LARGE-SCALE l_0 SPARSE INVERSE COVARIANCE ESTIMATION

Goran Marjanovic [♯] *Magnus Ulfarsson* [†] *Victor Solo* [‡]

[♯‡] School of Electrical Eng., University of New South Wales, Sydney, AUSTRALIA
[†] Dept. Electrical Eng., University of Iceland, Reykjavik, ICELAND

ABSTRACT

There has been significant interest in sparse inverse covariance estimation in areas such as statistics, machine learning, and signal processing. In this problem, the sparse inverse of a covariance matrix of a multivariate normal distribution is estimated. A Penalised Log-Likelihood (PLL) optimisation problem is solved to obtain the matrix estimator, where the penalty is responsible for inducing sparsity. The most natural sparsity promoting penalty is the non-convex l_0 function. Due to speed and memory limitations, the existing algorithms for dealing with the non-convex l_0 PLL problem are unable to be used in high dimensional settings. Here we address this issue by presenting a new block iterative approach for this problem, which can handle large-scale data sizes. Simulations demonstrate that our approach outperforms existing methods for this problem.

***Index Terms*—** sparsity, non-convex, l_0 regularisation, iterative shrinkage thresholding, momentum, accelerated gradient, inverse covariance estimation, large-scale

1. INTRODUCTION

In the current age of big data acquisition, there is an ever growing interest in the sparse inverse covariance (precision) matrix estimation. For instance, in large dimensional applications such as biological inferencing in gene networks, discovery of functional brain connectivity patterns or analysis of social interactions, the sparse inverse covariance matrix plays a crucial role in identifying relationships between variables, such as genes, brain regions, or individuals respectively. Specifically, if $\mathbf{x}_{p\times 1}$ is a normally distributed random vector, i.e. $\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, where $\boldsymbol{\mu}_{p\times 1}$ is the mean and $\boldsymbol{\Sigma}_{p\times p}$ is the covariance matrix, the inverse covariance $\boldsymbol{\Omega} = \boldsymbol{\Sigma}^{-1}$ captures the conditional dependency of the components $\mathbf{x}[1], \ldots, \mathbf{x}[p]$ of $\mathbf{x}$ [1–4] – having $\boldsymbol{\Omega}[i,j] = 0$, $i \neq j$, corresponds to $\mathbf{x}[i]$ and $\mathbf{x}[j]$ being conditionally independent. In fact, the zero pattern in $\boldsymbol{\Omega}$ specifies the structure of the underlying undirected graph, i.e. $\boldsymbol{\Omega}[i,j] = 0$ corresponds to no link between variables (nodes) i and j. This provides a fundamental approach to the reduction of large dimensional data.

Following the parsimony principle, the estimation objective is to choose the simplest model, i.e. the sparsest $\boldsymbol{\Omega}$ (undirected graph) that adequately explains the data. The sparsity requirement improves the interpretability of the model and reduces over-fitting.

The maximum log-likelihood estimator of $\boldsymbol{\Sigma}$ is given by the sample covariance matrix $\mathbf{S} \succeq \mathbf{0}$, constructed using n normally distributed data samples $\{\mathbf{x}_i\}_{i=1}^n$. In real life applications, it is frequently the case that $n < p$, which makes $\mathbf{S}$ non-invertible. Hence, to estimate $\boldsymbol{\Omega}$, here we consider the popular PLL formulation – we minimise, over the set of positive definite (symmetric) matrices, a sparsity penalised log-likelihood objective function, i.e.

$$\min_{\mathbf{X}\succ\mathbf{0}} F(\mathbf{X}) = \min_{\mathbf{X}\succ\mathbf{0}} -\log\det(\mathbf{X}) + \text{tr}(\mathbf{XS}) + \lambda\|\mathbf{X}\|_0 \quad (1)$$

$\text{tr}(\cdot)$ is the trace of a matrix, $\|\cdot\|_0$ is the sparsity promoting penalty, and $\lambda > 0$ is a penalty (regularisation) constant. The first two terms in (1) together represent the negative of the log-likelihood of $\mathbf{X}$, which promotes goodness-of-fit of the estimator. The penalty is the non-convex l_0 function, defined by

$$\|\mathbf{X}\|_0 = \sum_{i=1}^{p}\sum_{j=1}^{p} \mathbb{I}\left(\mathbf{X}[i,j] \neq 0\right) \quad (2)$$

where $\mathbb{I}(\cdot)$ is the indicator function, equaling 1 if $(\cdot)$ is true, and zero otherwise. $\|\mathbf{X}\|_0$ is the natural sparsity inducing penalty that counts the number of non-zeros in $\mathbf{X}$, and thus, forces many of the estimator entries to become zero. It is considered in many sparse signal estimation problems, e.g. inverse covariance estimation, linear regression, matrix completion, and others [5–17].

The l_0 vs. l_1 sparsity penalty – The convex l_1 norm is a popular sparsity surrogate for the non-convex l_0 penalty (2), making the resulting l_1 PLL objective function, for sparse inverse covariance estimation, also convex. Having convexity is very useful when developing algorithms for minimising large-scale instances of the l_1 PLL, e.g. see [18–22]. These latest algorithms rely on CD, quasi-Newton and Majorisation-Minimisation (MM) flavoured techniques. However, the procedures considered heavily depend on the fact that the objective function is convex, and hence, are unique to the l_1 PLL. Due to non-convexity and non-continuity of the l_0 PLL, these techniques cannot be directly exploited for minimising (1), and, if applied blindly, could result in an unstable algorithm.

Lastly, replacing the l_0 penalty with the l_1 penalty in sparsity penalised problems is undesirable because it is increasingly becoming known that using the l_1 norm results in less sparse and negatively biased solutions (estimators) [7–9, 15, 16, 23–25].

Prior Work – Existing Methods for Minimising the l_0 PLL criterion (1) – The l_0 function (2) is not continuous and non-convex. This makes the l_0 PLL criterion (1) also non-convex, and thus, difficult to minimise. Despite this, several algorithms have recently been

[♯] Email: g.marjanovic@unsw.edu.au. Author information also available on Research Gate: https://www.researchgate.net/profile/Goran_Marjanovic4

[†] This work was partially supported by the Research Fund of the University of Iceland and the Icelandic Research Fund (130635-051). Email: mou@hi.is. Author information also available on Research Gate: https://www.researchgate.net/profile/Magnus_Ulfarsson

[‡] This work was partially supported by an ARC (Australian Research Council) grant. Email: v.solo@unsw.edu.au.

developed for its minimisation, see [7–9, 16, 17]. They are either coordinate-wise [7, 16] and/or block-wise [8, 9, 17] Cyclic Descent (CD) methods. The coordinate-wise CD algorithm updates each matrix entry at a time by fixing all the other entries. Similarly, the block-wise CD algorithm updates each row-column at a time by keeping all the other rows-columns fixed. To update the entries in the target row-column [8, 9, 17] use a coordinate-wise CD algorithm.

The existing methods perform well on relatively smaller problem sizes, i.e. when $p < 10^3$. However, for large-scale instances of (1), i.e. when p is several thousand, they are not desirable. Examples of large problem sizes occur in brain imaging and gene expression analysis applications, and so, in the current age of big data, there is an ever increasing need to scale up algorithms.

Due to the presence of the log-likelihood in (1), one must deal with the elements and/or columns of the inverse of a sparse matrix iterate – a dense matrix. Hence, a large p introduces certain limitations: (i) Computing the inverse – this is expensive even if rank one/two updates (Sherman-Morrison-Woodbury (SMW) formulae) are considered since they require up to $\mathcal{O}(p^2)$ products. (ii) Storing the inverse – a sparse matrix iterate will always fit in memory since we only need to store its non-zeros and their locations. However, because its inverse is usually dense, it has $\mathcal{O}(p^2)$ values, and so, it might not fit into memory. Thus, formulae such as the SMW are not desirable, and because the existing methods heavily rely on them, they are not suitable for solving large-scale instances of (1).

Current Contribution – For minimising large-scale instances of the l_0 PLL criterion (1), here we present a block-wise CD algorithm that incorporates a fast MM type procedure. We note that there are no methods capable of handling large-scale instances of (1). In the algorithm we present: (i) Only matrix-vector and vector-vector products (predominantly sparse) are used. (ii) Only the sparse matrix iterate and two vectors need to be stored in memory per iteration. (iii) No computation of (dense) matrix inverses is required. This makes our algorithm fundamentally different to the existing algorithms for minimising (1).

The rest of the paper is organised as follows. Section 2 describes the algorithm procedure, which is stated in Section 3. Simulations are provided in Section 4, while concluding remarks in Section 5.

Additional Notation – Given a vector $\mathbf{v}$, $\mathbf{v}[i]$ is its i-th entry. $\mathbf{M}[i,j]$ is the ij-th entry of a matrix $\mathbf{M}$. $\sigma_s(\mathbf{M})$ and $\sigma_l(\mathbf{M})$ denotes the smallest and the largest eigenvalue of a square $\mathbf{M}$ respectively. If $f : \mathbb{R}^p \to \mathbb{R}^q$, then given $\mathbf{x} \in \mathbb{R}^p$, $f(\mathbf{x})[i]$ denotes the i-th component of $f(\mathbf{x}) \in \mathbb{R}^q$. Given a vector $\mathbf{v}$, $\mathrm{diag}(\mathbf{v})$ is a diagonal matrix with $\mathbf{v}$ on its main diagonal, i.e. $\mathrm{diag}(\mathbf{v})[i,i] = \mathbf{v}[i]$.

2. ALGORITHM DEVELOPMENT

The algorithm proceeds in two stages. The first stage is a Block-wise CD (BCD) procedure, where the plan is to optimise over one row and column of the variable matrix $\mathbf{X}$ at a time. The BCD reduces the problem of optimising over a matrix in (1) to a problem of optimising over a vector. The second stage does the latter optimisation using a fast and novel MM type method. We now describe in detail the BCD and the MM procedures in the following two subsections.

2.1. The Block-wise Cyclic Descent (BCD) Stage

Letting $\mathbf{X}$ denote the current iterate, the new row-column to be updated is identified and the rest of the matrix entries are fixed. To describe the updating procedure we have to deal with permutations of $\mathbf{X}$ and $\mathbf{S}$, which have the identified target row and column placed at the end i.e.

$$\mathbf{X}_\pi = \begin{bmatrix} \mathbf{V} & \mathbf{u} \\ \mathbf{u}^T & w \end{bmatrix} \quad \mathbf{S}_\pi = \begin{bmatrix} \mathbf{\Gamma} & \boldsymbol{\gamma} \\ \boldsymbol{\gamma}^T & \gamma_0 \end{bmatrix} \tag{3}$$

where $\mathbf{V} \succ \mathbf{0}$ and $\mathbf{\Gamma} \succeq \mathbf{0}$ are $(p-1)\times(p-1)$, $\mathbf{u}, \boldsymbol{\gamma}$ are $(p-1)\times 1$, and $w > 0$, $\gamma_0 > 0$ are scalars. Since $F|_{\mathbf{X},\mathbf{S}} = F|_{\mathbf{X}_\pi,\mathbf{S}_\pi}$, updating the rows-columns in $\mathbf{X}$ is equivalent to updating $\mathbf{u}$ and w. So, substituting (3) in $F(\cdot)$, and using the well known properties of determinant and trace functions, it is easy to see that

$$\begin{aligned} F(\mathbf{X}) = F(\mathbf{X}_\pi) &= -\log\det(\mathbf{X}_\pi) + \mathrm{tr}(\mathbf{X}_\pi\mathbf{S}_\pi) + \lambda\|\mathbf{X}_\pi\|_0 \\ &= -\log\left\{\det(\mathbf{V})(w - \mathbf{u}^T\mathbf{V}^{-1}\mathbf{u})\right\} + \mathrm{tr}(\mathbf{V}\mathbf{\Gamma}) + \mathrm{tr}(\mathbf{u}\boldsymbol{\gamma}^T) \\ &\quad + \boldsymbol{\gamma}^T\mathbf{u} + \gamma_0 w + \lambda\|\mathbf{V}\|_0 + 2\lambda\|\mathbf{u}\|_0 \end{aligned}$$

By fixing $\mathbf{V}$, minimising $F(\mathbf{X})$ reduces to minimising

$$-\log(w - \mathbf{u}^T\mathbf{V}^{-1}\mathbf{u}) + 2\boldsymbol{\gamma}^T\mathbf{u} + \gamma_0 w + 2\lambda\|\mathbf{u}\|_0 \tag{4}$$

with respect to $\mathbf{u}$ and $w > 0$, where $\|\mathbf{u}\|_0 = \sum_{i=1}^{p-1}\mathbb{I}(\mathbf{u}[i] \neq 0)$. The positive definite constraint on $\mathbf{X}$ becomes $w - \mathbf{u}^T\mathbf{V}^{-1}\mathbf{u} > 0$, which is implicitly enforced by $\log(\cdot)$. (4) is now minimised using a fast and novel MM type procedure described next.

2.2. The Majorisation-Minimisation (MM) Stage

Firstly, elementary calculus applied to (4) yields the minimiser

$$\widehat{w} = \widehat{w}(\mathbf{u}) = \mathbf{u}^T\mathbf{V}^{-1}\mathbf{u} + \gamma_0^{-1} > 0 \tag{5}$$

Since $\widehat{w} - \mathbf{u}^T\mathbf{V}^{-1}\mathbf{u} > 0$, substituting (5) back into (4) gives $-\log(\gamma_0^{-1}) + 2\boldsymbol{\gamma}^T\mathbf{u} + \gamma_0\mathbf{u}^T\mathbf{V}^{-1}\mathbf{u} + 1 + 2\lambda\|\mathbf{u}\|_0$, which we now need to minimise with respect to $\mathbf{u}$, and is equivalent to minimising

$$J(\mathbf{u}) = \frac{1}{2}\gamma_0\mathbf{u}^T\mathbf{V}^{-1}\mathbf{u} + \boldsymbol{\gamma}^T\mathbf{u} + \lambda\|\mathbf{u}\|_0 \tag{6}$$

with respect to $\mathbf{u}$. Our approach for achieving this extends our approach in [5] for minimising an l_0 penalised least squares criterion. Specifically, we use an accelerated MM flavoured method – the Momentumised Iterative Shrinkage Thresholding (MIST) algorithm, with an additional line-search procedure to efficiently minimise the non-convex function (6). MIST relies on the basic MM technique [26], which uses a majorisation function to achieve minimisation. So, defining $\phi(\mathbf{u}) = \frac{1}{2}\gamma_0\mathbf{u}^T\mathbf{V}^{-1}\mathbf{u} + \boldsymbol{\gamma}^T\mathbf{u}$, the function $Q_\mu(\mathbf{z},\mathbf{u}) = \phi(\mathbf{u}) + \nabla\phi(\mathbf{u})^T(\mathbf{z}-\mathbf{x}) + \frac{\mu}{2}\|\mathbf{z}-\mathbf{x}\|_2^2 + \lambda\|\mathbf{z}\|_0$ is a majorisation function of $J(\cdot)$, i.e. it is easy to check that $J(\mathbf{z}) \leq Q_\mu(\mathbf{z},\mathbf{u})$ for any $\mathbf{u},\mathbf{z}$, and $\mu \geq \gamma_0\,\sigma_l(\mathbf{V}^{-1})$. So, letting $\mathcal{P}_\mu(\mathbf{u}) = \arg\min_{\mathbf{z}} Q_\mu(\mathbf{z},\mathbf{u})$ leads to the basic MM result

$$J(\mathcal{P}_\mu(\mathbf{u})) \leq Q_\mu(\mathcal{P}_\mu(\mathbf{u}),\mathbf{u}) \leq Q_\mu(\mathbf{u},\mathbf{u}) = J(\mathbf{u}) \tag{7}$$

Thus, in the basic MM algorithm, if $\mathbf{u}$ is the current iterate, letting $\mathcal{P}_\mu(\mathbf{u})$ be the next iterate guarantees descent in $J(\cdot)$. MIST accelerates this algorithm [5] by incorporating a momentum term, and relies on the following result that extends the basic MM result (7)

Theorem 1. Let $\mathbf{B}_\mu = \mu\mathbf{I} - \gamma_0\mathbf{V}^{-1}$, where $\mu > \gamma_0\,\sigma_l(\mathbf{V}^{-1})$, and

$$\alpha = 2\eta\left(\frac{\boldsymbol{\delta}^T\mathbf{B}_\mu(\mathcal{P}_\mu(\mathbf{u}) - \mathbf{u})}{\boldsymbol{\delta}^T\mathbf{B}_\mu\boldsymbol{\delta}}\right), \quad \eta \in [0,1] \tag{8}$$

where $\boldsymbol{\delta} \neq \mathbf{0}$. Then,

$$J(\mathcal{P}_\mu(\mathbf{u}+\alpha\boldsymbol{\delta})) \leq Q_\mu\left(\mathcal{P}_\mu(\mathbf{u}+\alpha\boldsymbol{\delta}), \mathbf{u}+\alpha\boldsymbol{\delta}\right) \leq J(\mathbf{u}) \quad (9)$$

The proof is omitted due to lack of space. MIST is essentially a repeated application of Theorem 1 – if $\mathbf{u}$ is the current iterate, then $\mathcal{P}_\mu(\mathbf{u}+\alpha\boldsymbol{\delta})$ is the next iterate, where $\alpha\boldsymbol{\delta}$ is the momentum term. Note that setting $\eta = 0$ gives the basic MM algorithm. We choose $\boldsymbol{\delta}$ to be the difference between the current iterate $\mathbf{u}$ and the previous iterate $\mathbf{u}_-$, i.e. $\boldsymbol{\delta} = \mathbf{u} - \mathbf{u}_-$. (9) makes MIST a monotonic method, i.e. $J(\mathbf{u}) \leq J(\mathbf{u}_-)$. It was demonstrated in [5] that having $\alpha\boldsymbol{\delta} \neq \mathbf{0}$ accelerates the basic MM algorithm.

Remark 1. **(line-search)** Implementing MIST requires knowing the Lipschitz constant $\gamma_0\sigma_l(\mathbf{V}^{-1})$. However, for large-scale instances of criterion (6), this quantity is too expensive to compute. We therefore use a back-tracking step-size rule – noting that $\sigma_l(\mathbf{V}^{-1}) = 1/\sigma_s(\mathbf{V})$, and $\min_i \mathbf{V}[i,i] \geq \sigma_s(\mathbf{V})$, we let $\mu_0 = \frac{\gamma_0}{\min_i \mathbf{V}[i,i]}$, set $\mu = \mu_0\zeta^r$, where $\zeta > 1$, and choose the smallest $r = 0, 1, \ldots,$ such that $\Delta = Q_\mu(\mathcal{P}_\mu(\mathbf{u}+\alpha\boldsymbol{\delta}), \mathbf{u}+\alpha\boldsymbol{\delta}) - J(\mathcal{P}_\mu(\mathbf{u}+\alpha\boldsymbol{\delta})) \geq 0$.

Lastly, an expression for $\mathcal{P}_\mu(\cdot)$ is needed. Letting $g(\mathbf{u}) = \mathbf{u} - (1/\mu)\nabla\phi(\mathbf{u})$, it is easy to show that its components $\mathcal{P}_\mu(\cdot)[i]$ are given by the well known hard-thresholding operator (a shrinkage-thresholding map) [5, Section 2.1], [9, 13, 27]

$$\mathcal{P}_\mu(\mathbf{u})[i] = \mathcal{H}_{\lambda/\mu}(g(\mathbf{u}))[i] = g(\mathbf{u})[i]\mathbb{I}(|g(\mathbf{u})[i]| > \sqrt{2\lambda/\mu}) \quad (10)$$

Remark 2. **(Dense Inverse Matrices)** Updating $\mathbf{u}, w$ using MIST and (5) requires computing $\mathbf{v} = \mathbf{V}^{-1}\mathbf{u}$, where $\mathbf{V}^{-1}$ is dense and $\mathbf{u}$ is sparse. When p is large, obtaining $\mathbf{v}$ by direct methods that rely on the Cholesky factorisation is not efficient since up to $\mathcal{O}(p^3)$ operations are required. Iterative methods, such as Conjugate Gradient (CG), is preferred because the cost of each iteration is proportional to the number of non-zeros in the sparse matrix $\mathbf{V}$ – a small number.

3. ALGORITHM STATEMENT

The algorithm for minimising the l_0 PLL – The Momentimised IST for sparse Inverse Covariance (MISTIC) estimation, is given below

The MISTIC Algorithm for minimising the l_0 PLL criterion (1)

Input: Initial $\mathbf{X} \succ \mathbf{0}$ (in sparse format, see [28]), $\mathbf{S}$, $\lambda > 0$.

01. Identify the new target row-column of $\mathbf{X}$, i.e. $\mathbf{u}, w$.
02. Identify the corresponding row-column of $\mathbf{S}$, i.e. $\boldsymbol{\gamma}, \gamma_0$.

Beginning of MIST (with line-search) for minimising $J(\cdot)$ in (6):

03. Let $\mu_0 = \frac{\gamma_0}{\min_i \mathbf{V}[i,i]}$, and choose $\zeta > 1$ and $N \geq 1$.
04. $\mathbf{u}_- \leftarrow \mathbf{u}$
05. $\mathbf{v} = \gamma_0\mathbf{V}^{-1}\mathbf{u}$ *% compute using CG*
06. $\mathbf{v}_- \leftarrow \mathbf{v}$
07. **for** $k = 1, \ldots, N$ **do:**
08. $\quad \boldsymbol{\delta} = \mathbf{u} - \mathbf{u}_-$
09. $\quad r = 0$ and $\Delta = -\epsilon < 0$
10. $\quad$ **while** $\Delta < 0$ **do:** *% line-search, see Remark 1*
11. $\quad\quad \mu = \mu_0\zeta^r$
12. $\quad\quad \boldsymbol{\epsilon} = \mathbf{B}_\mu\boldsymbol{\delta} = \mu\boldsymbol{\delta} - (\mathbf{v} - \mathbf{v}_-)$
13. $\quad\quad \beta = \boldsymbol{\delta}^T\mathbf{B}_\mu\boldsymbol{\delta} = \boldsymbol{\epsilon}^T\boldsymbol{\delta}$
14. $\quad$ **while** $\beta \leq 0$ **&** $k \neq 1$ **do:**
15. $\quad\quad\quad r \leftarrow r + 1$
16. $\quad\quad\quad \mu = \mu_0\zeta^r$
17. $\quad\quad\quad \boldsymbol{\epsilon} = \mu\boldsymbol{\delta} - (\mathbf{v} - \mathbf{v}_-)$
18. $\quad\quad\quad \beta = \boldsymbol{\epsilon}^T\boldsymbol{\delta}$
19. $\quad$ **end**
20. $\quad \mathbf{g} = \mathbf{u} - \frac{1}{\mu}\nabla\phi(\mathbf{u}) = \mathbf{u} - \frac{1}{\mu}(\mathbf{v} + \boldsymbol{\gamma})$
21. $\quad$ **if** $\beta < 10^{-15}$ **then:** *% improves stability*
22. $\quad\quad \alpha = 0$
23. $\quad$ **else**
24. $\quad\quad \mathbf{p} = \mathcal{H}_{\lambda/\mu}(\mathbf{g}) - \mathbf{u}$
25. $\quad\quad$ Choose $\eta \in [0, 1]$ and compute

$$\alpha = 2\eta\left(\frac{\boldsymbol{\delta}^T\mathbf{B}_\mu(\mathcal{P}_\mu(\mathbf{u}) - \mathbf{u})}{\boldsymbol{\delta}^T\mathbf{B}_\mu\boldsymbol{\delta}}\right) = 2\eta\left(\frac{\boldsymbol{\epsilon}^T\mathbf{p}}{\beta}\right)$$

26. $\quad$ **end**
27. $\quad \widehat{\mathbf{u}} = \mathcal{P}_\mu(\mathbf{u}+\alpha\boldsymbol{\delta}) = \mathcal{H}_{\lambda/\mu}\left(\mathbf{g} + \frac{\alpha}{\mu}\boldsymbol{\epsilon}\right)$
28. $\quad \widehat{\mathbf{v}} = \gamma_0\mathbf{V}^{-1}\widehat{\mathbf{u}}$ *% compute using CG*
29. $\quad \mathbf{d} = \widehat{\mathbf{u}} - (\mathbf{u} + \alpha\boldsymbol{\delta})$
30. $\quad$ Compute $\mathbf{z} = \gamma_0\mathbf{V}^{-1}\mathbf{d}$, i.e.

$$\mathbf{z} = \gamma_0\mathbf{V}^{-1}\widehat{\mathbf{u}} - \gamma_0\mathbf{V}^{-1}\mathbf{u} - \alpha\gamma_0\mathbf{V}^{-1}(\mathbf{u} - \mathbf{u}_-) = \widehat{\mathbf{v}} - (1+\alpha)\mathbf{v} + \alpha\mathbf{v}_-$$

31. $\quad$ Compute Δ, i.e.

$$\Delta = Q_\mu(\mathcal{P}_\mu(\mathbf{u}+\alpha\boldsymbol{\delta}), \mathbf{u}+\alpha\boldsymbol{\delta}) - J(\mathcal{P}_\mu(\mathbf{u}+\alpha\boldsymbol{\delta})) = \frac{1}{2}\mathbf{d}^T\mathbf{B}_\mu\mathbf{d} = \frac{1}{2}\mathbf{d}^T(\mu\mathbf{d} - \mathbf{z})$$

32. $\quad\quad r \leftarrow r + 1$
33. $\quad$ **end** *% end of line-search*
34. $\quad \mathbf{u}_- \leftarrow \mathbf{u}$
35. $\quad \mathbf{u} \leftarrow \widehat{\mathbf{u}}$
36. $\quad \mathbf{v}_- \leftarrow \mathbf{v}$
37. $\quad \mathbf{v} \leftarrow \widehat{\mathbf{v}}$
38. **end**

End of MIST (with line-search) for minimising $J(\cdot)$ in (6).

39. Using the final $\mathbf{u}$ and $\mathbf{v}$ from MIST (above), denoted by $\mathbf{u}_+$ and $\mathbf{v}_+$ respectively, compute $w_+ = \widehat{w}$ using (5), i.e.

$$w_+ = \widehat{w}(\mathbf{u}_+) = \gamma_0^{-1}\mathbf{u}_+^T\mathbf{v}_+ + \gamma_0^{-1} = \gamma_0^{-1}(\mathbf{u}_+^T\mathbf{v}_+ + 1) \quad (11)$$

40. Replace the old row-column in $\mathbf{X}$, i.e. $\mathbf{u}, w$, with the new row-column, i.e. $\mathbf{u}_+, w_+$. Go to 01.

Theorem 2. If $\mathbf{X} \succ \mathbf{0}$ is the current iterate and $\mathbf{X}^+$ is $\mathbf{X}$ with the updated target row-column, then $\mathbf{X}^+ \succ \mathbf{0}$ and $F(\mathbf{X}^+) \leq F(\mathbf{X})$.

Remark 3. (a) MISTIC only consists of matrix-vector and vector-vector products. The matrix iterate we are dealing with is sparse and so are a lot of the vectors, making these products efficient. (b) The non-sparse vectors are $\mathbf{v}, \mathbf{v}_-, \boldsymbol{\epsilon}, \mathbf{g}, \widehat{\mathbf{v}}$ and $\mathbf{z}$. $\mathbf{v}$ and $\widehat{\mathbf{v}}$ are calculated efficiently using CG because $\mathbf{V}$ is sparse. $\boldsymbol{\epsilon}$, $\mathbf{g}$ and $\mathbf{z}$ are obtained simply by vector addition (efficient). Operations with the non-sparse vectors are vector addition, multiplication (inner-products) all done with sparse vectors, and zeroing – all efficient operations. (c) Initialising $\mathbf{X}$ such that it is in the sparse format is done using the Matlab function `sparse` [28]. In this case, the target row-column of $\mathbf{X}$, i.e. $\mathbf{u}$, and hence, all the variables in MISTIC will also be in the sparse format. (d) The vector $\mathcal{H}_{\lambda/\mu}(\mathbf{a})$ (for a given $\mathbf{a}$) used in lines 24 and 27 of the algorithm is constructed by: (i) finding the index set $\mathcal{I} = \{i : |\mathbf{a}[i]| > \sqrt{2\lambda/\mu}\}$ – a small set since $\mathcal{H}_{\lambda/\mu}(\cdot)$ is sparse, and (ii) using $\mathcal{I}$ in the Matlab function `sparse` to add the non-zeros $\mathbf{a}[i]$, $i \in \mathcal{I}$, forming $\mathcal{H}_{\lambda/\mu}(\mathbf{a})$. (e) The algorithm requires storing only two vectors (in each iteration of MIST): $\mathbf{u}_-$, $\mathbf{v}_-$.

4. SIMULATIONS

We compare MISTIC with all the existing methods [7–9, 16, 17] for minimising the l_0 PLL (1). The l_qCOV algorithm from [9, 17] with $q = 0$ is identical to the LOCOV algorithm from [8]. The algorithm from [7, 16] will be referred to as l_0CD. To test the performance of MIST for minimising $J(\cdot)$, we make additional comparisons. Specifically, we compare MISTIC with two algorithms that use the described BCD procedure (Section 2.1), where MIST with Line-Search (LS) (lines 03-38) is replaced with: (i) the basic MM algorithm with LS (Section 2.2), and (ii) a direct (naive) implementation of the fast FISTA method from [29] with LS and the hard-thresholding map (10). We refer to these methods as BCD-MM and BCD-FISTA respectively, and note that, they do not exist.

All algorithms are initialised with a diagonal matrix $\text{diag}(\mathbf{s})$ (in sparse format), and $\mathbf{s}[i] = 1/\mathbf{S}[i,i]$, see [17, Theorem 3]. All algorithms are terminated if $\frac{F(\mathbf{X})-F(\mathbf{X}_+)}{|F(\mathbf{X})|} \leq \texttt{tol}$, where $\mathbf{X}_+$ is a single sweep of all the rows-columns of $\mathbf{X}$, or when 30 iterations (sweeps) have been reached. In MISTIC, $N = 0.5p$, $\zeta = 2$, and $\eta = 1$, and MIST with LS is terminated if $\|\boldsymbol{\delta}\|_2 \leq 10^{-5}$. The CG (Polak-Ribiere) method is terminated if the norm of the residual $\leq 10^{-4}$. All algorithms are implemented in Matlab R2013a, and run on a 3.20 GHz Intel Core i5 machine with 16 GB of RAM and Windows 7 OS.

Two types of ground truth sparse $\boldsymbol{\Omega}$'s are constructed for estimation, corresponding to: (i) Chain Graphs – $\boldsymbol{\Omega}$ is set to be $\boldsymbol{\Omega}[i,i] = 1.25$ and $\boldsymbol{\Omega}[i,i-1] = \boldsymbol{\Omega}[i-1,i] = -0.5$. (ii) Graphs with random sparse structure – We consider the procedure in [30, Example 1]. Specifically, we first generate a sparse matrix $\mathbf{U}$ with non-zeros equal to ± 1, set $\boldsymbol{\Omega}$ to be $\mathbf{U}^T\mathbf{U}$ and then add a diagonal term to ensure $\boldsymbol{\Omega} \succ \mathbf{0}$. We control the number of non-zeros in $\mathbf{U}$ so that the resulting $\boldsymbol{\Omega}$ has approximately $2p$ non-zeros.

Given $\boldsymbol{\Omega}$, we draw a limited number, $n = 0.4p$, of data samples from $\mathcal{N}(\mathbf{0}, \boldsymbol{\Omega}^{-1})$ to construct $\mathbf{S}$. λ is chosen such that the algorithm solutions have (approx.) the correct number of non-zeros. Their quality is measured by the Matthew's Correlation Coefficient [31]: $\text{MCC} = \frac{\text{TP}\times\text{TN}-\text{FP}\times\text{FN}}{\sqrt{(\text{TP}+\text{FP})(\text{TP}+\text{FN})(\text{TN}+\text{FP})(\text{TN}+\text{FN})}}$, where TP, TN, FP, FN are the number of true positives/negatives, false positives/negatives. The closer the MCC is to 1 the preferable the solution [9, 31].

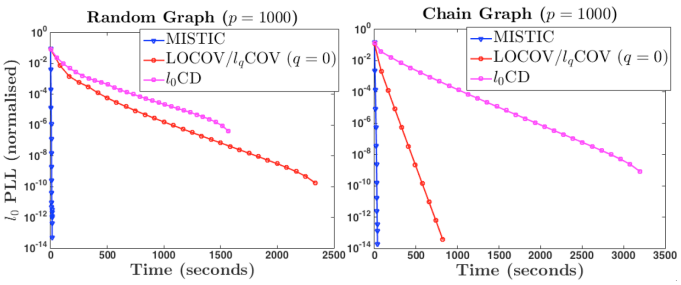

Fig. 1: Plot of runtime vs. $\frac{F(\mathbf{X})-F(\mathbf{X}_\star)}{|F(\mathbf{X}_\star)|}$, where $\mathbf{X}_\star$ is the final iterate ($\texttt{tol} = 10^{-13}$). (Left) MCC ≥ 0.88 for all. (Right) MCC ≥ 0.997 for all. The qualities of all the solutions are very similar, and MISTIC clearly outperforms the existing methods for $p = 1000$.

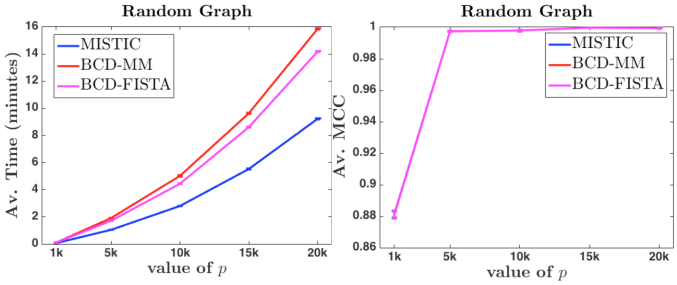

Fig. 2: $\texttt{tol} = 10^{-4}$. (Left) p vs. average runtimes. (Right) p vs. average MCC. The averages are over 5 instances of $\mathbf{S}$. The vertical bars represent standard error. $p = \{1, 5, 10, 15, 20\} \times 10^3$. We see that MISTIC outperforms the two constructed methods. The runtimes for LOCOV/l_qCOV ($q = 0$) and l_0CD are not included because they are too large, e.g. for $p = 5000$, the former takes 10.46 hours, while the latter 2 hours to reach a solution.

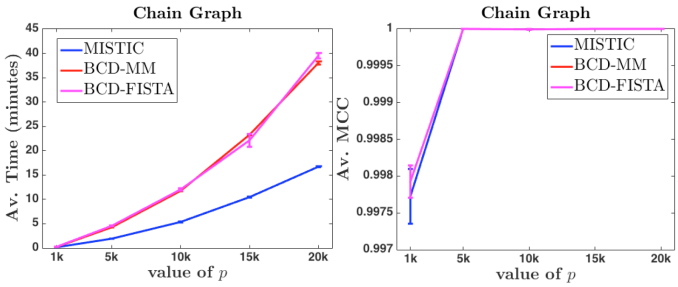

Fig. 3: $\texttt{tol} = 10^{-4}$. (Left) p vs. average runtimes. (Right) p vs. average MCC. The averages are over 5 instances of $\mathbf{S}$. The vertical bars are standard errors. $p = \{1, 5, 10, 15, 20\} \times 10^3$. The runtimes for LOCOV/l_qCOV ($q = 0$) and l_0CD are too large to include, e.g. for $p = 5000$ it takes > 3 hours to reach a solution.

5. CONCLUSION

We have presented a BCD algorithm (MISTIC) that uses a fast MM type procedure (MIST with line-search), for minimising large-scale instances of the l_0 PLL problem (1). The given simulations show that MISTIC significantly outperforms the existing methods that deal with the l_0 PLL criterion. Additionally, using MIST in the block-wise procedure resulted in a superior performance compared to using the basic MM algorithm and the naive implementation of FISTA.

6. REFERENCES

[1] O. Banerjee, L. E. Ghaoui, and A. d'Aspremont, "Model selection through sparse maximum likelihood estimation for multivariate gaussian or binary data," *J. Mach. Learn. Res.*, vol. 9, pp. 485–516, 2008.

[2] A. P. Dempster, "Covariance selection," *Biometrics*, vol. 28, pp. 157–175, 1972.

[3] J. Whittaker, *Graphical Models in Applied Mathematical Analysis*. New York: Wiley, 1990.

[4] S. L. Lauritzen, *Graphical Models*. Oxford: Oxford University Press, 1991.

[5] G. Marjanovic, M. O. Ulfarsson, and A. O. Hero III, "MIST: l_0 Sparse linear regression with momentum," *IEEE ICASSP*, 2015.

[6] M. O. Ulfarsson, V. Solo, and G. Marjanovic, "Sparse and low rank decomposition using l_0 penalty," *IEEE ICASSP*, 2015.

[7] G. Marjanovic and A. O. Hero III, "l_0 Sparse Inverse Covariance Estimation," *IEEE T. Signal Proces.*, vol. 63, no. 12, pp. 3218–3231, 2015.

[8] G. Marjanovic and V. Solo, "l_0 sparse graphical modeling," *IEEE ICASSP*, pp. 2084–2087, 2011.

[9] ——, "On l_q optimization and sparse inverse covariance selection," *IEEE T. Signal Proces.*, vol. 62, no. 7, 2014.

[10] R. Mazumder, T. Hastie, and R. Tibshirani, "Spectral regularization algorithms for learning large incomplete matrices," *J. Mach. Learn. Res.*, vol. 11, pp. 2287–2322, 2010.

[11] M. Ulfarsson and V. Solo, "Vector l_0 sparse variable PCA," *IEEE T. Signal Proces.*, vol. 59, no. 5, pp. 1949–1958, 2011.

[12] A. Seneviratne and V. Solo, "On vector l_0 penalized multivariate regression," *IEEE ICASSP*, pp. 3613–3616, 2012.

[13] T. Blumensath, M. Yaghoobi, and M. E. Davies, "Iterative hard thresholding and l_0 regularisation," *IEEE ICASSP*, vol. 3, pp. 1–4, 2007.

[14] T. Blumensath and M. Davies, "Iterative thresholding for sparse approximations," *J. Fourier Anal. Appl.*, vol. 14, no. 5, pp. 629–654, 2008.

[15] D. Lin, E. Pitler, D. P. Foster, and L. H. Ungar, "In Defense of l_0," *The Journal of Machine Learning Research* 7, pp. 1861–1885, 2006.

[16] G. Marjanovic and A. O. Hero III, "l_0 sparse inverse covariance estimation," 2014, arXiv: http://arxiv.org/abs/1408.0850.

[17] ——, "On l_q estimation of sparse inverse covariance," *IEEE ICASSP*, 2014.

[18] C.-J. Hsieh, M. A. Sustik, I. S. Dhillon, and P. Ravikumar, "Sparse inverse covariance matrix estimation using quadratic approximation," 2011.

[19] C.-J. Hsieh, M. A. Sustik, I. S. Dhillon, P. Ravikumar, and R. Poldrack, "Big & Quic: Sparse inverse covariance estimation for a million variables," pp. 2339–2347, 2013.

[20] E. Treister and J. Turek, "A block–coordinate descent approach for large–scale sparse inverse covariance estimation," *NIPS*, vol. 27, 2014.

[21] P. A. Olsen, F. Oztoprak, J. Nocedal, and S. J. Rennie, "Newton-like methods for sparse inverse covariance estimation," *NIPS*, 2012.

[22] D. Guillot, B. Rajaratnam, B. T. Rolfs, A. Maleki, and I. Wong, "Iterative thresholding algorithm for sparse inverse covariance estimation," *NIPS*, 2012.

[23] G. Marjanovic and V. Solo, "On exact l_q denoising," *IEEE ICASSP*, pp. 6068–6072, 2013.

[24] ——, "l_q sparsity penalized linear regression with cyclic descent," *IEEE T. Signal Proces.*, vol. 62, no. 6, pp. 1464–1475, 2014.

[25] ——, "On l_q optimization and matrix completion," *IEEE T. Signal Proces.*, vol. 60, no. 11, pp. 5714–5724, 2012.

[26] J. M. Ortega and W. C. Rheinboldt, *Iterative Solutions of Nonlinear Equations in Several Variables*. New York: Academic, 1970.

[27] G. Marjanovic, M. O. Ulfarsson, and A. O. Hero III, "MIST: l_0 Sparse linear regression with momentum," 2015, http://arxiv.org/abs/1409.7193.

[28] MATLAB command: `sparse`, weblink: http://www.mathworks.com/help/matlab/ref/sparse.html.

[29] A. Beck and M. Teboulle, "A fast iterative shrinkage-thresholding algorithm for linear inverse problems," *SIAM J. Imaging Sciences*, vol. 2, no. 1, pp. 183–202, 2009.

[30] L. Li and K. C. Toh, "An inexact interior point method for l_1-regularized sparse covariance selection," *Math. Program. Comp.*, no. 3, pp. 291–315, 2010.

[31] P. Baldi, S. Brunak, Y. Chauvin, C. A. F. Andersen, and H. Nielsen, "Assessing the accuracy of prediction algorithms for classification: An overview," *Bioinformatics*, vol. 16, pp. 412–424, 2000.