An Outlier-Robust Smoothness-based Graph Learning Approach

Hesam Araghi, Massoud Babaie-zadeh

[a]*Sharif University of Technology, Tehran, Iran*

Abstract

Graph learning (GL) is a tool for finding direct relationships between the nodes of a network, and hence, inferring the graph topology from the data. Recently, many GL algorithms have been proposed in the field of graph signal processing, which are based on smoothness of the graph signals on the learned graph. However, although it is possible for the input graph signals to be contaminated by outliers, for example due to sensor failures or temporary faulty information records, existing techniques are very vulnerable to outliers. So, the goal is to infer a graph topology to be, as much as possible, insensitive to this kind of data corruptions. To this aim, due to the sparse nature of outlier data, we propose a new approach for robustifying GL algorithms by incorporating L1-norm or squared L1-norm terms into the objective function of smoothness based GL methods, yielding to a non-convex minimization problem. A novel iterative minimization method is introduced to solve the resulting non-convex problem. Moreover, the convergence of the algorithm is established despite of its non-convex nature. In simulations, the high performance of the proposed algorithm is demonstrated in presence of a considerably large amount of outliers.

Keywords: Graph signal processing, graph learning, outlier compensation, block coordinate descent, convergence analysis

1. Introduction

Graph signals are useful tools for representing data with complicated structures such as brain, social, or sensor networks, to name a few. In the case that the underlying graph of the data is known a priori, there exist powerful tools in the field of graph signal processing (GSP) that can offer

a good range of data analyzing and inferring techniques [1, 2, 3]. However, the number of applications in which the underlying graphs of the data are available may be limited, emphasizing the necessity of learning them from the data. This strengthens the motivation of proposing many GSP-based graph learning (GL) algorithms in recent years, trying to extract the graph from the data [4, 5].

Before GSP point of view, GL techniques are mainly based on probabilistic models. In these models, the joint probability distribution of the data is considered to be determined by the graph topology [6]. An important model is Gaussian Markov random field (GMRF), which uses inverse covariance matrix (i.e. precision matrix) to represent the graph. For examples, the works [7, 8, 9] make a multivariate Gaussian assumption for the data distribution, and their goal is to find the precision matrix by GMRF estimation. Afterwards, some prior information, such as sparsity, is introduced in GL methods, that are based on precision matrix estimation [7, 8]. Particularly, one of the most renowned method is the graphical LASSO algorithm proposed in [8]. However, in these works, the precision matrix is only constrained to be sparse. Thus, the resulting graph may have negative edge weights or self-loops, which may not be very well-interpretable in some applications [4]. So to prevent the mentioned problems, the precision matrix is restricted to have Laplacian matrix structure in [9].

Introducing the GSP field helps the proposed GL methods to offer more interpretable graphs. In [10], data is modeled by factor analysis with a prior distribution that forces the signal to be smooth on the graph. Then, data smoothness over the underlying graph has been used in Maximum Likelihood Estimator (MLE) to infer the graph structure. It leads to a similar optimization problem as in the precision matrix estimation but with constraints that guarantee the resulting matrix to have the structure of a Laplacian matrix [10]. In [11], various regularization terms are provided in order to extract more accurate graphs. More computationally-efficient optimization techniques are also derived for each of the cases. Other previous works include GL by structural dictionary learning [12, 13], using pre-defined spectral templates to learn the eigenvalues of the Laplacian matrix [14], using non-convex regularization terms [15], and graph learning problem in the presence of incomplete data [16]. For learning multiple graphs from the data, the authors of [17] proposed graph Laplacian mixture model (GLMM) algorithm. K-graphs [18] is also another multiple GL algorithm inspired by K-means. Unlike GLMM and K-graphs, which assume that the graph signals are in-

dependent in different time stamps, there are dynamic GL algorithms that consider the time dependency of signals. In [19], a graphical LASSO based algorithm, called TICC, is proposed that partitions the whole time span into multiple segments, and assigns for each segment a constant precision matrix. Similar to TICC [19], a dynamic GL algorithm, called dynamic K-graphs, is proposed in [20], but since it works with Laplacian matrices, it can offer more interpretable graph structures. Recently, the authors of the [21] use non-convex penalties for graph topology changes to improve the dynamic graph learning performance. In [22], a time-varying framework is proposed for graph learning models to reach the optimal time-varying solution.

However, for the graph learning task, compensating the effect of outlier data is not considered in the previous graph learning algorithms in the GSP field [2]. Maybe the most related work is [23], where the authors propose a robust graph learning model for clustering and semisupervised classification tasks, while our main focus is on the graph learning task. Their method can be viewed as an extension of graph regularized robust principal component analysis (RPCA) method [24] where the graph is constructed automatically by the algorithm. The outlier data can be caused by the reasons such as a sensor failure in sensor networks, an intentionally unfair score for a movie by a user in the user/movie rating data, and salt-and-pepper-like noise in a set of digital image patches. The nature of outlier noise is different from that of white additive noise. For example, there are small deviations from the true values in the sensor readings due to the thermal noise or manufacturing inaccuracies, while sensor failures are rare, but capable of causing larger error values. In other words, the outlier data has a sparse nature, in contrast to additive white noise. On the other hand, as will be shown in the simulations of this paper, existing GL methods are vulnerable to outlier data. Hence, it is important to robustify the GL algorithms to outlier data.

In this paper, an outlier-robust GL algorithm is proposed with the capability of compensating outliers in the data. More precisely, the goal is to *jointly* learn the graph topology of the data and removing outliers from the data. To this end, smoothness-based GL approaches are reconsidered in order to provide a tool for learning graphs in the presence of outlier data. Utilizing the sparsity property of the outlier data, ℓ_1-norm or squared ℓ_1-norm is added to the objective function. Note that in compressed sensing literature, adopting family of ℓ_1-norms has been a common way of dealing with outliers in the data [25, 26, 27]. However, in graph learning, this modification of the problem results in a non-convex minimization. So, another contribution of

this paper is to introduce a novel block coordinate descent (BCD) method that not only solves the resulting non-convex problem with less computational complexity, but also has a more straightforward convergence analysis. In fact, we will show that the proposed BCD iterative method converges to a stationary point. Moreover, the results of the numerical simulations show that our algorithm, in the presence of outlier data, has a better GL accuracy in comparison to an state-of-the-art uncompensated GL algorithm. The sensitivity of the algorithm to the input data scale is also studied in the simulations, and it is demonstrated that the squared ℓ_1-norm version of the algorithm is appreciably less sensitive to the scale of the input data.

The rest of the paper is organized as follows. In Section 2, some required basic knowledge on GSP and proximal operators are very briefly reviewed. Then, our outlier-robust GL algorithm is proposed in Section 3, and its convergence analysis is also studied. Finally, Section 4 is devoted to simulation results.

2. Background

In this section, a brief review on GSP basics and GL algorithms that are based on the smoothness assumption is provided. Additionally, a basic description of the proximal operator is presented, which is required in our work.

2.1. Graph Signal Processing

Let $\mathcal{G}(\mathcal{V}, \mathcal{E}, \mathbf{W})$ represent an undirected, positive weighted graph, where $\mathcal{V}$ and $\mathcal{E} \in \mathcal{V} \times \mathcal{V}$ are the sets of nodes and edges, respectively, with N number of nodes ($|\mathcal{V}| = N$). The weighted adjacency matrix $\mathbf{W} \in \mathbb{R}^{N\times N}$ is symmetric with zero values on its diagonal entries. The (i, j)-th entry of $\mathbf{W}$, denoted by w_{ij}, is the edge weight between the nodes i and j. If there is no edge between the nodes i and j, then $w_{ij} = w_{ji} = 0$. The diagonal matrix $\mathbf{D}$ is the degree matrix containing the degree of the i-th node in its (i, i)-th entry, i.e. $d_{ii} = \sum_{j=1}^{N} w_{ij}$. Another important matrix is graph Laplacian matrix defined as $\mathbf{L} \triangleq \mathbf{D} - \mathbf{W}$ [6], and the set of valid graph Laplacian matrices is

$$\mathcal{L} = \left\{\mathbf{L} \in \mathbb{R}^{N\times N} : \mathbf{L} = \mathbf{L}^T, l_{ij} \leq 0 \ (\forall i \neq j), \mathbf{L}.\mathbf{1} = \mathbf{0}\right\}, \tag{1}$$

where $\mathbf{0}$, $\mathbf{1}$, and $(.)^T$ denote all-zeros vector, all-one vector, and matrix transportation, respectively.

A graph signal $\mathbf{x} \in \mathbb{R}^{N\times 1}$ is a vector whose entries assign real values to the vertices of the graph [2, 1]. An important expression in GSP field is the graph Laplacian quadratic form [1] defined as

$$\mathbf{x}^T\mathbf{L}\mathbf{x} = \frac{1}{2}\sum_{i,j} w_{ij}(x_i - x_j)^2, \tag{2}$$

where it can be considered as a smoothness measurement criterion for a graph signal over the graph. A smaller value of quadratic form indicates that the graph signal $\mathbf{x}$ has small deviations along the connected nodes, specially the strongly connected ones. Equation (3) also shows that the graph Laplacian matrix is positive semi-definite (PSD) since the edge weights are all non-negative.

Being symmetric and PSD, an eigenvalue decomposition can be performed on the Laplacian matrix, $\mathbf{L} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^T$, with orthonormal set of eigenvectors $\mathbf{V}$, and the diagonal matrix $\mathbf{\Lambda}$ containing the corresponding non-negative eigenvalues $\lambda_1 \leq \ldots \leq \lambda_N$, as its diagonal entries. The columns of $\mathbf{V}$ are interpreted as Graph Fourier Transform (GFT) basis, where λ_n's are the coresponding frequencies [1]. The smaller λ_n results in smoother $\mathbf{v}_n$ over the graph. Therefore, the GFT of a graph signal $\mathbf{x}$ can be obtained as $\widehat{\mathbf{x}} = \mathbf{V}^T\mathbf{x}$, and the inverse GFT as $\mathbf{x} = \mathbf{V}\widehat{\mathbf{x}}$.

2.2. Smoothness-based GL approach

Let $\mathbf{x}_1, \ldots, \mathbf{x}_M$ be a set of graph signals. In smoothness-based GL algorithms [10, 11, 18, 4], the idea is to learn the graph by minimizing the Laplacian quadratic function (2) for all these signals,

$$\text{trace}(\mathbf{X}^T\mathbf{L}\mathbf{X}) = \frac{1}{2}\sum_{m=1}^{M}\sum_{i,j} w_{ij}(x_{im} - x_{jm})^2, \tag{3}$$

where $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_M] \in \mathbb{R}^{N\times M}$ is the matrix consisting of all the graph signals. However, minimizing (3) with respect to the Laplacian matrix $\mathbf{L}$ (or equivalently w_{ij}'s) results in the trivial solution of all zeros for edge weights. This can be avoided by adding regularization terms to the objective function,

$$\underset{\mathbf{L}\in\mathcal{L}}{\text{minimize}}\ \text{trace}(\mathbf{X}^T\mathbf{L}\mathbf{X}) + f(\mathbf{L}), \tag{4}$$

where $f(\mathbf{L})$ contains the regularization terms and $\mathcal{L}$ is the set of all valid Laplacian matrices.

2.3. Proximal Operator

Let $f : \mathbb{R}^m \to \mathbb{R} \cup \{\infty\}$ be a proper, semi-continuous, convex function [28] with a non-empty domain. The proximal operator of f at point $\mathbf{x} \in \mathbb{R}^m$ with the parameter λ is defined as [29, 30, 31]

$$\text{prox}_{\lambda f}(\mathbf{x}) = \underset{\mathbf{v}}{\text{argmin}} \frac{1}{2\lambda} \|\mathbf{x} - \mathbf{v}\|^2 + f(\mathbf{v}). \tag{5}$$

The definition depicts that $\text{prox}_{\lambda f}(\mathbf{x})$ tries to find a trade-off between minimizing the function and being close to the point $\mathbf{x}$. The resulting point tends to be close to the minimizer of the function f for a large λ, and close to $\mathbf{x}$ for a small λ [31].

There exist lots of functions that have closed-form proximal operators [31]. Vector norms are examples of such functions. In this paper, the proximal operator of ℓ_1- norm, i.e. $f(\mathbf{x}) = \|\mathbf{x}\|_1$ will be used, which can be easily obtained as [31]

$$\left(\text{prox}_{\lambda\|\cdot\|_1}(\mathbf{x})\right)_i = \begin{cases} x_i - \lambda & x_i > \lambda \\ 0 & |x_i| \leq \lambda \\ x_i + \lambda & x_i < -\lambda \end{cases}, \tag{6}$$

for $i = 1, \ldots, m$. This proximal operator is known as soft thresholding [31], which is denoted by $\text{prox}_{\lambda\|\cdot\|_1}(\mathbf{x}) = \text{soft}(\mathbf{x}, \lambda)$.

3. The Proposed Outlier-robust GL Algorithm

In this section, our proposed GL algorithm, which is robust to outlier data, is presented. The model of graph signals is first modified in order to address the outliers in the data. The problem is then described as a non-convex optimization problem, which is solved be alternating minimization. At the last subsection, the convergence of our iterative method is established.

3.1. Incorporating robustness against outliers into uncompensated GL approach

The minimization problem (4) may not be appropriate in the presence of outlier values in the input data. Thus, for removing the outlier, the following graph signal model is utilized

$$\mathbf{y}_m = \mathbf{x}_m + \hat{\mathbf{o}}_m, \tag{7}$$

where $\mathbf{x}_m$ is the uncontaminated graph signal, $\hat{\mathbf{o}}_m$ is the vector that contains the unknown outlier values, and $\mathbf{y}_m$ is the graph signal contaminated by the outliers. Given $\mathbf{y}_1, \ldots, \mathbf{y}_M$, a similar idea as in [32] is adopted to make the GL algorithm robust to outliers, that is, the GL problem is reformulated as the following optimization problem:

$$\{\mathbf{L}^*, \mathbf{O}^*\} = \underset{\mathbf{L}\in\mathcal{L},\mathbf{O}}{\operatorname{argmin}} \ \frac{1}{2}\sum_{m=1}^{M}(\mathbf{y}_m - \mathbf{o}_m)^T\mathbf{L}(\mathbf{y}_m - \mathbf{o}_m) + \frac{1}{2}f(\mathbf{L}) + g(\mathbf{O}), \quad (8)$$

where $\mathbf{o}_m$'s and $\mathbf{L}$ are the optimization variables, $\mathbf{O} \triangleq [\mathbf{o}_1, \ldots, \mathbf{o}_M]$, ${\mathbf{o}^*}_m$ is the optimal value of $\mathbf{o}_m$, which is the estimation of the true outlier vector $\hat{\mathbf{o}}_m$, $\mathbf{O}^* \triangleq [{\mathbf{o}^*}_1, \ldots, {\mathbf{o}^*}_M]$, and $\mathbf{L}^*$ is the optimal value of $\mathbf{L}$. The first term in (8) enforces the graph signal $\mathbf{y}_m - \mathbf{o}_m$, which is an estimation of $\mathbf{x}_m$, to be smooth over the graph $\mathbf{L}$, for $1 \leq m \leq M$. Similar to (4), the function $f(\mathbf{L})$ is for regularization of the resulting Laplacian matrix $f(\mathbf{L})$. The function $g(\mathbf{O})$ in the last term is to promote the sparsity of the outlier matrix. The above problem can be rewritten in matrix form as

$$\{\mathbf{L}^*, \mathbf{O}^*\} = \underset{\mathbf{L}\in\mathcal{L},\mathbf{O}}{\operatorname{argmin}} \ \frac{1}{2}\operatorname{trace}\Big((\mathbf{Y} - \mathbf{O})^T\mathbf{L}(\mathbf{Y} - \mathbf{O})\Big) + \frac{1}{2}f(\mathbf{L}) + g(\mathbf{O}). \quad (9)$$

Different functions may be utilized for $g(\cdot)$, such as ℓ_1-norm ($\|\mathbf{O}\|_1$), squared ℓ_1-norm ($\frac{1}{2}\|\mathbf{O}\|_1^2$), sum of ℓ_2-norms ($\sum_{m=1}^{M}\|\mathbf{o}_m\|_2$), and sum of squared ℓ_1-norm ($\frac{1}{2}\sum_{m=1}^{M}\|\mathbf{o}_m\|_1^2$). In this paper, ℓ_1-norm and squared ℓ_1-norm are used. As it can be found in the simulations, squared ℓ_1-norm has a better insensitivity to the scale of the input data.

To solve the GL problem with outlier, a BCD (also known as alternating minimization) [33] approach is adopted. The first idea coming into mind is to alternatively minimize the objective function with respect to (w.r.t.) the graph Laplacian matrix $\mathbf{L}$ and the outlier matrix $\mathbf{O}$. Minimizing (9) w.r.t. the Laplacian matrix $\mathbf{L}$ is equivalent to the uncompensated GL in (4) with $\mathbf{X} = \mathbf{Y} - \mathbf{O}$. Next, minimizing w.r.t. $\mathbf{O}$ results in the following minimization problem

$$\underset{\mathbf{O}}{\text{minimize}} \quad \frac{1}{2}\operatorname{trace}\Big((\mathbf{Y} - \mathbf{O})^T\mathbf{L}(\mathbf{Y} - \mathbf{O})\Big) + g(\mathbf{O}), \quad (10)$$

where finding a closed form solution for it is challenging. However, efficient numerical minimization methods can be used for solving the convex program

in (10), such as alternating direction method of multipliers (ADMM) [34], proximal gradient descent [31, 30], or CVX package [35]. Proximal gradient descent method [31, 30] is a good option for the problem of (10) since ℓ_1-norm and squared ℓ_1-norm functions used for $g(\cdot)$ have closed form proximal operators. Additionally, this method has a well-studied convergence analysis [36, 30], and it is faster than the general-purpose tools like CVX.

Proximal gradient is an iterative method, where in each iteration, a gradient step is taken on the smooth part of the objective function (i.e. $\frac{1}{2}\,\mathrm{trace}((\mathbf{Y}-\mathbf{O})^T\mathbf{L}(\mathbf{Y}-\mathbf{O}))$ in (10)), and then, the proximal operator of the non-smooth part (i.e. $g(\mathbf{O})$ in (10)) is calculated at the resulting point. Particularly, outlier matrix in $\{k+1\}$-th iteration, $\mathbf{O}^{\{k+1\}}$, is obtained from the k-th iteration, $\mathbf{O}^{\{k\}}$, as follows

$$\mathbf{O}^{\{k+1\}} = \mathrm{prox}_g\left(\mathbf{O}^{\{k\}} - \eta_k\,\mathbf{L}\big(\mathbf{Y}-\mathbf{O}^{\{k\}}\big)\right), \tag{11}$$

where η_k is the step size and $\mathbf{L}(\mathbf{Y}-\mathbf{O}^{\{k\}})$ is the gradient of the smooth part in (10) [37]. Here, the step size plays an important role in the convergence of the iterative algorithm. Specially, it can be shown that the method converges for a fixed step size $\eta_k = \eta$, when $\mathbf{L}(\mathbf{Y}-\mathbf{O})$ is Lipschitz continuous with constant L and $0 < \eta \leq 1/L$ [31, 30]. Assuming $\lambda_{\max}$ to be the greatest eigenvalue of the Laplacian matrix $\mathbf{L}$, any $L \geq \lambda_{\max}$ is a Lipschitz constant for $\mathbf{L}(\mathbf{Y}-\mathbf{O})$ [31].

The outlier-robust GL algorithm based on the above coordinate descent approach is summarized in Algorithm 1.

However, the focus of this paper is not Algorithm 1. The reasons are: Firstly, the convergence analysis of Algorithm 1 is more tricky than the convergence of the upcoming proposed method. For example, since the Laplacian matrix is not positive definite there is no guarantee that one may always find a unique solution for $\mathbf{O}$ in problem (10). Secondly, there exists no closed form solution for problem (10). Moreover, it requires another iterative algorithm, resulting in an inner loop nested inside the main outer loop. Thus, it can make the main algorithm to be potentially computationally-expensive and time-demanding. Another issue is the introduction of the gradient descent step size parameter η, suggesting the need for being tuned.

To avoid the above mentioned issues, one of the contributions of this paper, another BCD approach for solving (8) is introduced. In this method, instead of minimizing the whole outlier matrix $\mathbf{O}$ as in Algorithm 1, each row of the outlier matrix is separately minimized in each step of the alternating minimization algorithm. More precisely, (9) is firstly rewritten as

Algorithm 1 Outlier-robust GL algorithm based on proximal gradient method

Input: Graph signal matrix $\mathbf{Y} \in \mathbb{R}^{N\times M}$, outlier regularization parameter μ, and the step size of the inner proximal gradient descent method η.

Output: graph Laplacian matrix $\mathbf{L}^*$ and outlier matrix $\mathbf{O}^* \in \mathbb{R}^{N\times M}$

Initialize outlier matrix with all-zero matrix: $\mathbf{O} = \mathbf{0}_{N\times M}$.
repeat
  Update graph Laplacian matrix $\mathbf{L}$ using uncompensated GL in (4).
  repeat
    Update $\mathbf{O}$ using (11).
  until convergence
until convergence
$\mathbf{L}^* = \mathbf{L}$ and $\mathbf{O}^* = \mathbf{O}$

$$\{\mathbf{L}^*, \mathbf{O}^*\} = \underset{\mathbf{L}\in\mathcal{L}, \{\mathbf{o}_{[n]}\}_{n=1}^N}{\operatorname{argmin}} \frac{1}{2}\operatorname{trace}\Big((\mathbf{Y}-\mathbf{O})^T\mathbf{L}(\mathbf{Y}-\mathbf{O})\Big) + \frac{1}{2}f(\mathbf{L}) + g(\mathbf{O}), \quad (12)$$

where the row-vector $\mathbf{o}_{[n]} \in \mathbb{R}^{1\times M}$ is the n-th row of the matrix $\mathbf{O}$, i.e. $\mathbf{O} = [\mathbf{o}_{[1]}^T, \ldots, \mathbf{o}_{[N]}^T]^T$. Then, in iterations of the BCD-based algorithm for solving (12), the blocks $\mathbf{L}$, $\mathbf{o}_{[1]}$, $\ldots$, and $\mathbf{o}_{[N]}$ are updated cyclically. Taking only the n-th row of the outlier matrix as the update variable and fixing the rest results in the convex optimization problem

$$\underset{\mathbf{o}\in\mathbb{R}^{1\times M}}{\text{minimize}} \ \frac{1}{2}\operatorname{trace}\left(\begin{bmatrix}\mathbf{y}_{[n]}-\mathbf{o}\\ \mathbf{Y}_{[\backslash n]}-\mathbf{O}_{[\backslash n]}\end{bmatrix}^T \cdot \begin{bmatrix} l_{nn} & \mathbf{l}_n^T \\ \mathbf{l}_n & \mathbf{L}_{nn}\end{bmatrix} \cdot \begin{bmatrix}\mathbf{y}_{[n]}-\mathbf{o}\\ \mathbf{Y}_{[\backslash n]}-\mathbf{O}_{[\backslash n]}\end{bmatrix}\right) + g(\mathbf{O}), \quad (13)$$

where $\mathbf{y}_{[n]} \in \mathbb{R}^{1\times M}$ is the n-th row of $\mathbf{Y}$, the matrix $\mathbf{L}_{nn} \in \mathbb{R}^{(N-1)\times(N-1)}$ is the submatrix of Laplacian matrix formed by omitting the n-th row and the n-th column of $\mathbf{L}$, the scalar l_{nn} is the n-th diagonal entry of $\mathbf{L}$, the vector $\mathbf{l}_n \in \mathbb{R}^{(N-1)\times 1}$ contains the entries of the n-th column of $\mathbf{L}$ except the n-th one, and the matrices $\mathbf{Y}_{[\backslash n]} \in \mathbb{R}^{(N-1)\times M}$ and $\mathbf{O}_{[\backslash n]} \in \mathbb{R}^{(N-1)\times M}$ are the corrupted graph signal matrix and the outlier matrix, respectively, with their n-th rows being omitted.

The objective function in subproblem (13) can be rewritten w.r.t. $\mathbf{o}_{[n]}$ in the following compact form

$$\mathbf{o}_{[n]}^T = \underset{\mathbf{o}\in\mathbb{R}^M}{\operatorname{argmin}} \ \frac{1}{2}\|\mathbf{o} - \mathbf{b}_{[n]}^T\|_2^2 + g_n(\mathbf{o}), \quad (14)$$

where $\mathbf{b}_{[n]} \triangleq \mathbf{y}_{[n]} - \frac{\mathbf{l}_n^T(\mathbf{Y}_{[\setminus n]} - \mathbf{O}_{[\setminus n]})}{l_{nn}}$, and $g_n(\mathbf{o}) = \mu'\|\mathbf{o}\|_1$ for the ℓ_1-norm, or $g_n(\mathbf{o}) = \frac{\mu'}{2}(\|\mathbf{o}\|_1 + \|\mathbf{O}_{[\setminus n]}\|_1)^2$ for the squared ℓ_1-norm cases, with $\mu' = \frac{\mu}{l_{nn}}$.

Solving subproblem (14) is equivalent to finding the proximal mapping of the function $g_n(\mathbf{o})$ at point $\mathbf{b}_{[n]}^T$. Let $\breve{\mathbf{o}} = \text{prox}_{g_n}(\mathbf{o})$ be the proximal mapping of the function $g_n(\cdot)$ at point $\mathbf{o}$. For the ℓ_1-norm case, i.e. for $g_n(\mathbf{o}) = \mu'\|\mathbf{o}\|_1$, according to Section 2, the proximal operator is $\breve{\mathbf{o}} = \text{soft}(\mathbf{o}, \mu')$, which is the soft-thresholding of the vector $\mathbf{o}$ with threshold μ' [31]. However, finding the proximal mapping for the squared ℓ_1-norm case, i.e. for $g_n(\mathbf{o}) = \frac{\mu'}{2}(\|\mathbf{o}\|_1 + c)^2$, is not as straightforward as the ℓ_1-norm case. The proximal operator of the squared ℓ_1-norm function without constant c (i.e. $\frac{\mu'}{2}\|\mathbf{o}\|_1^2$) can be obtained by Algorithm 4 of [38]. Adopting the same idea, it is possible to obtain a closed-form expression for the proximal operator of $g_n(\mathbf{o}) = \frac{\mu'}{2}(\|\mathbf{o}\|_1 + c)^2$:

Theorem 1. *The proximal operator of the function $g(\mathbf{o}) = \frac{\mu'}{2}(\|\mathbf{o}\|_1 + c)^2$ at point $\mathbf{x} \in \mathbb{R}^M$ is given by an expression presented as Algorithm 2.*

Proof. Refer to Appendix A. □

Remark. Note that Algorithm 2 is not iterative, and it provides a closed-form for the proximal operator $\text{prox}_g(\mathbf{x})$, which is unique and exact.

The final iterative algorithm for minimizing the main objective function (12) is given in Algorithm 3. Each iteration of the algorithm consists of $N+1$ steps. In each step, the graph Laplacian matrix, $\mathbf{L}$, or one of the N rows of the outlier matrix ($\mathbf{o}_{[n]}$) are updated. The updates are performed iteratively until the algorithm converges.

3.2. Convergence Analysis

Algorithm 3 is a BCD method, and it tries to solve the non-differentiable non-convex optimization problem (8) for learning the graph in the presence of outliers. In general, analyzing the convergence of the BCD method for non-convex problems may not be very straightforward. A broad class of non-convex problems have the following objective function form,

$$f(\mathbf{u}_1, \ldots, \mathbf{u}_Q) = f_0(\mathbf{u}_1, \ldots, \mathbf{u}_Q) + \sum_{i=1}^{Q} f_i(\mathbf{u}_i), \tag{15}$$

where $\mathbf{u}_1, \ldots, \mathbf{u}_Q$ are the block coordinates. In this form, $f_0(\cdot)$ is the smooth differential part of $f(\cdot)$, and $f_i(\cdot)$'s contain the non-differentiable parts. Constraints can also be incorporated into (15) by indicator functions. For example, the constraint $\mathbf{L} \in \mathcal{L}$ can be taken into account using the indicator

Algorithm 2 Calculating the proximal operator of $g(\mathbf{o}) = \frac{\mu'}{2}(\|\mathbf{o}\|_1 + c)^2$ at point $\mathbf{x} \in \mathbb{R}^M$

Input: Vector $\mathbf{x} \in \mathbb{R}^M$, constant c, and parameter μ'.

1. Set $\bar{u}_m = |x_m| - \mu' c$ for each $m = 1, \ldots, M$.
2. Sort $\bar{u}_m$'s as $\bar{u}_{(1)} \geq \ldots \geq \bar{u}_{(M)}$.
3. **if** $\bar{u}_{(1)} > 0$ **then**
4. $\quad \rho = \max \left\{ 1 \leq \rho' \leq M | \bar{u}_{(\rho')} > \frac{\mu'}{1+\mu'\rho'} \sum_{m=1}^{\rho'} \bar{u}_{(m)} \right\}$
5. $\quad$ Compute $\breve{x}_m = \text{sign}(x_m) \,.\, \text{soft}(\bar{u}_m, \tau)$ for $m = 1, \ldots, M$, where $\tau = \frac{\mu'}{1+\mu'\rho} \sum_{m=1}^{\rho} \bar{u}_{(m)}$.
6. **else**
7. $\quad \breve{x}_m = 0$ for $m = 1, \ldots, M$.
8. **end if**

Output: Vector $\breve{\mathbf{x}}$, where its m-th element is $\breve{x}_m$.

function $\iota_{\mathcal{L}}(\mathbf{L})$, where

$$\iota_{\mathcal{L}}(\mathbf{L}) = \begin{cases} 0, & \mathbf{L} \in \mathcal{L} \\ +\infty, & \mathbf{L} \notin \mathcal{L} \end{cases}. \tag{16}$$

For Algorithm 1, there are two block coordinates ($Q = 2$), i.e. one for the graph Laplacian $\mathbf{u}_1 = \text{vec}(\mathbf{L})$ and one for the outlier matrix $\mathbf{u}_2 = \text{vec}(\mathbf{O})$, where $\text{vec}(\cdot)$ indicates the vectorization symbol. In this case, $f_0(\mathbf{u}_1, \mathbf{u}_2) = \frac{1}{2} \text{trace}\left((\mathbf{Y} - \mathbf{O})^T \mathbf{L} (\mathbf{Y} - \mathbf{O})\right)$, $f_1(\mathbf{u}_1) = \frac{1}{2} f(\mathbf{L}) + \iota_{\mathcal{L}}(\mathbf{L})$, and $f_2(\mathbf{u}_2) = g(\mathbf{O})$.

On the other hand, for Algorithm 3, the objective function of (12) can be written in form of (15) for $g(\mathbf{O}) = \mu \|\mathbf{O}\|_1$, but not for $g(\mathbf{O}) = \frac{\mu}{2} \|\mathbf{O}\|_1^2$. Particularly, there are $Q = N + 1$ blocks, one for $\mathbf{u}_1 = \text{vec}(\mathbf{L})$ and N for $\mathbf{u}_{n+1} = \mathbf{o}_{[n]}^T$, for $1 \leq n \leq N$. For $g(\mathbf{O}) = \mu \|\mathbf{O}\|_1$:

$$f_0(\mathbf{u}_1, \ldots, \mathbf{u}_{N+1}) = \frac{1}{2} \text{trace}\left((\mathbf{Y} - \mathbf{O})^T \mathbf{L} (\mathbf{Y} - \mathbf{O})\right), \tag{17}$$

$$f_1(\mathbf{u}_1) = \frac{1}{2} f(\mathbf{L}) + \iota_{\mathbf{L}}(\mathcal{L}), \tag{18}$$

$$f_{n+1}(\mathbf{u}_{n+1}) = \mu \|\mathbf{o}_{[n]}\|_1, \quad \text{for } 1 \leq n \leq N. \tag{19}$$

However, it should be noted that the objective function of (12) for $g(\mathbf{O}) = \frac{\mu}{2} \|\mathbf{O}\|_1^2$ cannot be written in the form of (15), since the term $g(\mathbf{O}) = \frac{\mu}{2} \left(\sum_{n=1}^{N} \|\mathbf{o}_{[n]}\|_1 \right)^2$ is not separable with respect to $\mathbf{o}_{[n]}$'s.

The seminal work of Tseng [39] studies the convergence analysis of BCD for non-convex problems with some specific conditions, which are also applicable to the problems of this paper. In this section, the convergence of Algorithm 3 is studied with the help of the results in [39][1]. For the sake of clarity, $Q = N + 1$ blocks at the k-th iteration of the algorithm are gathered in an unknown vector variable $\mathbf{u}^{\{k\}} = \text{vec}\left(\{\mathbf{L}^{\{k\}}, \mathbf{o}_{[1]}^{\{k\}}, \mathbf{o}_{[2]}^{\{k\}}, \ldots, \mathbf{o}_{[N]}^{\{k\}}\}\right)$.

For $g(\mathbf{O}) = \mu\|\mathbf{O}\|_1$, the objective function (denoted by $h_1(\cdot)$) is

$$\begin{aligned} h_1(\mathbf{L}, \{\mathbf{o}_{[n]}\}_{n=1}^N) = &\frac{1}{2}\,\text{trace}\left((\mathbf{Y}-\mathbf{O})^T\mathbf{L}(\mathbf{Y}-\mathbf{O})\right) \\ &+ \frac{1}{2}f(\mathbf{L}) + \iota_{\mathcal{L}}(\mathbf{L}) + \mu\sum_{n=1}^{N}\|\mathbf{o}_{[n]}\|_1. \end{aligned} \tag{20}$$

Lemma 1. *The objective function in* (20) *is* regular *(as defined in [39]), that is, any coordinate-wise minimum point of function $h_1(\cdot)$ is also a stationary point of $h_1(\cdot)$. Here, the definition of a stationary point is, according to [39], a point that moving away from it in any feasible direction results in increasing of the objective function for a sufficiently small neighborhood.*

Proof. According to Lemma 3.1. of [39], if the objective function $f(\cdot)$ is in the separable form of (15), and $f_0(\cdot)$ has an open domain and is Gâteaux-differentiable [40] on its domain, then $f(\cdot)$ is regular.

For (20), the function $f_0(\cdot)$ defined in (17) has an open domain. Moreover, the following operator exists for $f_0(\cdot)$

$$\begin{aligned} df_0(\mathbf{L}, \mathbf{O}; \mathbf{L}', \mathbf{O}') &\triangleq \lim_{t\to 0}\frac{f_0(\mathbf{L}+t\mathbf{L}', \mathbf{O}+t\mathbf{O}') - f_0(\mathbf{L}, \mathbf{O})}{t} \\ &= \frac{1}{2}\,\text{trace}\left((\mathbf{Y}-\mathbf{O})^T\mathbf{L}'(\mathbf{Y}-\mathbf{O})\right) - \text{trace}\left(\mathbf{O}'^T\mathbf{L}(\mathbf{Y}-\mathbf{O})\right), \end{aligned} \tag{21}$$

and is linear with respect to $\mathbf{L}'$ and $\mathbf{O}'$, for any $\mathbf{L} \in \mathcal{L}$, $\mathbf{O}$, $\mathbf{L}' \in \mathcal{L}$, and $\mathbf{O}'$. Thus, $f_0(\cdot)$ is Gâteaux-differentiable on its domain, and consequently, $f(\cdot)$ is regular. □

[1]The convergence analysis of Algorithm 1 would be tricky because the conditions of the theorems of [39] are not satisfied for the minimization problem (9). Thus, it remains as a question for further research.

Algorithm 3 Outlier-robust GL algorithm

Input: Graph signal matrix $\mathbf{Y} \in \mathbb{R}^{N \times M}$ and outlier regularization parameter μ.

Output: graph Laplacian matrix $\mathbf{L}^*$ and outlier matrix $\mathbf{O}^* \in \mathbb{R}^{N \times M}$

Initialize outlier matrix with all-zero matrix: $\mathbf{O} = \mathbf{0}_{N \times M}$.
repeat
  Update graph Laplacian matrix $\mathbf{L}$ using uncompensated GL in (4).
  for $n = 1, \ldots, N$ **do**
    Update $\mathbf{o}_{[n]}$ by solving (14).
  end for
until convergence
$\mathbf{L}^* = \mathbf{L}$ and $\mathbf{O}^* = \mathbf{O}$

Theorem 2. *For $g(\mathbf{O}) = \mu\|\mathbf{O}\|_1$, the cluster points generated by Algorithm 3, i.e. $\{\mathbf{u}^{\{k\}}\}_{k=1,2,\ldots}$, converge to a stationary point.*

Proof. The required conditions mentioned in Theorem 4.1. part (b) of [39] can be satisfied by the objective function in (20). Particularly, the level set $\mathcal{U}^{\{0\}} = \{\mathbf{u} : h_1(\mathbf{u}) \leq h_1(\mathbf{u}^{\{0\}})\}$ is compact, and $h_1(\cdot)$ is continuous on $\mathcal{U}^{\{0\}}$. Moreover, from Lemma 1, $h_1(\cdot)$ is regular at every point $\mathbf{u} \in \mathcal{U}^{\{0\}}$. Thus, based on Theorem 4.1. part (b) of [39], if a set of N blocks, out of $N+1$ blocks, can be found such that the objective function $h_1(\cdot)$ becomes pseudoconvex [41] for every pair of blocks in this set, then every cluster point generated by the BCD method using a cyclic rule (as defined in [39]) converges to a stationary point. In the objective function $h_1(\cdot)$, the set of N blocks $\{\mathbf{o}_{[n]}\}_{n=1}^N$ meets this condition. This is because $h_1(\cdot)$ is convex w.r.t. the outlier matrix $\mathbf{O}$, hence, it would also be pseudoconvex w.r.t. to the pair $(\mathbf{o}_{[i]}, \mathbf{o}_{[j]})$ for any $1 \leq i, j \leq N$ $(i \neq j)$. Consequently, the BCD method introduced in Algorithm 3 for $g(\mathbf{O}) = \frac{\mu}{2}\|\mathbf{O}\|_1$ converges to a stationary point of $h_1(\cdot)$. □

Theorem 3. *For $g(\mathbf{O}) = \frac{1}{2}\|\mathbf{O}\|_1^2$, the cluster points $\{\mathbf{u}^{\{k\}}\}_{k=1,2,\ldots}$ generated by Algorithm 3 converge to a coordinate-wise minimum point of the objective function.*

Proof. In this case, the objective function (denoted by $h_2(\cdot)$) is

$$h_2(\mathbf{L}, \{\mathbf{o}_{[n]}\}_{n=1}^N) = \frac{1}{2}\operatorname{trace}\left((\mathbf{Y}-\mathbf{O})^T\mathbf{L}(\mathbf{Y}-\mathbf{O})\right) + \frac{1}{2}f(\mathbf{L}) + \iota_{\mathcal{L}}(\mathbf{L}) + \frac{\mu}{2}\left(\sum_{n=1}^N \|\mathbf{o}_{[n]}\|_1\right)^2. \tag{22}$$

As stated before, $h_2(\cdot)$ cannot be written in the separable form of (15) due to the squared term, and hence the objective function $h_2(\cdot)$ is not regular. However, based on Theorem 4.1. part (c) of [39], the convergence to a coordinate-wise minimum point still holds. The level set $\mathcal{U}^{\{0\}} = \{\mathbf{u} : h_1(\mathbf{u}) \leq h_1(\mathbf{u}^{\{0\}})\}$ is again compact, and $h_2(\cdot)$ is continuous on this level set. According to the Theorem 4.1. part (c) of [39], if there is *at most one* minimum point for the $N-1$ blocks of $\mathbf{u}_2, \ldots, \mathbf{u}_N$, then every cluster point of the BCD method of Algorithm 3 generated by the cyclic rule is a coordinate-wise minimum point. In the minimization problem (12), we derived closed-form unique solutions for $\mathbf{o}_{[1]}, \ldots, \mathbf{o}_{[N]}$. In other words, we solved the optimization step of (14) in closed form for the squared ℓ_1-norm case, and obtained a unique solution for each of $\mathbf{o}_{[1]}, \ldots, \mathbf{o}_{[N]}$ using Algorithm 2. Thus, there is at most one minimum point for each of $\mathbf{o}_{[n]}$'s, which makes the condition of the mentioned theorem be satisfied, and the cluster points of $\{\mathbf{u}^{\{k\}}\}_{k=1,2,\ldots}$ converge to a coordinate-wise minimum point. □

However, in comparison with Theorem 2, this convergence is weaker because the minimization guarantee only holds for coordinate directions.

4. Numerical Simulations

In this section, the performance of the outlier-robust GL method in Algorithm 3 (OR-GL) is demonstrated in the presence of outlier data. For our simulations, synthetic data is used. This helps us to have the true underlying graph and so the performance of GL algorithms can be assessed. The effects of different outlier ratios and different additive noise levels on the algorithm are presented, and it is shown that outlier data can adversely affect uncompensated GL algorithms. In another experiment, the robustness of the algorithm to different input data scales is demonstrated for squared ℓ_1-norm. In this section, a well-known GL algorithm [11] is chosen as the uncompensated GL (U-GL) algorithm. The accuracy of the obtained graph is

measured by signal to noise ratio (SNR) defined as (this is the same criterion used in [18])

$$\mathrm{SNR}_{\mathbf{L}} = 10 \log_{10} \frac{\|\tilde{\mathbf{L}}^{(\mathrm{true})}\|_F^2}{\|\tilde{\mathbf{L}}^{(\mathrm{true})} - \tilde{\mathbf{L}}\|_F^2}, \tag{23}$$

where $\tilde{\mathbf{L}}^{(\mathrm{true})}$ and $\tilde{\mathbf{L}}$ are the true and the estimated normalized Laplacian matrices, respectively. The normalization is done by dividing the Laplacian matrix by its trace, i.e.

$$\tilde{\mathbf{L}} \triangleq \frac{\mathbf{L}}{\mathrm{trace}(\mathbf{L})}. \tag{24}$$

The procedure of generating synthetic data begins with creating a random graph as the true underlying graph. The function `GSP_sensor` in `GSPBox` toolbox [42] is used for this purpose. Throughout the simulations, we use $N = 30$ number of nodes for the generated graphs. For producing smooth graph signals over the resulted graph, white noise vectors $\mathbf{w} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ are graph low-pass filtered [1], where $\mathbf{I} \in \mathbb{R}^{N \times N}$ is the identity matrix. More precisely, the procedure is as follows:

1. Calculate the GFT of the white noise, i.e. $\widehat{\mathbf{w}} = \mathbf{V}^T \mathbf{w}$.
2. Suppress the high-frequency coefficients of $\widehat{\mathbf{w}}$ by point-wise multiplying it by the GFT of a low-pass filter, i.e. $\widehat{\boldsymbol{\psi}}$. Specifically, the GFT coefficients of the resulting smooth graph signal are calculated as $\widehat{x}_n^{(\mathrm{smooth})} = \widehat{\psi}_n . \widehat{w}_n$ for $n = 1, \ldots, N$.
3. Calculate the inverse GFT of the modified coefficients, i.e. $\mathbf{x}^{(\mathrm{smooth})} = \mathbf{V}\widehat{\mathbf{x}}^{(\mathrm{smooth})}$.

As the low-pass filter $\widehat{\psi}_n$, the same filter as in [10] is used, that is,

$$\widehat{\psi}_n = \begin{cases} 1/\sqrt{\lambda_n} & \lambda_n \neq 0 \\ 0 & \lambda_n = 0 \end{cases}. \tag{25}$$

For the additive noise, a white Gaussian noise vector $\mathbf{v} \sim \mathcal{N}(\mathbf{0}, \sigma_v^2 \mathbf{I})$ is added to each smooth graph signal $\mathbf{x}^{(\mathrm{smooth})}$,

$$\mathbf{x} = \mathbf{x}^{(\mathrm{smooth})} + \mathbf{v}. \tag{26}$$

The SNR of the input signal is calculated as

$$\mathrm{SNR}_{\mathrm{in}} = 10 \log_{10} \frac{\mathbb{E}\{\|\mathbf{x}^{(\mathrm{smooth})}\|_2^2\}}{\mathbb{E}\{\|\mathbf{v}\|_2^2\}} = \frac{\mathbb{E}\{\|\mathbf{x}^{(\mathrm{smooth})}\|_2^2\}}{N\sigma_v^2}, \tag{27}$$

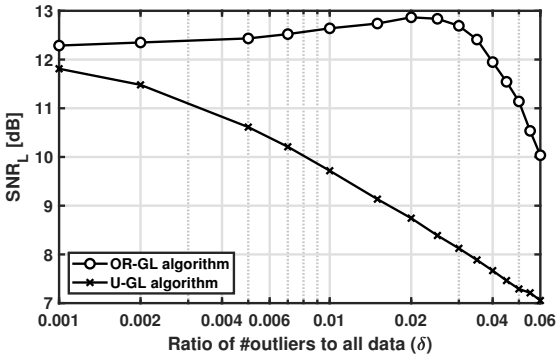

Figure 1: GL performance of the proposed outlier-robust GL (OR-GL) and the uncompensated GL [11] (U-GL) algorithms for different ratios of outlier data δ with $\text{SNR}_{\text{in}} = 10$ dB.

where $\mathbb{E}\{\cdot\}$ is the expectation symbol.

Then, all graph signals are gathered as the columns of the matrix $\mathbf{X} \in \mathbb{R}^{N\times M}$. For all the simulations, we generate $M = 1000$ number of graph signals. For simulating the outlier effect on the data, an outlier value is added to each entry of the matrix $\mathbf{X}$ with probability δ, named as *outlier ratio*. These outlier values are sampled from a Gaussian distribution with mean $\eta_o = 5\,\sigma_x$ and standard deviation $\sigma_o = 0.2\,\sigma_x$, where σ_x is the standard deviation of all entries of the data matrix $\mathbf{X}$. The sign of the outlier is randomly selected with equal probability. After imposing the outlier values to the matrix $\mathbf{X}$, the matrix $\mathbf{Y}$, which contains the graph signals corrupted by the outlier values, is obtained.

The only parameter in Algorithm 3 is the outlier regularization coefficient μ, which could be set using cross validation. In our simulations, to tune the parameter μ, graph signals are generated from a set of known graphs for the case $\delta = 0.01$ and $\text{SNR}_{\text{in}} = 10$ dB, and the best value is chosen with the grid search. Then, this value is used for different outlier ratios and input SNRs in all simulations of this section. As the result, for $g(\mathbf{O}) = \mu\|\mathbf{O}\|_1$, $\mu = 1$ is chosen, and for $g(\mathbf{O}) = \frac{\mu}{2}\|\mathbf{O}\|_1^2$, μ is set equal to 10^{-4}. The stopping criterion for the algorithms is to run them for 50 iterations. In practice, this number of iterations is large enough to reach the convergence. Each point in the figures is the average of 1000 independent realizations.

Fig. 1 shows the performance of GL for different outlier ratios in the range $0.001 \leq \delta \leq 0.06$. The proposed method is compared with the uncompen-

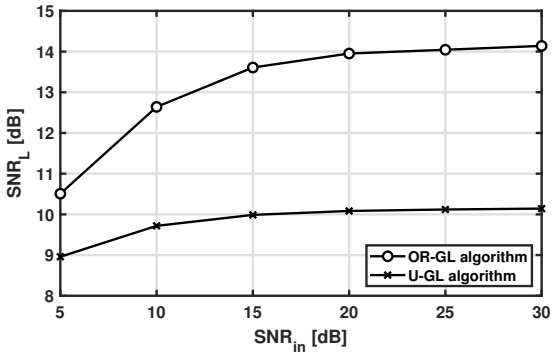

Figure 2: GL performance of the proposed outlier-robust GL (OR-GL) and the uncompensated GL [11] (U-GL) algorithms for different values of SNR_{in} with $\delta = 0.01$.

sated GL algorithm of [11] (U-GL). In the lower outlier ratios, e.g. $\delta = 0.001$, the $\text{SNR}_{\mathbf{L}}$ for the U-GL algorithm is close to the OR-GL algorithm. Moving to higher δ's, $\text{SNR}_{\mathbf{L}}$ decreases for U-GL, while OR-GL has almost the same $\text{SNR}_{\mathbf{L}}$ level until[2] $\delta = 0.04$. Thus, the sensitivity to outlier data is easily seen for the U-GL algorithm. By more increasing the outlier ratio, a fast performance drop occurs in OR-GL, and the reason is that the sparsity assumption for the outlier-corrupted entries is no longer valid.

Fig. 2 shows $\text{SNR}_{\mathbf{L}}$ versus SNR_{in}, while the ratio of outlier is kept fixed at $\delta = 0.01$. For the different amounts of additive noise, OR-GL has a higher $\text{SNR}_{\mathbf{L}}$ than U-GL.

In the next experiment, the GL performances are compared for two cases of outlier regularization functions: 1) $g(\mathbf{O}) = \|\mathbf{O}\|_1$ and 2) $g(\mathbf{O}) = \frac{1}{2}\|\mathbf{O}\|_1^2$, for different scales of the input data. The previously generated data is multiplied by a scale factor ranging from 10^{-3} to 10^3 in logarithmic steps, for $\text{SNR}_{\text{in}} = 10$ dB and $\delta = 0.01$. The ℓ_1-norm penalty term performs reliably for the scales 0.8 to 20. However, squared ℓ_1-norm penalty term maintains its own best performance for the whole range. This is because the term $\|\mathbf{O}\|_1^2$ scales with

[2]The subtle increase of $\text{SNR}_{\mathbf{L}}$ despite the increase of the outlier ratio δ from 0.001 to 0.02 is partially related to the effect of outlier regularization coefficient μ. In fact, since μ is tuned for the case $\delta = 0.01$, the performance around this outlier ratio is higher than the low ratios, and the maximum $\text{SNR}_{\mathbf{L}}$ in $\delta = 0.02$ indicates that the value $\mu = 1$ is optimum for $\delta = 0.02$. However, as it can be seen in Fig. 1, this effect is subtle, that is, the $\text{SNR}_{\mathbf{L}}$ curve is almost constant for $\delta \leq 0.03$, and has a sudden drop for $\delta > 0.03$.

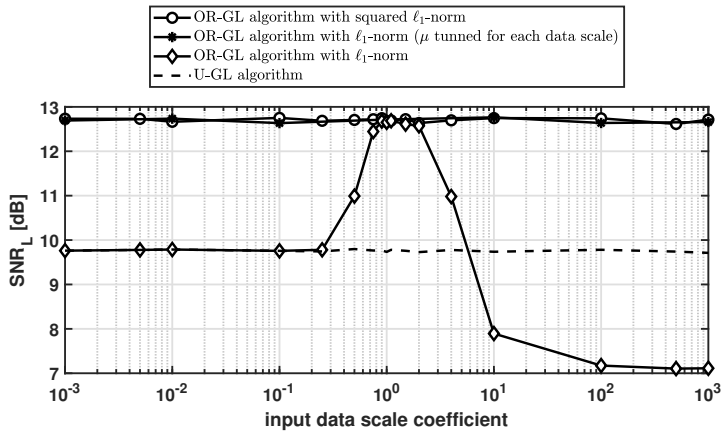

Figure 3: GL performance of the uncompensated GL algorithm of [11] and the proposed algorithm based on ℓ_1-norm and squared ℓ_1-norm for different scales of the input data. Moreover, the performance of the algorithm with ℓ_1-norm is also added when the coefficient μ is tuned for each data scale separately. The outlier ratio is $\delta = 0.01$ and the input SNR equals $\text{SNR}_\text{in} = 10$ dB.

power two, the same order of the first term in (12), i.e. quadratic function. However, the term $\|\mathbf{O}\|_1$ changes linearly w.r.t. the data scale. This means that the ℓ_1-norm and the quadratic function terms scale with different orders, and so the coefficient μ is not well-tuned for the whole range of scales. To validate this for ℓ_1-norm case, another curve (with star marks) has been included in Fig. 3 to see the effect of adjusting the regularization parameter μ separately for different input data scales. Particularly, μ is grid searched for a set of data scales, and the higher GL performance is reported for each data scale. As it is expected, the curve for ℓ_1-norm case with μ tuned for each data scale shows the similar performance as the squared ℓ_1-norm case. However, the algorithm with the squared ℓ_1-norm regularization term does not need to be tuned for each data scale separately.

5. Conclusion

In this paper, an algorithm for smoothness-based graph learning problem in the presence of outlier data was proposed by incorporating ℓ_1 and squared ℓ_1 norms to the objective function of GL. Precisely, a novel BCD minimization approach was introduced to solve the resulted non-convex optimization problem. Besides its low computational advantage, it provides a

straightforward way to establish the convergence of the iterative algorithm. Numerical simulations demonstrated that uncompensated GL methods are sensitive to the existence of outlier data in the input graph signals. However, the proposed method is capable of robustifying GL algorithms to outliers. Specifically, the performance of the outlier-robust GL does not drop until a certain amount of outliers, after which it exhibits a rapid performance drop. This rapid drop of performance is because our main assumption of outlier sparseness does not hold anymore. It was also shown that the squared ℓ_1-norm version of the algorithm is more robust to the scale of the data in comparison to the simple ℓ_1-norm. The expense of this good property is that calculating the proximal operator of the squared ℓ_1-norm is considerably harder that the ℓ_1-norm.

Appendix A. Proof of Theorem 1

In this appendix, a similar approach as [38] is adopted, except that there is a constant c in the squared ℓ_1-norm function. Before going to the proof, a few lemmas are stated.

Lemma 2. *Consider the following minimization problem for a given vector* $\mathbf{x} \in \mathbb{R}^N$,

$$\mathbf{p}_1(\mathbf{x}) = \underset{\mathbf{v}\in\mathbb{R}^M}{\operatorname{argmin}} \frac{1}{2}\|\mathbf{x}-\mathbf{v}\|^2 + \frac{\mu'}{2}(\|\mathbf{v}\|_1 + c)^2. \tag{P1}$$

If the solution for (P1) *is denoted by* $\mathbf{p}_1(\mathbf{x})$*, then the following statements are true for* $\mathbf{p}_1(\mathbf{x})$*:*

a) If $\mathbf{x} \succcurlyeq \mathbf{0}$*, then* $\mathbf{p}_1(\mathbf{x}) \succcurlyeq \mathbf{0}$*, where* $\succcurlyeq$ *denotes element-wise greater than or equal.*

b) Let $\boldsymbol{\alpha} \in \{\pm 1\}^M$ *be an arbitrary non-zero sign vector, then* $\mathbf{p}_1(\boldsymbol{\alpha}\odot\mathbf{x}) = \boldsymbol{\alpha}\odot\mathbf{p}_1(\mathbf{x})$*, where* $\odot$ *denotes element-wise (Hadamard) product of two vectors.*

Proof. Refer to the proof of Lemma 9 in [38]. □

Lemma 3. *Consider the following constrained minimization problem*

$$\begin{aligned} \mathbf{p}_2(\mathbf{x}) = \underset{\mathbf{v}}{\operatorname{argmin}} \quad & \frac{1}{2}\|\mathbf{x}-\mathbf{v}\|^2 + \frac{\mu'}{2}(\mathbf{v}^T\mathbf{1} + c)^2 \\ \text{s.t.} \quad & \mathbf{v} \succcurlyeq \mathbf{0}, \end{aligned} \tag{P2}$$

where $\mathbf{p}_2(\mathbf{x})$ *is the solution of* (P2)*. Then, the following equation relates the solutions of* (P1) *and* (P2)

$$\mathbf{p}_1(\mathbf{x}) = \mathbf{p}_2(\mathbf{x}_+) \odot \mathrm{sg}(\mathbf{x}), \tag{A.1}$$

where $\mathrm{sg}(\mathbf{x}) \in \{\pm 1\}^M$ *is a vector containing the signs*[3] *of the entries of* $\mathbf{x}$*, i.e.* $\big(\mathrm{sg}(\mathbf{x})\big)_i = \mathrm{sign}(x_i)$*, and* $\mathbf{x}_+ \in \mathbb{R}_+^M$ *is a vector of absolute values of the entries of* $\mathbf{x}$*, resulting in* $\mathbf{x} = \mathrm{sg}(\mathbf{x}) \odot \mathbf{x}_+$*.*

Proof. Since $\mathbf{x}_+ \succcurlyeq \mathbf{0}$, Lemma 2.a ensures that $\mathbf{p}_1(\mathbf{x}_+) \succcurlyeq \mathbf{0}$. Therefore, $\mathbf{p}_1(\mathbf{x}_+)$ is also the solution of the following constrained minimization problem

$$\begin{aligned} \underset{\mathbf{v}}{\text{argmin}} \quad & \frac{1}{2}\left\|\mathbf{x}_+ - \mathbf{v}\right\|^2 + \frac{\mu'}{2}(\|\mathbf{v}\|_1 + c)^2 \\ \text{s.t.} \quad & \mathbf{v} \succcurlyeq \mathbf{0}. \end{aligned} \tag{A.2}$$

Then, the ℓ_1-norm in the second term can be replaced by $\mathbf{v}^T\mathbf{1}$ due to the $\mathbf{v} \succcurlyeq \mathbf{0}$ constraint, yielding $\mathbf{p}_1(\mathbf{x}_+) = \mathbf{p}_2(\mathbf{x}_+)$. Additionally, from Lemma 2.b, it can be obtained that $\mathbf{p}_1(\mathbf{x}_+) = \mathrm{sg}(\mathbf{x}) \odot \mathbf{p}_1(\mathbf{x})$, which results in (A.1). □

Lemma 4. *Define* $\mathbf{v}^* \triangleq \mathbf{p}_2(\mathbf{x})$ *and* $\bar{\mathbf{u}} \triangleq \mathbf{x}_+ - \mu' c\,\mathbf{1}$ *(i.e.* $\bar{u}_m \triangleq |x_m| - \mu' c$ *for* $1 \le m \le M$*). Then, for problem* (P2)*,* $\bar{u}_j \le \bar{u}_i$ *results in* $v_j^* \le v_i^*$*, meaning that* v_m^**'s are arranged in the same descending order as* $\bar{u}_m$*'s.*

Proof. Karush–Kuhn–Tucker (KKT) conditions [33, 43] for (P2) are written as

$$\mathbf{v}^* - \bar{\mathbf{u}} + (\mu' \mathbf{1}^T \mathbf{v}^*)\,\mathbf{1} = \boldsymbol{\lambda}, \tag{A.3}$$

$$\mathbf{v}^* \succcurlyeq \mathbf{0}, \tag{A.4}$$

$$\boldsymbol{\lambda} \succcurlyeq \mathbf{0}, \tag{A.5}$$

$$\boldsymbol{\lambda}^T \mathbf{v}^* = 0, \tag{A.6}$$

where (A.6) is the complementary slackness condition, Lagrangian multipliers are gathered in the vector $\boldsymbol{\lambda}$, and equation $\bar{\mathbf{u}} = \mathbf{x}_+ - \mu' c\,\mathbf{1}$ is used in (A.3). From $\bar{u}_j \le \bar{u}_i$ and (A.3), it can be found that $\lambda_j - \lambda_i \ge v_j^* - v_i^*$. Now assume by contradiction that $v_j^* > v_i^*$, then $\lambda_j - \lambda_i > 0$, and also $v_j^* > 0$ because of $v_i^* \ge 0$. However, due to the complementary slackness condition in (A.6), letting $v_j^* > 0$ makes $\lambda_j = 0$ and results in $\lambda_i < 0$, which is in contradiction with non-negativity of λ_i as in (A.5). □

[3]The sign of zero entries can be chosen arbitrary negative or positive.

Proof of Theorem 1. The proximal operator of the function $g(\mathbf{o}) = \frac{\mu'}{2}(\|\mathbf{o}\|_1 + c)^2$ at point $\mathbf{x} \in \mathbb{R}^M$ is the solution of (P1). However, it is easier to solve (P2) instead of (P1) since there is no need to deal with the non-differentiable ℓ_1-norm function. Afterwards, the solution of (P1) can be obtained using Lemma 3. Let ${v^*}_m$'s and $\bar{u}_m$'s are sorted as ${v^*}_{(1)} \geq \ldots \geq {v^*}_{(M)}$ and $\bar{u}_{(1)} \geq \ldots \geq \bar{u}_{(M)}$, respectively. According to Lamma 4, equation (A.3) also holds for the sorted versions of ${v^*}_m$'s and $\bar{u}_m$'s, i.e.

$$ {v^*}_{(m)} - \bar{u}_{(m)} + \mu' \sum_{i=1}^{M} {v^*}_{(i)} = \lambda_{(m)} \tag{A.7} $$

for $1 \leq m \leq M$, where $\lambda_{(m)}$ is the corresponding Lagrangian multiplier for ${v^*}_{(m)}$.

Solving (P2) is performed for two separate cases. In the first case, it is supposed that all constraints in (P2) for v^*_m's are binding, i.e. $\mathbf{v}^* = \mathbf{0}$. According to (A.3) and (A.5), all $\bar{u}_m$'s are non-positive. Thus, if $\bar{u}_{(1)} \leq 0$, the solution of (P2) is the all-zero vector, i.e. $\mathbf{p}_2(\mathbf{x}) = \mathbf{v}^* = \mathbf{0}$.

In the second case, it is assumed without loss of generality that ρ number of constraints for ${v^*}_m$'s are non-binding, i.e. ${v^*}_{(1)} \geq \ldots \geq {v^*}_{(\rho)} > 0$ and ${v^*}_{(\rho+1)} = \ldots = {v^*}_{(M)} = 0$. Summing (A.7) for $m = 1, \ldots, \rho$ and using (A.6), it can be obtained that

$$ \tau \triangleq \mu' {\mathbf{v}^*}^T \mathbf{1} = \frac{\mu' \sum_{m=1}^{\rho} \bar{u}_{(m)}}{1 + \rho\mu'}, \tag{A.8} $$

in which, the equality $\sum_{m=1}^{\rho} {v^*}_{(m)} = \sum_{m=1}^{M} {v^*}_m$ is utilized.

Since $\bar{u}_{(m)} > \tau$ for $1 \leq m \leq \rho$, and $\bar{u}_{(m)} \leq \tau$ for $\rho + 1 \leq m \leq M$, ρ can be found as $\rho = \max\left\{m | \bar{u}_{(m)} > \tau\right\}$. Finally, from (A.3), it is obtained that $\mathbf{p}_2(\mathbf{x}) = \mathbf{v}^* = \text{soft}(\bar{\mathbf{u}}, \tau)$, where τ is defined as in (A.8), and then $\mathbf{p}_1(\mathbf{x})$ can be found using (A.1) in Lemma 3. This concludes the proof of Theorem 1. □

References

[1] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, 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. 30 (2013) 83–98.

[2] A. Sandryhaila, J. M. F. Moura, Discrete signal processing on graphs, IEEE Trans. Signal Process. 61 (2013) 1644–1656.

[3] A. Ortega, P. Frossard, J. Kovacevic, J. M. F. Moura, P. Vandergheynst, Graph signal processing: Overview, challenges, and applications, Proc. IEEE 106 (5) (2018) 808–828. doi:10.1109/jproc.2018.2820126.

[4] X. Dong, D. Thanou, M. Rabbat, P. Frossard, Learning graphs from data: A signal representation perspective, IEEE Signal Process. Mag. 36 (2019) 44–63.

[5] G. Mateos, S. Segarra, A. G. Marques, A. Ribeiro, Connecting the dots: Identifying network structure via graph signal processing, IEEE Signal Process. Mag. 36 (3) (2019) 16–43. doi:10.1109/msp.2018.2890143.

[6] D. Koller, N. Friedman, Probabilistic graphical models: Principles and techniques, MIT Press, Cambridge, MA, 2009.

[7] A. P. Dempster, Covariance selection, Biometrics (1972) 157–175.

[8] J. Friedman, T. Hastie, R. Tibshirani, Sparse inverse covariance estimation with the graphical lasso, Biostatistics 9 (3) (2008) 432–441.

[9] B. Lake, J. Tenenbaum, Discovering structure by learning sparse graphs, in: Proc. 32nd Annu. Meeting Cogn. Sci. Soc. (CogSci'10), Portland, OR, 2010, pp. 778–784.

[10] X. Dong, D. Thanou, P. Frossard, P. Vandergheynst, Learning laplacian matrix in smooth graph signal representations, IEEE Trans. Signal Process. 64 (2016) 6160–6173.

[11] V. Kalofolias, How to learn a graph from smooth signals, in: Proc. Int. Conf. Artif. Intell. Statist. (AISTATS'16), Vol. 51, Cadiz, Spain, 2016, pp. 920–929.

[12] D. Thanou, D. I. Shuman, P. Frossard, Parametric dictionary learning for graph signals, in: IEEE Global Conf. Signal Inf. Process. (GlobalSIP'13), Austin, TX, 2013, pp. 487–490.

[13] H. P. Maretic, D. Thanou, P. Frossard, Graph learning under sparsity priors, in: IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP'17), New Orleans, LA, 2017, pp. 6523–6527.

[14] S. Segarra, A. G. Marques, G. Mateos, A. Ribeiro, Network topology inference from spectral templates, IEEE Trans. Signal Inform. Process. Netw. 3 (2017) 467–483.

[15] J. Ying, J. V. M. Cardoso, D. Palomar, Nonconvex sparse graph learning under laplacian constrained graphical model, in: Advances in Neural Information Processing Systems (NeurIPS'20), 2020, pp. 7101–7113.

[16] P. Berger, G. Hannak, G. Matz, Efficient graph learning from noisy and incomplete data, IEEE Trans. Signal Inf. Process. Netw. 6 (2020) 105–119.

[17] H. P. Maretic, P. Frossard, Graph laplacian mixture model, arXiv:1810.10053v1 [cs.LG] (Oct. 2018).

[18] H. Araghi, M. Sabbaqi, M. Babaie-Zadeh, K-Graphs: An algorithm for graph signal clustering and multiple graph learning, IEEE Signal Process. Lett. 26 (10) (2019) 1486–1490. doi:10.1109/LSP.2019.2936665.

[19] D. Hallac, S. Vare, S. Boyd, J. Leskovec, Toeplitz inverse covariance-based clustering of multivariate time series data, in: ACM Int. Conf. Knowl. Discovery Data Mining (SIGKDD'17), Halifax, Canada, 2017, pp. 215–223. doi:10.1145/3097983.3098060.

[20] H. Araghi, M. Babaie-Zadeh, S. Achard, Dynamic k-graphs: An algorithm for dynamic graph learning and temporal graph signal clustering, in: 28th Eur. Signal Proc. Conf. (EUSIPCO'20), 2021, pp. 2195–2199.

[21] R. Mirzaeifard, V. C. Gogineni, N. K. D. Venkategowda, S. Werner, Dynamic graph topology learning with non-convex penalties, in: 30th Eur. Signal Proc. Conf. (EUSIPCO'22), 2022, pp. 682–686.

[22] A. Natali, E. Isuf, M. Coutino, G. Leus, Learning time-varying graphs from online data, arXiv:2110.11017v2 [cs.LG] (Oct. 2022).

[23] Z. Kang, H. Pan, S. C. H. Hoi, Z. Xu, Robust graph learning from noisy data, IEEE Trans. Cyber. 50 (5) (2020) 1833–1843.

[24] N. Shahid, V. Kalofolias, X. Bresson, M. Bronstein, P. Vandergheynst, Robust principal component analysis on graphs, in: Proc. IEEE Int. Conf. Comput. Vis. (ICCV'15), Santiago, Chile, 2015, pp. 2812–2820.

[25] R. Gribonval, M. Nielsen, Sparse representations in unions of bases, IEEE Trans. Inf. Theory 49 (12) (2003) 3320–3325.

[26] D. L. Donoho, Compressed sensing, IEEE Trans. Inf. Theory 52 (4) (2006) 1289–1306.

[27] E. J. Candes, M. B. Wakin, An introduction to compressive sampling, IEEE Signal Proc. Mag. 25 (2) (2008) 21–30.

[28] R. T. Rockafellar, Convex Analysis, Princeton University Press., Princeton, NJ, 1970.

[29] N. Komodakis, J. C. Pesquet, Playing with duality: An overview of recent primal-dual approaches for solving large-scale optimization problems, IEEE Signal Process. Mag. 32 (6) (2015) 31–54.

[30] P. L. Combettes, J. C. Pesquet, Proximal splitting methods in signal processing, in: Springer Optimization and Its Applications, Springer New York, 2011, pp. 185–212.

[31] N. Parikh, S. Boyd, Proximal algorithms, Foundations and Trends in Optimization 1 (3) (2014) 127–239. doi:10.1561/2400000003.

[32] G. Mateos, G. B. Giannakis, Robust PCA as bilinear decomposition with outlier-sparsity regularization, IEEE Trans. Signal Process. 60 (2012) 5176–5190.

[33] D. P. Bertsekas, Nonlinear programming, 2nd Edition, Athena Scientific, 1999, Ch. 2, pp. 267–270.

[34] S. Boyd, N. Parikh, E. Chu, B. Peleato, J. Eckstein, Distributed optimization and statistical learning via the alternating direction method of multipliers, Foundations and Trends in Machine Learning 3 (1) (2011) 1–122. doi:10.1561/2200000016.

[35] M. Grant, S. Boyd, CVX: Matlab software for disciplined convex programming, version 2.1, `http://cvxr.com/cvx` (March 2014).

[36] P. L. Combettes, V. R. Wajs, Signal recovery by proximal forward-backward splitting, Multiscale Model. Simul. 4 (4) (2005) 1168–1200. doi:10.1137/050626090.

[37] K. B. Petersen, M. S. Pedersen, The matrix cookbook (Nov. 2012).

[38] A. F. T. Martins, N. A. Smith, , E. P. Xing, P. M. Q. Aguiar, M. A. T. Figueiredo, Online multiple kernel learning for structured prediction, arXiv:1010.2770v1 [stat.ML] (Oct. 2018).

[39] P. Tseng, Convergence of a block coordinate descent method for nondifferentiable minimization, Journal of Optimization Theory and Applications 109 (3) (2001) 475–494. doi:10.1023/a:1017501703105.

[40] Y. Benyamini, J. Lindenstrauss, Geometric Nonlinear Functional Analysis, Vol. 1, American Mathematical Society, Providence, RI, 2000, Ch. 4, pp. 83–98.

[41] O. L. Mangasarian, Pseudo-convex functions, J. Soc. Ind. and Appl. Math. 3 (2) (1965) 281–290.

[42] N. Perraudin, J. Paratte, D. Shuman, L. Martin, V. Kalofolias, P. Vandergheynst, D. K. Hammond, GSPBOX: A toolbox for signal processing on graphs, arXiv:1408.5781v2 [cs.IT] (Aug. 2014).
URL `https://epfl-lts2.github.io/gspbox-html/index.html`

[43] J. Nocedal, S. J. Wright, Numerical Optimization, Springer, New York, NY, 2006.