A FAST PROXIMAL POINT ALGORITHM FOR GENERALIZED GRAPH LAPLACIAN LEARNING

Zengde Deng *Anthony Man-Cho So*

Department of Systems Engineering and Engineering Management, CUHK, Hong Kong

ABSTRACT

Graph learning is one of the most important tasks in machine learning, statistics and signal processing. In this paper, we focus on the problem of learning the generalized graph Laplacian (GGL) and propose an efficient algorithm to solve it. We first fully exploit the sparsity structure hidden in the objective function by utilizing soft-thresholding technique to transform the GGL problem into an equivalent problem. Moreover, we propose a fast proximal point algorithm (PPA) to solve the transformed GGL problem and establish the linear convergence rate of our algorithm. Extensive numerical experiments on both synthetic data and real data demonstrate that the soft-thresholding technique accelerates our PPA method and PPA can outperform the current state-of-the-art method in terms of speed.

Index Terms— Graph learning, generalized graph Laplacian, augmented Lagrangian, sparsity structure, linear convergence

1. INTRODUCTION

Graphs are mathematical structures that are composed of vertices and edges. In machine learning, statistics and signal processing areas, graphs are often used to analyze the structure of high-dimensional data. It assigns scalars to nodes and edges to form weighted graphs where nodes can represent the objects of interest and edges represent the relationships between objects. These weighted graphs arise from different fields such as brain networks [2], image and video coding [23] and also have many applications in transportation, sensor networks [5], and so on. For instance, in operations research, we usually use weighted graphs to define problems such as shortest path and network flow [8]. In image processing, the correlation of neighboring pixel values can be represented by weighted graphs [10]. Moreover, the graph-based model is widely used in machine learning areas such as semi-supervised learning [26], where the goal is to classify data using a few labeled samples. We refer the reader to the recent surveys [12, 4] for other examples and applications.

In this paper, we focus on the problem of learning the *generalized graph Laplacian* (GGL). The basic goal of graph learning is to optimize a weighted graph with a given structure based on observed data. Given a data matrix C (e.g. empirical covariance) and a penalty matrix H, we aim to minimize the following objective function:

$$\operatorname{tr}(CX) - \log\det(X) + \|H \circ X\|_1, \tag{1}$$

where $\circ$ is the element-wise product and $\|\cdot\|_1$ is the element-wise l_1 penalty. The first two terms give a log-determinant divergence whose minimizer is the maximum likelihood estimator of the inverse covariance matrix, while the last term is a sparse regularization term that controls the sparsity of the desired graph.

In our problem, we consider specific *Laplacian structural constraints* that encode the information about the graph. Graph Laplacian matrices have many applications such as spectral partitioning [13], text mining [11], and filtering [15]. We consider the following generalized graph Laplacian [6] for a given connectivity matrix A (all elements are 0 or 1):

$$\mathcal{Q}(A) = \left\{ X \succ 0 \,\middle|\, \begin{matrix} X_{ij} \le 0 \text{ if } A_{ij} = 1 \\ X_{ij} = 0 \text{ if } A_{ij} = 0 \end{matrix} \text{ for } i \neq j \right\}. \tag{2}$$

Thus, our GGL problem is given as follows:

$$\min_{X \in \mathcal{Q}(A)} \operatorname{tr}(CX) - \log\det(X) + \|H \circ X\|_1, \tag{O}$$

where $H = \alpha(\mathbf{I} - \mathbf{1}\mathbf{1}^T)$ denotes the off-diagonal l_1 penalty, $\mathbf{I}$ and $\mathbf{1}$ are the identity matrix and all ones vector respectively.

There are other ways to identify the graph matrix of a graph model. A problem that is closely related to GGL is sparse inverse covariance estimation (SICE). The most well-known algorithm to solve this problem is the graphical Lasso proposed by Friedman [7]. Subsequently, there are other algorithms for solving this problem such as PPA [22], ALM [20], and QUIC [9]. The main difference between inverse covariance estimation and our GGL problem lies in that the former allows both negative and positive edge weights, while we focus on graphs with nonnegative edge weights due to the nature of the Laplacian matrix [6]. It has been shown in [6] that when signals/data have the property that each entry can be estimated by the nonnegative linear combination of other entries, GGL can provide more accurate graph estimation than covariance inverse estimation.

Recently, there is a line of works that focus on graph Laplacian problems such as GGL, combinatorial graph Lapalcian (CGL), and diagonally dominant graph Laplacian (DDGL). For the GGL problem, Slawaski [21] proposed a primal algorithm that applies to symmetric M-matrices. Pavez [16] constructed an efficient dual block coordinate descent (BCD) algorithm by analyzing the KKT conditions of the GGL problem. Egilmez [6] later extended this method to build a general framework of BCD, which can handle GGL, CGL, and DDGL problems. In this paper, we will compare with this BCD method on the GGL problem. The most time-consuming part of the BCD algorithm lies in solving a quadratic programming (QP) subproblem, which can be expensive in large-scale settings. Note that in both [16] and [6], the authors only showed the convergence of their algorithms but no convergence rates were provided.

We propose an efficient algorithm to solve the GGL problem in this paper and our major contributions are two-fold: 1) We use the soft-thresholding technique to transform the GGL problem into an equivalent problem that exploits the sparsity structure of the objective function. Moreover, extensive numerical experiments demonstrate that this transformation speeds up the algorithm as it results in more sparsity constraints. 2) We propose a fast proximal point algorithm (PPA) to solve the transformed GGL problem. In the inner iteration, we use the preconditioned conjugate gradient (PCG) to solve the subproblem inexactly. Furthermore, we establish the linear convergence of our PPA method.

Due to limited space, we defer the proofs of the lemmas and theorems to the full version of our paper.

2. SOFT-THRESHOLDING TECHNIQUE

We introduce the soft-thresholding function

$$(C_\alpha)_{ij} = \begin{cases} C_{ij}, & i = j, \\ C_{ij} - \alpha, & \text{if } i \neq j \text{ and } C_{ij} > \alpha, \\ 0, & \text{otherwise,} \end{cases} \tag{3}$$

and set $G = \{(i,j) | (C_\alpha)_{ij} = 0\}$. Theorem 1 shows that the original problem (O) can be transformed into the equivalent problem (4) via the soft-thresholding function (3).

Theorem 1. *The following problem*

$$\begin{aligned} \min_{X \succ 0, X_{ij} \leq 0, i \neq j} \quad & \operatorname{tr}(C_\alpha X) - \log\det(X) \\ \textit{s.t.} \quad & X_{ij} = 0, \quad (i,j) \in G \cup V, \end{aligned} \tag{4}$$

where $V = \{(i,j), i \neq j | A_{ij} = 0\}$ is equivalent to the original problem (O).

Theorem 1 shows that we can exploit the sparsity structure in G using our soft-thresholding function, in addition to the basic sparsity constraint in V obtained from connectivity matrix A. As we shall see, this allows us to speed up computation.

3. PROXIMAL POINT ALGORITHM METHOD

Utilizing the soft-thresholding technique proposed in the Section 2, we obtain Problem (4). We now introduce the slack variable x to transform (4) into following problem:

$$\begin{aligned} \min_{X \succ 0, x \geq 0} \quad & \operatorname{tr}(C_\alpha X) - \log\det(X) \\ \text{s.t.} \quad & X_{ij} = 0, \qquad (i,j) \in G \cup V, \\ & X_{ij} + x_{ij} = 0, \quad i \neq j \text{ and } (i,j) \notin G \cup V. \end{aligned} \tag{5}$$

Note that Problem (5) can be compactly represented as

$$\begin{aligned} \min_{X \succ 0,\ x \geq 0} \quad & \operatorname{tr}(CX) - \log\det(X) \\ \text{s.t.} \quad & \mathcal{A}(X) + Bx = 0, \end{aligned} \tag{P}$$

where $C := C_\alpha$, $B \in \mathbb{R}^{m \times l}$, $\mathcal{A} : \mathbb{S}^n \to \mathbb{R}^m$ is a linear mapping and $\mathcal{A}^T : \mathbb{R}^m \to \mathbb{S}^n$. Note that $\mathcal{A}(X) = [\langle A_1, X\rangle, \ldots, \langle A_m, X\rangle]^T$ and $\mathcal{A}^T(y) = \sum_{i=1}^m y_i A_i$. We can rewrite the constraints in (5) in the form of (P) and omit the details here. Now, we define the feasible set of (P) as

$$\mathcal{F}_P = \{X \in \mathbb{S}^n_{++}, x \in \mathbb{R}^l_+ : \mathcal{A}(X) + Bx = 0\}.$$

Before we proceed, let us state the following lemma, which will be used to derive our proximal point algorithm (PPA).

Lemma 1 ([22, Lem. 2.1])**.** *Let $X \in \mathbb{S}^n$ with $X = UDU^T$ where $D = diag(d)$ with $d = (d_1, \ldots, d_n)$ are the eigenvalues of X. Assume $d_1 \geq \cdots \geq d_r > 0 \geq d_{r+1} \geq \cdots \geq d_n$. For given $\gamma \geq 0$, set $\psi_\gamma^+(x) = (\sqrt{x^2 + 4\gamma} + x)/2$ and $\psi_\gamma^-(x) = (\sqrt{x^2 + 4\gamma} - x)/2$. We define $X_1 = \psi_\gamma^+(X) = U diag(\psi_\gamma^+(d))U^T$ and $X_2 = \psi_\gamma^-(X) = U diag(\psi_\gamma^-(d))U^T$. Then, ψ_γ^+ is differentiable and the derivative $(\psi_\gamma^+)'(X)[H]$ at X for any $H \in \mathbb{S}^n$ is given by*

$$(\psi_\gamma^+)'(X)[H] = U(\Omega \circ (U^T H U))U^T,$$

where $\Omega \in \mathbb{S}^n$ is $\Omega_{ij} = \frac{\psi_\gamma^+(d_i) + \psi_\gamma^+(d_j)}{\sqrt{d_i^2 + 4\gamma} + \sqrt{d_j^2 + 4\gamma}}$ with $i, j = 1, \ldots, n$.

We define the Lagrangian function of Problem (P) by

$$\mathcal{L}(X, x; y) = \begin{cases} \begin{array}{l} \langle C, X\rangle - \log\det X \\ \quad -\langle y, \mathcal{A}(X) + Bx\rangle, \end{array} & (X, x) \in \mathbb{S}^n_+ \times \mathbb{R}^l_+, \\ \infty, & \text{otherwise.} \end{cases}$$

Thus, the essential objective function of (P) is

$$f(X, x) = \max_{y \in \mathbb{R}^m} \mathcal{L}(X, x; y) = \begin{cases} \langle C, X\rangle - \log\det X, & (X, x) \in \mathcal{F}_P, \\ \infty, & \text{otherwise.} \end{cases}$$

Using Moreau-Yosida regularization, we have

$$\begin{aligned} F_\lambda(X, x) &= \min_{Y \in \mathbb{S}^n_{++}, z \in \mathbb{R}^l_+} \left\{ f(Y, z) + \frac{1}{2\lambda}(\|Y - X\|^2 + \|z - x\|^2) \right\} \\ &= \min_{Y \in \mathbb{S}^n_{++}, z \in \mathbb{R}^l_+} \sup_{y \in \mathbb{R}^m} \left\{ \mathcal{L}(Y, z; y) + \frac{1}{2\lambda}\|Y - X\|^2 + \frac{1}{2\lambda}\|z - x\|^2 \right\} \\ &= \sup_{y \in \mathbb{R}^m} \Theta_\lambda(X, x, y), \end{aligned}$$

where $\lambda > 0$, the interchange of min and sup comes from [19], and $\Theta_\lambda(X, x, y) = \min_{z \in \mathbb{R}^l_+} \left\{ -\langle B^T y, z\rangle + \frac{1}{2\lambda}\|z - x\|^2 \right\} + \min_{Y \in \mathbb{S}^n_{++}} \left\{ \langle C - \mathcal{A}^T y, Y\rangle - \log\det Y + \frac{1}{2\lambda}\|Y - X\|^2 \right\}$. To calculate $\Theta_\lambda(X, x, y)$, we have the following lemma:

Algorithm 1 Inexact Proximal Point Algorithm (PPA)

```
1: Input: $X^0 \in \mathbb{S}^n_{++}, x^0 \in \mathbb{R}^l_+$, $\gamma_k = 0$ for all $k$, stopping criterion $\epsilon$.
2: for $k = 0, 1, \dots$ do
3:     Get an approximate solution
           $y^{k+1} \approx \underset{y\in\mathbb{R}^m}{\operatorname{argmax}} \left\{\Theta_{\lambda_k}(X^k, x^k, y)\right\}.$    (6)
4:     Update $X$ and $x$ by
           $X^{k+1} = \psi^+_{\lambda_k}(W_{\lambda_k}(X^k, y^{k+1})),$
           $x^{k+1} = \psi^+_{\gamma_k}(W_{\lambda_k}(x^k, y^{k+1})).$    (7)
5:     Check the stopping criterion
           $\|X^{k+1}-X^k\|_F/\|X^k\|_F \le \epsilon, \|x^{k+1}-x^k\|/\|x^k\| \le \epsilon.$
       If it is not satisfied, set $\lambda_{k+1} = 2\lambda_k$.
6: end for
```

Lemma 2. *For any $Y \in \mathbb{S}^n$ and $\lambda > 0$, we have*

$$\min_{X\succ 0}\left\{-\log\det X + \frac{1}{2\lambda}\|X-Y\|^2\right\} = \frac{1}{2\lambda}\|\psi^-_\lambda(Y)\|^2 - \log\det(\psi^+_\lambda(Y)).$$

where $\psi^+_\lambda(\cdot)$ and $\psi^-_\lambda(\cdot)$ are defined in Lemma 1.

Utilizing Lemma 2 and performing some routine calculations, we obtain

$$\Theta_\lambda(X,x,y) = \frac{1}{2\lambda}\|X\|^2 - \frac{1}{2\lambda}\|\psi^+_\lambda(W_\lambda(X,y))\|^2 + \frac{1}{2\lambda}\|x\|^2 - \frac{1}{2\lambda}\|\psi^+_\gamma(W_\lambda(x,y))\|^2 - \log\det\psi^+_\lambda(W_\lambda(X,y)) + n,$$

where $W_\lambda(X,y) = X - \lambda(C - \mathcal{A}^T y)$, $W_\lambda(x,y) = x + \lambda B^T y$, and $\gamma = 0$. Moreover, we give the first- and second-order derivatives of $\Theta_\lambda(X,x,y)$ w.r.t. y in the following lemma.

Lemma 3 ([22, Lem. 3.2])**.** *For any $y \in \mathbb{R}^m$ and $X \succ 0, x \ge 0$, we have*

$$\nabla_y\Theta_\lambda(X,x,y) = -\mathcal{A}\psi^+_\lambda(W_\lambda(X,y)) - B\psi^+_\gamma(W_\lambda(x,y)),$$
$$\nabla^2_{yy}\Theta_\lambda(X,x,y) = -\lambda(\mathcal{A}(\psi^+_\lambda)'(W_\lambda(X,y))\mathcal{A}^T + B(\psi^+_\gamma)'(W_\lambda(x,y))B^T),$$

where $(\psi^+_\lambda)'(\cdot)$ is defined in Lemma 1.

Denote $y_\lambda(X,x) = \operatorname{argmax}_{y\in\mathbb{R}^m}\Theta_\lambda(X,x,y)$, we see that $\psi^+_\lambda(W_\lambda(X, y_\lambda(X,x)))$ and $\psi^+_\gamma(W_\lambda(x, y_\lambda(X,x))$ are optimal solutions to $F_\lambda(X,x)$ w.r.t. Y and z from our earlier discussions. Hence, $F_\lambda(X,x) = \Theta_\lambda(X,x,y_\lambda(X,x))$. Moreover, by Danskin's Theorem [1], we have

$$\nabla_X F_\lambda(X,x) = \frac{1}{\lambda}(X - \psi^+_\lambda(W_\lambda(X, y_\lambda(X,x)))),$$
$$\nabla_x F_\lambda(X,x) = \frac{1}{\lambda}(x - \psi^+_\gamma(W_\lambda(x, y_\lambda(X,x)))). \quad (8)$$

Then, we use the PPA method to solve Problem (P); i.e., $(X^{k+1}, x^{k+1}) = \operatorname{argmin}_{Y\in\mathbb{S}^n_{++}, z\in\mathbb{R}^l_+}\{f(Y,z) + \frac{1}{2\lambda_k}\|Y - X^k\|^2 + \frac{1}{2\lambda_k}\|z - x^k\|^2\}$. Hence, we get the following update rule from [18, Thm 2.26]:

$$X^{k+1} = X^k - \lambda_k\nabla_X F_\lambda(X^k, x^k) = \psi^+_{\lambda_k}(W_{\lambda_k}(X^k, y_{\lambda_k}(X^k, x^k))),$$
$$x^{k+1} = x^k - \lambda_k\nabla_x F_\lambda(X^k, x^k) = \psi^+_{\gamma_k}(W_{\lambda_k}(x^k, y_{\lambda_k}(X^k, x^k))).$$

Note that in practice it is computationally expensive to get the maximizer $y_\lambda(X,x)$ of $\Theta_\lambda(X,x,y)$. Hence, we propose an efficient inexact proximal point algorithm in Algorithm 1.

To solve the subproblem (6), we introduce a Newton-CG method in Algorithm 2. For simplicity, we denote $\theta_k(y) = \Theta_{\lambda_k}(X^k, x^k, y)$. The stopping criterion to solve the subproblem (6) follows that given in Rockafellar [17]:

$$\sup\theta_k(y) - \theta_k(y^{k+1}) \le \nu_k^2/2\lambda_k, \quad \text{(A)}$$
$$\sup\theta_k(y) - \theta_k(y^{k+1}) \le \delta_k^2/2\lambda_k\|X^{k+1} - X^k\|^2, \quad \text{(B)}$$

where $\sum_{k=0}^\infty \nu_k < \infty$ and $\sum_{k=0}^\infty \delta_k < \infty$.

From Lemma 3 and the positive definiteness property [14] of $(\psi^+_\lambda)'(W_\lambda(X,y))$, we get that $-\nabla^2_{yy}\theta_k(y^i)$ is positive definite, hence $-\nabla^2_{yy}\theta_k(y^i) + \kappa_i I$ is also positive definite as long as $\nabla_y\theta_k(y^i) \ne 0$ from (9). Then, we apply preconditioned CG (PCG) to solve the linear system and d^i is an ascent direction. The global convergence and local quadratic convergence of Algorithm 2 can be established using the techniques in [25].

Algorithm 2 Newton-CG method

```
1: Input: Given $\alpha \in (0, \frac{1}{2}), \beta \in (0,1)$ and $\tau_1, \tau_2 \in (0,1)$.
   Choose $y^0 \in \mathbb{R}^m$.
2: for $i = 0, 1, \dots$ do
3:     Use PCG method to get an ascent direction $d^i$:
           $(\nabla^2_{yy}\theta_k(y^i) - \kappa_i I)d = -\nabla_y\theta_k(y^i),$    (9)
       where $\kappa_i = \tau_1\min\{\tau_2, \|\nabla_y\theta_k(y^i)\|\}$.
4:     By linesearch, $\mu_i = \beta^{m_i}$, where $m_i$ is the first nonnegative integer $m$ such that
           $\theta_k(y^i + \beta^m d^i) \ge \theta_k(y^i) + \alpha\beta^m\langle\nabla_y\theta_k(y^i), d^i\rangle.$
5:     Set $y^{i+1} = y^i + \mu_i d^i$.
6: end for
```

4. CONVERGENCE ANALYSIS

In this section, we establish the convergence of our proximal point algorithm and provide its linear convergence rate.

Theorem 2. *If we choose stopping criterion (A) in Algorithm 1 and the dual of (P) is feasible, then the sequence $\{X^k, x^k\} \subset \mathbb{S}^n_{++} \times \mathbb{R}^l_+$ is bounded and converges to a unique optimal solution (X^*, x^*) of Problem (P).*

Theorem 3. *If we choose stopping criterion (B) and both (P) and its dual are feasible, then the sequence $\{X^k, x^k\} \subset \mathbb{S}^n_{++} \times \mathbb{R}^l_+$ is bounded and converges to a unique optimal solution (X^*, x^*) of Problem (P) linearly.*

Table 1: Comparison of running time (seconds) for PPA and BCD on real datasets with (a) the fully connected graph and (b) the sparsity structural constraints and we set $\alpha = 0.02$.

(a)

dataset	size	PPA	BCD	O-PPA	O-BCD
freeFlyingRobot-1	798	**13**	31	35	37
freeFlyingRobot-2	1338	**43**	146	124	181
bcsstk08	1074	**23**	77	96	95
jagmesh4	1440	**48**	182	82	223
lshp1561	1561	**65**	233	116	286

(b)

dataset	size	PPA	BCD	O-PPA	O-BCD
freeFlyingRobot-1	798	**13**	30	25	30
freeFlyingRobot-2	1338	**38**	144	80	144
bcsstk08	1074	**19**	57	71	58
jagmesh4	1440	**48**	182	63	182
lshp1561	1561	**63**	230	89	231

5. NUMERICAL EXPERIMENTS

In this section, we evaluate the performance of our algorithm and the BCD method in [6] on both synthetic and real datasets. Note that we do not compare with the well-known graphical Lasso method [7], as it has been shown in [6] that BCD is ten times faster than graphical Lasso. We denote by PPA and BCD (resp. O-PPA and O-BCD) the algorithms for solving Problem (P) (resp. Problem (O)).

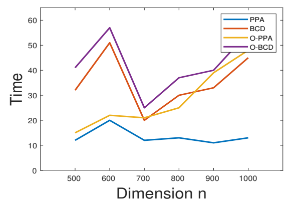
The stopping criterion for our PPA method is stated in Algorithm 1, while for the BCD method we use $\|X^{k+1} - X^k\|_F/\|X^k\|_F \le \epsilon$. We set $\epsilon = 10^{-6}$ for both PPA and BCD. The codes were written in MATLAB and run on a PC with i5-4590 CPU at 3.3 GHz with 8 GB memory.

For synthetic data, we generate a positive definite matrix $\Sigma^{-1} \in \mathbb{S}^n_{++}$ with density $= 0.3$ of nonzero entries in the same way as in [22]. For real data, we use actual graphs from *SuiteSparse Matrix Collection* [3] to design Σ^{-1} as in [24]. The details of the procedure for generating Σ^{-1} are omitted here. Then, we sample kn instances from the multivariate Gaussian distribution $N(0,\Sigma)$ to generate a sample covariance matrix C. In all experiments ,we just set $k = 100$.

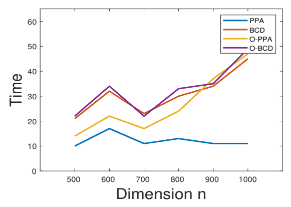
We consider the following scenarios in our tests: (a) The fully connected graph, which means that we set all off-diagonal elements of A to be 1. This is a reasonable setting if we do not know any sparsity information of X. (b) Σ^{-1} has the sparsity structure $A_{ij} = 0$ if $\Sigma^{-1}_{ij} = 0$, which means that we want our estimator to have the same sparsity pattern as Σ^{-1}.

We plot the running time vs different synthetic data size n, which varies from 500 to 1000 in Fig. 1 on both scenarios, and set penalty parameter $\alpha = 0.02$. We can observe that our proposed PPA method, which solves the transformed problem (P), can be much faster than BCD, O-PPA, and O-BCD. It is interesting to see that PPA can be much faster than O-PPA, although they are the same methods that solve the equivalent problems (P) and (O), respectively. Hence, the soft-thresholding technique, which exploits more sparsity information hidden in (O) to construct (P), can truly accelerates our PPA method.

For real data, we report the running time of PPA and BCD on both problems (P) and (O) with 2 scenarios in Table 1 and set $\alpha = 0.02$. Our PPA method for (P) is again several times

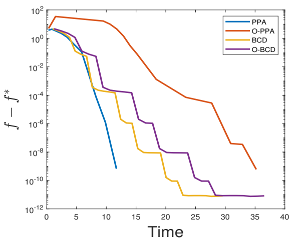


(a)

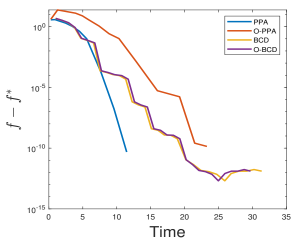


(b)

Fig. 1: Running time (seconds) on synthetic datasets with $n = 500, \ldots, 1000$ for (a) the fully connected graph and (b) the sparsity structural constraints and we set $\alpha = 0.02$.



(a)



(b)

Fig. 2: Convergence result with running time (seconds) on the freeFlyingRobot-1 dataset for (a) the fully connected graph and (b) the sparsity structural constraints and we set $\alpha = 0.02$.

faster than other methods. Moreover, we plot the convergence curves on the freeFlyingRobot-1 dataset in Fig. 2. It is clear that the soft-thresholding technique greatly accelerates our PPA method when comparing the curves of PPA and O-PPA. For the BCD method, it is interesting to observe that sometimes it also has the acceleration effect.

6. CONCLUSION

In this paper, we utilized a soft-thresholding function to transform the generalized graph Laplacian (GGL) problem into an equivalent problem and proposed a fast proximal point algorithm (PPA) to solve the transformed problem with linear convergence guarantee. By exploiting the sparsity structure via the soft-thresholding function, our PPA method for the transformed problem can be much faster than the current state-of-the-art method. The future work would be to extend our algorithmic framework to handle other graph-based problems.

7. REFERENCES

[1] Dimitri P Bertsekas. *Nonlinear Programming*. Athena scientific Belmont, 1999.

[2] Ed Bullmore and Olaf Sporns. Complex brain networks: Graph theoretical analysis of structural and functional systems. *Nature Reviews Neuroscience*, 10(3):186, 2009.

[3] Timothy A Davis and Yifan Hu. The University of Florida sparse matrix collection. *ACM Transactions on Mathematical Software (TOMS)*, 38(1):1, 2011.

[4] Xiaowen Dong, Dorina Thanou, Michael Rabbat, and Pascal Frossard. Learning graphs from data: A signal representation perspective. *IEEE Signal Processing Magazine*, 36(3):44–63, 2019.

[5] Hilmi E Egilmez and Antonio Ortega. Spectral anomaly detection using graph-based filtering for wireless sensor networks. In *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, pages 1085–1089. IEEE, 2014.

[6] Hilmi E Egilmez, Eduardo Pavez, and Antonio Ortega. Graph learning from data under Laplacian and structural constraints. *IEEE Journal of Selected Topics in Signal Processing*, 11(6):825–841, 2017.

[7] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*, 9(3):432–441, 2008.

[8] Frederick S Hillier. *Introduction to Operations Research*. McGraw-Hill Education, 2012.

[9] Cho-Jui Hsieh, Mátyás A Sustik, Inderjit S Dhillon, and Pradeep Ravikumar. QUIC: Quadratic approximation for sparse inverse covariance estimation. *Journal of Machine Learning Research*, 15(1):2911–2947, 2014.

[10] Anil K Jain. *Fundamentals of Digital Image Processing*. Englewood Cliffs, NJ: Prentice Hall,, 1989.

[11] Stephane Lafon and Ann B Lee. Diffusion maps and coarse-graining: A unified framework for dimensionality reduction, graph partitioning, and data set parameterization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(9):1393–1403, 2006.

[12] Gonzalo Mateos, Santiago Segarra, Antonio G. Marques, and Alejandro Ribeiro. Connecting the dots: Identifying network structure via graph signal processing. *IEEE Signal Processing Magazine*, 36(3):16–43, 2019.

[13] Frank McSherry. Spectral partitioning of random graphs. In *Foundations of Computer Science, 2001. Proceedings. 42nd IEEE Symposium on*, pages 529–537. IEEE, 2001.

[14] Fanwen Meng, Defeng Sun, and Gongyun Zhao. Semismoothness of solutions to generalized equations and the Moreau-Yosida regularization. *Mathematical programming*, 104(2-3):561–581, 2005.

[15] Peyman Milanfar. A tour of modern image filtering: New insights and methods, both practical and theoretical. *IEEE Signal Processing Magazine*, 30(1):106–128, 2013.

[16] Eduardo Pavez and Antonio Ortega. Generalized Laplacian precision matrix estimation for graph signal processing. In *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*, pages 6350–6354. IEEE, 2016.

[17] R Tyrrell Rockafellar. Augmented Lagrangians and applications of the proximal point algorithm in convex programming. *Mathematics of Operations Research*, 1(2):97–116, 1976.

[18] R Tyrrell Rockafellar and Roger J-B Wets. *Variational Analysis*, volume 317. Springer Science & Business Media, 2009.

[19] Ralph Tyrell Rockafellar. *Convex Analysis*. Princeton university press, 2015.

[20] Katya Scheinberg, Shiqian Ma, and Donald Goldfarb. Sparse inverse covariance selection via alternating linearization methods. In *Advances in Neural Information Processing Systems*, pages 2101–2109, 2010.

[21] Martin Slawski and Matthias Hein. Estimation of positive definite m-matrices and structure learning for attractive Gaussian Markov random fields. *Linear Algebra and its Applications*, 473:145–179, 2015.

[22] Chengjing Wang, Defeng Sun, and Kim-Chuan Toh. Solving log-determinant optimization problems by a Newton-CG primal proximal point algorithm. *SIAM Journal on Optimization*, 20(6):2994–3013, 2010.

[23] Cha Zhang and Dinei Florêncio. Analyzing the optimality of predictive transform coding using graph-based models. *IEEE Signal Processing Letters*, 20(1):106–109, 2013.

[24] Richard Y Zhang, Salar Fattahi, and Somayeh Sojoudi. Linear-time algorithm for learning large-scale sparse graphical models. *arXiv preprint arXiv:1802.04911*, 2018.

[25] Xin-Yuan Zhao, Defeng Sun, and Kim-Chuan Toh. A Newton-CG augmented Lagrangian method for semidefinite programming. *SIAM Journal on Optimization*, 20(4):1737–1765, 2010.

[26] Xiaojin Zhu, Zoubin Ghahramani, and John D Lafferty. Semi-supervised learning using Gaussian fields and harmonic functions. In *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, pages 912–919, 2003.