 0$, $\nu_{ij} = 0$, necessarily $W_{ij} = 0$. Finally, the case $\lambda_{ij} > 0, \nu_{ij} > 0$ can occur only if $W_{ij} = C_{ij} = 0$. In summary, we obtain

$$W_{ij} = \left[\frac{1}{\mu}(r - D_{ij})\right]_+^{C_{ij}} = \frac{1}{\mu}[r - D_{ij}]_+^{\mu C_{ij}}.$$

Again, the water level r needs to be chosen such that

$$\sum_{i=1}^{N}\sum_{j=1}^{N} W_{ij} = \frac{1}{\mu}\sum_{i=1}^{N}\sum_{j=1}^{N}[r - D_{ij}]_+^{\mu C_{ij}} = 2N.$$

Next, we take into account the degree constraints $a_i \le \sum_{j=1}^{N} W_{ij} \le b_i$, with dual constraints $\alpha_i, \beta_i \ge 0$, and slackness conditions $\alpha_i(a_i - \sum_j W_{ij}) = 0$, $\beta_i(\sum_j W_{ij} - b_i) = 0$. If $\alpha_i = \beta_i = 0$ the ith degree constraint is not active and we recover the solution from the previous case. If $\alpha_i > 0, \beta_i = 0$, then slackness implies $\sum_j W_{ij} = a_i$ and the gradient condition together with the other constraints leads to

$$W_{ij} = \frac{1}{\mu}[r - D_{ij} + \alpha_i]_+^{\mu C_{ij}},$$

with α_i chosen such that $\sum_j [r - D_{ij} + \alpha_i]_+^{\mu C_{ij}} = \mu a_i$. Similarly, for $\alpha_i = 0$, $\beta_i > 0$, slackness entails $\sum_j W_{ij} = b_i$ and furthermore

$$W_{ij} = \frac{1}{\mu}[r - D_{ij} - \beta_i]_+^{\mu C_{ij}},$$

with β_i chosen such that $\sum_j [r - D_{ij} - \beta_i]_+^{\mu C_{ij}} = \mu b_i$. The case $\alpha_i > 0$, $\beta_i > 0$ can occur only if $\sum_{j=1}^N W_{ij} = a_i = b_i$, i.e., when the inequality constraints degenerate to an equality constraint. Here,

$$W_{ij} = \frac{1}{\mu}[r - D_{ij} + \alpha_i - \beta_i]_+^{\mu C_{ij}},$$

with α_i and β_i chosen such that $\sum_j [r - D_{ij} + \alpha_i - \beta_i]_+^{\mu C_{ij}} = \mu a_i = \mu b_i$. All of these cases can be thought of discounting the water level r (by adding α_i or subtracting β_i) in the baseline solution in order to meet the degree constraint.

Appendix B
Proof of Lemma 2

We compute the inverse of the matrix $\mathbf{G} \triangleq \mu \mathbf{I}_K + \rho \mathbf{H}^{\mathrm{T}}\mathbf{H} + \rho \mathbf{J}_K$. We use the notation $\mathrm{Diag}(v_1, \ldots, v_N)$ and $\mathrm{Diag}(\mathbf{V}_1, \ldots, \mathbf{V}_N)$ to denote (block) diagonal matrices with diagonal elements $v_1, \ldots, v_N$ and diagonal blocks $\mathbf{V}_1, \ldots, \mathbf{V}_N$, respectively. Note that the rows $\mathbf{b}_i^{\mathrm{T}}$, $i = 1, \ldots, N$ of the matrix $\mathbf{B}$ are orthogonal with squared norms k_i and thus $\mathbf{B}\mathbf{B}^{\mathrm{T}} = \mathrm{Diag}(k_1, \ldots, k_N)$; furthermore,

$$\mathbf{B}^{\mathrm{T}}\mathbf{B} = \mathrm{Diag}(\mathbf{J}_{k_1}, \ldots, \mathbf{J}_{k_N}) \in \{0,1\}^{K \times K}. \quad (27)$$

Using (27), the Gramian of $\mathbf{H} = (-\mathbf{I}_K \;\; \mathbf{I}_K \;\; -\mathbf{B}^{\mathrm{T}} \;\; \mathbf{B}^{\mathrm{T}})^{\mathrm{T}}$ can be developed as

$$\mathbf{H}^{\mathrm{T}}\mathbf{H} = 2(\mathbf{I}_K + \mathbf{B}^{\mathrm{T}}\mathbf{B}) = 2\,\mathrm{Diag}(\mathbf{I}_{k_1} + \mathbf{J}_{k_1}, \ldots, \mathbf{I}_{k_N} + \mathbf{J}_{k_N}).$$

We thus obtain $\mathbf{G} = \mathbf{F} + \rho \mathbf{J}_K$ with

$$\mathbf{F} \triangleq \mathrm{Diag}(\eta \mathbf{I}_{k_1} + 2\rho \mathbf{J}_{k_1}, \ldots, \eta \mathbf{I}_{k_1} + 2\rho \mathbf{J}_{k_N}),$$

where we used the shorthand notation $\eta \triangleq \mu + 2\rho$. It is seen that $\mathbf{G}$ is the sum of the block diagonal matrix $\mathbf{F}$ and the rank-one matrix $\rho \mathbf{J}_K$. Furthermore, the diagonal blocks $\mathbf{F}_i \triangleq \eta \mathbf{I}_{k_i} + 2\rho \mathbf{J}_{k_i}$ themselves are rank-one modifications of a (scaled) identity matrix. We can thus apply the Sherman-Morrison formula [43] to get

$$\mathbf{G}^{-1} = (\mathbf{F} + \rho \mathbf{1}\mathbf{1}^{\mathrm{T}})^{-1} = \mathbf{F}^{-1} - \frac{\mathbf{F}^{-1}\rho \mathbf{1}\mathbf{1}^{\mathrm{T}}\mathbf{F}^{-1}}{1 + \rho \mathbf{1}^{\mathrm{T}}\mathbf{F}^{-1}\mathbf{1}}. \quad (28)$$

Furthermore, we have $\mathbf{F}^{-1} = \mathrm{Diag}(\mathbf{F}_1^{-1}, \ldots, \mathbf{F}_N^{-1})$. Applying the Sherman-Morrison formula once again,

$$\mathbf{F}_i^{-1} = (\eta \mathbf{I}_{k_i} + 2\rho \mathbf{1}_{k_i}\mathbf{1}_{k_i}^{\mathrm{T}})^{-1} = \frac{1}{\eta}\left[\mathbf{I}_{k_i} - \frac{2\rho}{\eta + 2\rho k_i}\mathbf{J}_{k_i}\right].$$

Using these inverses and the definition $\gamma_i \triangleq 1/(\eta + 2\rho k_i)$, we can compute $\mathbf{f} \triangleq \mathbf{F}^{-1}\mathbf{1} = (\mathbf{f}_1^{\mathrm{T}}, \ldots, \mathbf{f}_N^{\mathrm{T}})^{\mathrm{T}}$ with

$$\mathbf{f}_i = \mathbf{F}_i^{-1}\mathbf{1}_{k_i} = \frac{1}{\eta}\left[1 - \frac{2\rho k_i}{\eta + 2\rho k_i}\right]\mathbf{1}_{k_i} = \gamma_i \mathbf{1}_{k_i}.$$

Inserting these expressions into the last term of (28) yields

$$\frac{\mathbf{F}^{-1}\rho \mathbf{1}\mathbf{1}^{\mathrm{T}}\mathbf{F}^{-1}}{1 + \rho \mathbf{1}^{\mathrm{T}}\mathbf{F}^{-1}\mathbf{1}} = \frac{\rho}{1 + \rho\bar{\gamma}}\mathbf{f}\mathbf{f}^{\mathrm{T}},$$

where $\bar{\gamma} \triangleq \sum_{i=1}^N \gamma_i k_i$. Combining all intermediate results, we finally obtain

$$\mathbf{G}^{-1} = \frac{1}{\eta}\mathbf{I} - \frac{2\rho}{\eta}\,\mathrm{Diag}\{\gamma_1 \mathbf{J}_{k_1}, \ldots, \gamma_N \mathbf{J}_{k_N}\} - \frac{\rho}{1 + \rho\bar{\gamma}}\mathbf{f}\mathbf{f}^{\mathrm{T}}. \quad (29)$$

Note that $\mathbf{f}\mathbf{f}^{\mathrm{T}}$ is a block matrix whose (i,j)th block equals $\gamma_i\gamma_j \mathbf{1}_{k_i}\mathbf{1}_{k_j}^{\mathrm{T}}$. Let $\mathbf{v} \triangleq (\mathbf{v}_1^{\mathrm{T}}, \ldots, \mathbf{v}_N^{\mathrm{T}})^{\mathrm{T}}$ where $\mathbf{v}_i$ has length k_i and let $\mathbf{w} = -\mathbf{G}^{-1}\mathbf{v}$ be partitioned in a similar manner. With (29) it follows that

$$\mathbf{w}_i = -\frac{1}{\eta}\mathbf{v}_i + \left(\frac{2\rho\gamma_i\delta_i}{\eta} + \frac{\rho\gamma_i\bar{\delta}}{1 + \rho\bar{\gamma}}\right)\mathbf{1}_{k_i},$$

where $\delta_i = \mathbf{1}_{k_i}^{\mathrm{T}}\mathbf{v}_i$ and $\bar{\delta} = \sum_{i=1}^N \gamma_i\delta_i$.

References

[1] P. Berger, M. Buchacher, G. Hannak, and G. Matz, "Graph learning based on total variation minimization," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, Apr. 2018, pp. 6309–6313.

[2] 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, May 2013.

[3] A. Sandryhaila and J. M. F. 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, Sep. 2014.

[4] A. Sandryhaila and J. M. F. Moura, "Discrete signal processing on graphs," *IEEE Trans. Signal Process.*, vol. 61, no. 7, pp. 1644–1656, Apr. 2013.

[5] W. Liu, J. Wang, and S. F. Chang, "Robust and scalable graph-based semisupervised learning," *Proc. IEEE*, vol. 100, no. 9, pp. 2624–2638, Sep. 2012.

[6] T. Jebara, J. Wang, and S. Chang, "Graph construction and *b*-matching for semi-supervised learning," in *Proc. Int. Conf. Mach. Learn.*, Jun. 2009, pp. 441–448.

[7] F. Wang and C. Zhang, "Label propagation through linear neighborhoods," *IEEE Trans. Knowl. Data Eng.*, vol. 20, no. 1, pp. 55–67, Jan. 2008.

[8] U. Luxburg, "A tutorial on spectral clustering," *Statist. Comput.*, vol. 17, no. 4, pp. 395–416, Dec. 2007.

[9] K. Ozaki, M. Shimbo, M. Komachi, and Y. Matsumoto, "Using the mutual *k*-nearest neighbor graphs for semi-supervised classification of natural language data," in *Proc. Conf. Comput. Natural Lang. Learn.*, Jun. 2011, pp. 154–162.

[10] J. Wang, T. Jebara, and S.-F. Chang, "Semi-supervised learning using greedy max-cut," *J. Mach. Learn. Res.*, vol. 14, no. 1, pp. 771–800, Mar. 2013.

[11] J. L. Bentley, D. F. Stanat, and E. H. Williams, "The complexity of finding fixed-radius near neighbors," *Inf. Process. Lett.*, vol. 6, no. 6, pp. 209–212, Dec. 1977.

[12] X. Dong, D. Thanou, P. Frossard, and P. Vandergheynst, "Learning Laplacian matrix in smooth graph signal representations," *IEEE Trans. Signal Process.*, vol. 64, no. 23, pp. 6160–6173, Dec. 2016.

[13] V. Kalofolias, "How to learn a graph from smooth signals," in *Proc. Int. Conf. Artif. Intell. Statist.*, May 2016, vol. 51, pp. 920–929.

[14] V. Kalofolias and N. Perraudin, "Large scale graph learning from smooth signals," in *Proc. Int. Conf. Learn. Representations*, New Orleans LA, May 2019.

[15] S. P. Chepuri, S. Liu, G. Leus, and A. O. Hero, "Learning sparse graphs under smoothness prior," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, Mar. 2017, pp. 6508–6512.

[16] S. Segarra, A. G. Marques, G. Mateos, and A. Ribeiro, "Network topology inference from spectral templates," *IEEE Trans. Signal Inf. Process. Netw.*, vol. 3, no. 3, pp. 467–483, Sep. 2017.

[17] B. Pasdeloup, V. Gripon, G. Mercier, D. Pastor, and M. G. Rabbat, "Characterization and inference of graph diffusion processes from observations of stationary signals," *IEEE Trans. Signal Inf. Process. Netw.*, vol. 4, no. 3, pp. 481–496, Aug. 2017.

[18] S. Sardellitti, S. Barbarossa, and P. D. Lorenzo, "Graph topology inference based on transform learning," in *Proc. IEEE Global Conf. Signal Inf. Process. (GlobalSIP)*, Dec. 2016, pp. 356–360.

[19] S. I. Daitch, J. A. Kelner, and D. A. Spielman, “Fitting a graph to vector data,” in *Proc. Int. Conf. Mach. Learn.*, Jun. 2009, pp. 201–208.

[20] M. Sundin, A. Venkitaraman, M. Jansson, and S. Chatterjee, “A connectedness constraint for learning sparse graphs,” in *Proc. 25th Eur. Signal Process. Conf. EUSIPCO*, Aug. 2017, pp. 161–165.

[21] H. E. Egilmez, E. Pavez, and A. Ortega, “Graph learning from data under Laplacian and structural constraints,” *IEEE J. Sel. Topics Signal Process.*, vol. 11, no. 6, pp. 825–841, Sep. 2017.

[22] M. Slawski and M. Hein, “Estimation of positive definite M-matrices and structure learning for attractive Gaussian Markov random fields,” *Linear Algebra Appl.*, vol. 473, pp. 145–179, May 2015.

[23] E. Pavez and A. Ortega, “Generalized Laplacian precision matrix estimation for graph signal processing,” in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, Mar. 2016, pp. 6350–6354.

[24] B. M. Lake and J. B. Tenenbaum, “Discovering structure by learning sparse graph,” in *Proc. Cognitive Sci. Conf.*, Aug. 2010, pp. 778–783.

[25] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, “Distributed optimization and statistical learning via the alternating direction method of multipliers,” *Found. Trends Mach. Learn.*, vol. 3, no. 1, pp. 1–122, Jan. 2011.

[26] E. Ghadimi, A. Teixeira, I. Shames, and M. Johansson, “Optimal parameter selection for the alternating direction method of multipliers (ADMM): Quadratic problems,” *IEEE Trans. Autom. Control*, vol. 60, no. 3, pp. 644–658, Mar. 2015.

[27] S. Li and Y. Fu, “Learning balanced and unbalanced graphs via low-rank coding,” *IEEE Trans. Knowl. Data Eng.*, vol. 27, no. 5, pp. 1274–1287, May 2015.

[28] B. Huang and T. Jebara, “Fast *b*-matching via sufficient selection belief propagation,” in *Proc. Int. Conf. Artif. Intell. Statist.*, Apr. 2011, pp. 361–369.

[29] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, U.K.: Cambridge Univ. Press, 2004.

[30] P. Berger, G. Hannak, and G. Matz, “Graph signal recovery via primal-dual algorithms for total variation minimization,” *IEEE J. Sel. Topics Signal Process.*, vol. 11, no. 6, pp. 842–855, Sep. 2017.

[31] M. Zhu and T. Chan, “An efficient primal-dual hybrid gradient algorithm for total variation image restoration,” Univ. California, Los Angeles, Los Angeles, CA, USA, CAM Rep. 08-34, 2008.

[32] E. Esser, X. Zhang, and T. Chan, “A general framework for a class of first order primal-dual algorithms for TV minimization,” Univ. California, Los Angeles, Los Angeles, CA, USA, CAM Report 09-67, 2009.

[33] A. Chambolle and T. Pock, “A first-order primal-dual algorithm for convex problems with applications to imaging,” *J. Math. Imag. Vision*, vol. 40, no. 1, pp. 120–145, 2011.

[34] Y. Zhu, “An augmented ADMM algorithm with application to the generalized lasso problem,” *J. Comput. Graph. Statist.*, vol. 26, no. 1, pp. 195–204, Feb. 2017.

[35] G. Gilboa and S. Osher, “Nonlocal operators with applications to image processing,” *Multiscale Model. Simul.*, vol. 7, no. 3, pp. 1005–1028, Nov. 2008.

[36] N. Perraudin *et al.*, “GSPBOX: A toolbox for signal processing on graphs,” 2014, *arXiv:1408.5781*.

[37] H. Späth, “The minisum location problem for the Jaccard metric,” *OR Spectrum*, vol. 3, no. 2, pp. 91–94, 1981.

[38] X. Bresson, T. Laurent, D. Uminsky, and J. H. von Brecht, “Multiclass total variation clustering,” in *Proc. Int. Conf. Neural Inf. Process. Syst.*, Dec. 2013, pp. 1421–1429.

[39] P. Berger, T. Dittrich, and G. Matz, “Semi-supervised clustering based on signed total variation,” in *Proc. IEEE Global Conf. Signal Inf. Process. (GlobalSIP)*, Nov. 2018, pp. 793–797.

[40] K. Yin and X. Tai, “An effective region force for some variational models for learning and clustering,” *J. Sci. Comput.*, vol. 74, no. 1, pp. 175–196, Jan. 2018.

[41] H. Attouch, J. Bolte, P. Redont, and A. Soubeyran, “Proximal alternating minimization and projection methods for nonconvex problems: An approach based on the Kurdyka-Łojasiewicz inequality,” *Math. Oper. Res.*, vol. 35, no. 2, pp. 438–457, 2010.

[42] T. M. Cover and J. A. Thomas, *Elements of Information Theory*. New York, NY, USA: Wiley, 1991.

[43] G. H. Golub and C. F. Van Loan, *Matrix Computations*, 3rd ed. Baltimore, MD, USA: The Johns Hopkins Univ. Press, 1996.

Peter Berger received the Dipl.-Ing. degree in technical mathematics from the University of Innsbruck, Innsbruck, Austria, in 2011, and the Ph.D. degree in mathematics from the University of Vienna, Vienna, Austria, in 2015. From 2015 to 2019, he was with the Institute of Telecommunications, Vienna University of Technology, with an intermediate research period with the Department of Computer Science, Aalto University, Espoo, Finland. Since 2019, he has been a Research Engineer with MED-EL Medical Electronics. His research interests include the areas of signal processing, inverse problems, sampling theory, and numerical analysis.

Gabor Hannak received the M.Sc. and Ph.D. degrees in electrical engineering and information technology from the Vienna University of Technology, Vienna, Austria, with major in Telecommunications and Bayesian compressed sensing, in 2013 and 2017, respectively. In 2017, he was a Visiting Research Scholar with the University of Edinburgh, Edinburgh, U.K. Since 2019, he has been a Research Engineer with Nokia Bell Labs. His research interests include low-dimensional representations, compressed sensing, and signal processing.

Gerald Matz (SM’07) received the Dipl.-Ing. and Dr. Techn. degrees in electrical engineering and the Habilitation degree in communication systems from the Vienna University of Technology, Vienna, Austria, in 1994, 2000, and 2004, respectively.

He has directed or actively participated in several research projects funded by the Austrian Science Fund, by the Viennese Science and Technology Fund, and by the European Union. He currently is a tenured Associate Professor with the Institute of Telecommunications, Vienna University of Technology. He has held visiting positions with the Laboratoire des Signaux et Systèmes, Ecole Supérieure d’Electricité (France, 2004), the Communication Theory Lab, ETH Zurich (Switzerland, 2007), and with Ecole Nationale Supérieure d’Electrotechnique, d’Electronique, d’Informatique et d’Hydraulique de Toulouse (France, 2011). He has authored and coauthored more than 220 scientific articles in international journals, conference proceedings, and edited books. He is coeditor of the book *Wireless Communications over Rapidly Time-Varying Channels* (Academic, New York, NY, USA, 2011). His research interests include wireless networks, statistical signal processing, information theory, and big data.

Prof. Matz was a member of the IEEE SPS Technical Committee on Signal Processing Theory and Methods (2010–2015) and of the IEEE SPS Technical Committee on Signal Processing for Communications and Networking (2008–2013). He was an Associate Editor of the IEEE TRANSACTIONS ON INFORMATION THEORY (2013–2015), of the IEEE TRANSACTIONS ON SIGNAL PROCESSING (2006–2010), of the EURASIP journal Signal Processing (2007–2010), and of the IEEE Signal Processing Letters (2004–2008). He was the General Chair of Asilomar 2019 and was Technical Program Chair of Asilomar 2016 and Technical Program Co-Chair of EUSIPCO 2004. He has been a member of the Technical Program Committee of numerous international conferences. In 2006, he received the Kardinal Innitzer Most Promising Young Investigator Award. He is a member of the OVE.