A Blockwise Consistency Method for Parameter Estimation of Complex Models

Runmin Shi, Faming Liang *, Ye Luo, Qifan Song and Malay Ghosh

December 13, 2016

Abstract

The drastic improvement in data collection and acquisition technologies has enabled scientists to collect a great amount of data. With the growing dataset size, typically comes a growing complexity of data structures, of the patterns in the data, and of the models needed to account for the patterns. How to estimate the parameters of complex models has put a great challenge on the current statistical methodology. This paper proposes a *blockwise consistency* approach as a potential solution to the problem, which works by iteratively finding consistent estimates for each block of parameters conditional on the current estimates of the parameters in other blocks. The blockwise consistency approach decomposes the high dimensional parameter estimation problem to a series of lower dimensional parameter estimation problems, which often have much simpler structures than the original problem and thus can be easily solved. Moreover, under the framework provided by the blockwise consistency approach, a variety of methods, such as Bayesian and frequentist methods, can be jointly used to achieve a consistent estimator for the original high-dimensional complex model. The blockwise consistency

*To whom correspondence should be addressed: Faming Liang is Professor, Department of Biostatistics, University of Florida, Gainesville, FL 32611, Email: faliang@ufl.edu; Runmin Shi is Graduate Student, Department of Statistics, University of Florida, Gainesville, FL 32611; Ye Luo is Assistant Professor, Department of Economics, University of Florida, Gainesville, FL 32611; Qifan Song is Assistant Professor, Department of Statistics, Purdue University, West Lafayette, IN 47907; Malay Ghosh is Distinguished Professor, University of Florida, Gainesville, FL 32611.

approach is illustrated using two high-dimensional problems, variable selection and multivariate regression. The results of both problems show that the blockwise consistency approach can provide drastic improvements over the existing methods. Extension of the blockwise consistency approach to many other complex models is straightforward.

Keywords:

Coordinate Descent, Gaussian Graphical Model, Multivariate Regression, Precision Matrix, Variable Selection.

1 Introduction

The drastic improvement in data collection and acquisition technologies in the last decades has enabled scientists to collect a great amount of data, such as climate data and high-throughput biological assay data. With the growing dataset size typically comes a growing complexity of data structures, of the patterns in the data, and of the models needed to account for the patterns. Such complex models are often characterized as high dimensional, hierarchical, or highly nonlinear. Among the modern statistical methods, Markov chain Monte Carlo (MCMC) has proven to be very powerful and typically unique computational tools for analyzing data of complex structures. However, the feasibility of MCMC is being challenged in the era of big data, since it typically requires a large number of iterations and a complete scan of the full dataset for each iteration. The frequentist methods, such as maximum likelihood estimation and regularization methods, can be fast, but the optimization problem involved therein is usually difficult to handle when the dimension of the parameter space is high and/or the structure of the model is complicated. How to estimate the parameters of complex models with big data has put a great challenge on the current statistical methodology.

To tackle this problem, we propose a blockwise consistency (BwC) method, which is developed based on the coordinate descent method (Tseng, 2001; Tseng and Yun, 2009). The coordinate descent method has recently been adopted in statistics to solve the parameter estimation problem for sparse linear and logistic regression with convex (Friedman et al., 2010) or nonconvex separa-

ble regularization terms (Breheny and Huang, 2011; Mazumder et al., 2011), as well as the sparse precision matrix (Friedman et al., 2008). The BwC method extends the applications of the coordinate descent method to more general complex statistical models by asymptotically maximizing the following objective function,

$$\max_{\boldsymbol{\theta}} E_{\boldsymbol{\theta}_*} \log \pi(X|\boldsymbol{\theta}), \tag{1}$$

where $\pi(\cdot)$ denotes the likelihood function of a statistical model, $\boldsymbol{\theta}$ denotes a high-dimensional parameter vector, $\boldsymbol{\theta}_*$ denotes the true parameter vector, and $E_{\boldsymbol{\theta}_*}$ denotes expectation with respect to the true likelihood function $\pi(x|\boldsymbol{\theta}_*)$. Suppose $\boldsymbol{\theta}$ has been partitioned into a few blocks $\boldsymbol{\theta} = (\theta^{(1)}, \ldots, \theta^{(k)})$. Then with the new objective function, BwC decomposes the high-dimensional parameter estimation problem to a series of low-dimensional parameter estimation problems: *iteratively finding consistent estimates for the parameters of each block conditioned on the current estimates of the parameters of other blocks*. The low-dimensional problem has often a much simpler structure than the original problem and thus can be easily solved. Moreover, under the framework provided by BwC, a variety of methods, such as Bayesian and frequentist methods, can be jointly used, depending on their availability and convenience, to find a consistent estimate for the original high-dimensional complex model.

This paper demonstrates the use of BwC for two types of complex models. The first one is high-dimensional variable selection in the small-n-large-p scenario, i.e., the sample size is much smaller than the number of variables. For this problem, the dimension of the parameter space can be very high, often ranging from a few thousands to a few millions. Since the problem is ill-posed, regularization methods, e.g., Lasso (Tibshirani, 1996), SCAD (Fan and Li, 2001) and MCP (Zhang, 2010), are often used. However, the performance of these methods can quickly deteriorate as the dimension increases. As shown by our numerical results, BwC can provide a drastic improvement in both parameter estimation and variable selection over regularization methods. The second model is the multivariate high-dimensional regression, where BwC is used to iteratively select relevant variables and estimate the precision matrix. BwC has decomposed the complex problem to two sub-problems, variable selection and precision matrix estimation. Each has a simple structure with a bunch of algorithms available in the literature. The complex problem can then be solved with a

combined use of the existing algorithms. This facilitates big data analysis.

The remainder of this paper is organized as follows. Section 2 describes the BwC method. Section 3 demonstrates the use of BwC for high-dimensional variable selection problems. Section 4 demonstrates the use of BwC for high-dimensional multivariate regression. Section 5 concludes the paper with a brief discussion. In Appendix, we provide the proof for the main theorem.

2 The Blockwise Coordinate Consistency Method

2.1 The BwC method

Suppose that we are interested in estimating the parameters for a complex model with the density function given by $\pi(x|\boldsymbol{\theta})$, where $\boldsymbol{\theta}$ denotes a high-dimensional parameter vector. Without loss of generality, we assume that $\boldsymbol{\theta}$ has been partitioned into k blocks $\boldsymbol{\theta} = (\theta^{(1)}, \ldots, \theta^{(k)})$, and n samples $X_1, \ldots, X_n$ from the model have been observed. For the time being, we assume that $X_1, \ldots, X_n$ are independent and identically distributed (iid), although the independence might not be required for the validity of the BwC method. From Jensen's inequality, we have

$$E_{\boldsymbol{\theta}_*} \log \pi(X|\boldsymbol{\theta}) \leq E_{\boldsymbol{\theta}_*} \log \pi(X|\boldsymbol{\theta}_*). \tag{2}$$

Therefore, finding a consistent estimator of $\boldsymbol{\theta}$ can be viewed as an optimization problem—maximizing $E_{\boldsymbol{\theta}_*} \log \pi(X|\boldsymbol{\theta})$ with respect to $\boldsymbol{\theta}$. In a cyclic rule, BwC works as follows:

1. For each iteration t, set the index $s = t(\text{mod } k) + 1$.

2. For the index s, find an estimator of $\theta_t^{(s)}$ which asymptotically maximizes the objective function

$$W(\theta_t^{(s)}) = E_{\boldsymbol{\theta}_*} \log \pi(X|\hat{\theta}_{t-1}^{(1)}, \ldots, \hat{\theta}_{t-1}^{(s-1)}, \theta_t^{(s)}, \hat{\theta}_{t-1}^{(s+1)}, \ldots, \hat{\theta}_{t-1}^{(k)}) \tag{3}$$

   based on samples $X_1, \ldots, X_n$, where $\hat{\theta}_{t-1}^{(j)}$'s $(j \neq s)$ denote the current estimates and are treated as constants at iteration t. Let $\hat{\theta}_t^{(s)}$ denote the estimator of $\theta^{(s)}$ found at iteration t, and set

$$\hat{\theta}_t^{(j)} = \begin{cases} \hat{\theta}_t^{(s)}, & j = s, \\ \hat{\theta}_{t-1}^{(j)}, & \forall j \neq s. \end{cases}$$

Since maximizing (3) is, in form, equivalent to finding a consistent estimator of $\theta_t^{(s)}$ given the values of the other parameters, we call the method a *blockwise consistency* (BwC) method. The consistent estimators can be found under multiple procedures. For example, Bayesian and frequentist methods can be used to get consistent estimates for different blocks of $\boldsymbol{\theta}$, which leads to an interestingly combined use of Bayesian and frequentist methods.

2.2 Consistency of the BwC method

Let $\tilde{\theta}_t^{(s)} = \arg\max W(\theta_t^{(s)})$ denote the maximizer of $W(\theta_t^{(s)})$. As explained above, $\hat{\theta}_t^{(s)}$ forms a consistent estimate of $\tilde{\theta}_t^{(s)}$. Further, we assume the following condition holds:

(A) $\pi(x|\theta^{(1)}, \ldots, \theta^{(k)})$ is continuous in $\theta^{(i)} \in \Theta_i$ for all $i = 1, 2, \ldots, k$, where Θ_i denotes the parameter space of $\theta^{(i)}$.

Let $\widehat{W}_{n,t} = W(\widehat{\theta}_t^{(s)})$, where the subscript n indicates the dependence of $\hat{\theta}_t^{(s)}$ on the sample size n. Let $W_t^* = \max_{\theta_t^{(s)}} W(\theta_t^{(s)})$. Under condition (A), $\widehat{W}_{n,t}$ forms a consistent estimator of W_t^*, i.e., for any $\epsilon > 0$,

$$P(W_t^* - \widehat{W}_{n,t} > \epsilon) \to 0, \quad \text{as } n \to \infty. \tag{4}$$

To study the convergence of $\widehat{W}_{n,t}$, we also assume

(B) The level set $\Theta_0 = \{\boldsymbol{\theta} : E_{\boldsymbol{\theta}_*} \log \pi(X|\boldsymbol{\theta}) \geq E_{\boldsymbol{\theta}_*} \log \pi(X|\boldsymbol{\theta}_0)\}$ is compact, and $E_{\boldsymbol{\theta}_*} \log \pi(X|\boldsymbol{\theta})$ has at most one maximum in $\theta^{(i)}$ for $i = 1, 2, \ldots, k$.

Let $F_{n,t}(w) = P(\widehat{W}_{n,t} \leq w)$ denote the CDF of $\widehat{W}_{n,t}$. From (4) and the uniqueness of W_t^*, for any $t \geq 1$, we have

$$\lim_{n\to\infty} F_{n,t}(w) = \begin{cases} 1, & w \geq W_t^*, \\ 0, & \text{otherwise.} \end{cases} \tag{5}$$

Let $\tilde{\boldsymbol{\theta}}_t = (\tilde{\theta}_{t-1}^{(1)}, \ldots, \tilde{\theta}_{t-1}^{(s-1)}, \tilde{\theta}_t^{(s)}, \tilde{\theta}_{t-1}^{(s+1)}, \ldots, \tilde{\theta}_{t-1}^{(k)})$, where $\tilde{\theta}_t^{(s)}$ is the maximizer of $W(\theta_t^{(s)})$. Note that in the limit case $n \to \infty$, W_t^* is actually evaluated at $\tilde{\boldsymbol{\theta}}_t$ for each t, i.e., the exact maximum can be obtained at each t. Since the level set Θ_0 is compact, an induction argument on t shows that $\tilde{\boldsymbol{\theta}}_t$ is defined, $W_t^* \geq W_{t-1}^*$, and $\tilde{\boldsymbol{\theta}}_t \in \Theta_0$ for all $t = 0, 1, 2, \ldots$. The same reasoning has also been

used in proof for the convergence of the coordinate descent algorithm (Tseng, 2001). Further, it follows from Jensen's inequality (2) that $\{W_t^*\}$ is upper bounded. By the monotone convergence theorem, for such an increasing and upper bounded sequence, the limit exists. Let W^* denote the supermum of $\{W_t^*\}$, which corresponds to a blockwise maximum point of $E_{\boldsymbol{\theta}_*} \log \pi(X|\boldsymbol{\theta})$. Then $\lim_{t\to\infty} W_t^* = W^*$. Therefore,

$$\lim_{t\to\infty} \lim_{n\to\infty} F_{n,t}(w) = \begin{cases} 1, & w \geq W^*, \\ 0, & \text{otherwise}, \end{cases} \tag{6}$$

which is equivalent to the consistency of $\widehat{W}_{n,t}$; that is, for any $\epsilon > 0$,

$$P(W^* - \widehat{W}_{n,t} > \epsilon) \to 0, \quad \text{as } n \to \infty \text{ and } t \to \infty. \tag{7}$$

As implied by the order of the iterated limits, a large sample size is a prerequisite for the convergence of the algorithm. In summary, we have the following theorem.

Theorem 2.1 *If the conditions* (A) *and* (B) *hold, then* $\widehat{W}_{n,t}$ *converges in probability to a blockwise maximum point as* $n \to \infty$ *and* $t \to \infty$*. Furthermore, if the directional derivatives of* $E_{\boldsymbol{\theta}_*} \log \pi(X|\boldsymbol{\theta})$ *exist for all blocks, then the blockwise maximum point is also a local maximum point.*

Note that the proof of this theorem is independent of the number of blocks and the size of each block. Hence, the algorithm allows the number of parameters to increase with the sample size.

3 BwC for High Dimensional Variable Selection

Consider variable selection for a high dimensional linear regression

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\theta} + \boldsymbol{\epsilon}, \tag{8}$$

where $\boldsymbol{y} = (y_1, \cdots, y_n)^T \in R^n$ is the response vector, n is the sample size, $\boldsymbol{X} = (\boldsymbol{x}_1, \cdots, \boldsymbol{x}_p) \in R^{n\times p}$ is the design matrix, p is the number of explanatory variables (also known as predictors), $\boldsymbol{\theta} \in R^p$ is the vector of unknown regression coefficients, and $\boldsymbol{\epsilon} = (\epsilon_1, \cdots, \epsilon_n)^T \sim N_n(0, \sigma^2 I_n)$ is the Gaussian

random error. Here, we use the bold letter $\boldsymbol{x}_i$ to denote an n-vector realization of the ith predictor, and assume that p can be much greater than n and can also increase with n. Under such a small-n-large-p scenario, identification of true predictors for the model (8) is of particular interest, as this can generally improve the accuracy of prediction for future observations and deepen our understanding to the underlying system.

For this problem, a direct maximization of the objective function (1) will not work, as this will lead to a dense solution due to its ill-posed nature. To resolve this issue, we can treat the problem with a regularization method, i.e., maximizing a penalized log-likelihood function

$$\log \pi(D|\boldsymbol{\theta}) - \sum_{i=1}^{p} P_\lambda(|\theta_i|), \tag{9}$$

where $\pi(D|\boldsymbol{\theta})$ denotes the likelihood function of all observations, $P_\lambda(\cdot)$ is a penalty function, and λ is a tunable parameter and also known as the regularization parameter. The penalty shrinks the parameter space to a smaller one, which is equivalent to "increasing" the sample size. If the penalty function is chosen such that the resulting estimator is asymptotically consistent, then maximizing (9) is approximately equivalent to maximizing (1). In this case, BwC is still applicable. That is, we can partition the parameter vector $\boldsymbol{\theta}$ to a few blocks, and iteratively finding the consistent estimate using the regularization method for each block conditioned on others.

It is known that many choices of the penalty function can lead to consistent solutions to this problem, such as the l_1-penalty used in Lasso (Tibshirani, 1996), the smoothly clipped absolute deviation (SCAD) penalty used in Fan and Li (2001), the minimax concave penalty (MCP) used in Zhang (2010), and the reciprocal l_1-penalty (also known as rLasso) used in Song and Liang (2015a). The basic idea of regularization is to let the penalty $P_\lambda(|\theta_i|)$ compete against the gain in the loglikelihood function by adding the extra predictor $\boldsymbol{x}_i$. For example, for the SCAD penalty, if $\boldsymbol{x}_i$ is added into the true model, then the gain in the loglikelihood function is approximately equivalent to $\sigma^2(\chi^2_{n-|\boldsymbol{t}|} - \chi^2_{n-|\boldsymbol{t}|-1})$ with $\theta_i = O(1/\sqrt{n})$, and the gain is approximately $\sigma^2\chi^2_1$-distributed. Around the origin, the SCAD penalty function is reduced to the linear penalty $P_\lambda(|\theta|) = \lambda|\theta|$. To defeat the gain, the penalty $\lambda|\theta_i|$ needs to be larger than the gain. Therefore, the performance of these methods depends very much on the choice of λ as well as specific features of $\boldsymbol{x}_i$'s, though the methods are

consistent in variable selection under regularity conditions. In particular, the performance of these methods can deteriorate very quickly with an increasing number of predictors. To illustrate this issue, we consider the following example with four different values of $p = 500$, 1000, 2000 and 5000:

$$y_i = \theta_0 + \sum_{j=1}^{p} x_{ij}\theta_j + \epsilon_i, \quad i = 1, 2, \ldots, n, \tag{10}$$

where $n = 100$, and ϵ_i's are iid normal random errors with mean 0 and variance 1. The true value of θ_j's are $\theta_j = 1$ for $j = 1, 2, \ldots, 10$ and 0 otherwise. Let $\boldsymbol{x}_j = (x_{1j}, x_{2j}, \ldots, x_{nj})'$ denote the jth predictor for $j = 1, 2, \ldots, p$, which are given by

$$\boldsymbol{x}_1 = \boldsymbol{z}_1 + \boldsymbol{e}, \quad \boldsymbol{x}_2 = \boldsymbol{z}_2 + \boldsymbol{e}, \ldots, \ldots, \boldsymbol{x}_p = \boldsymbol{z}_p + \boldsymbol{e}, \tag{11}$$

where $\boldsymbol{e}, \boldsymbol{z}_1, \ldots, \boldsymbol{z}_p$ are iid normal random vectors drawn from $N(0, I_n)$. For each value of p, 10 datasets are independently generated.

The regularization methods, including Lasso, SCAD and MCP, were applied to the datasets. These methods have been implemented in the R-package *SIS* (Fan et al., 2015), and were run under their default settings: The variables were first scanned according to the iterative sure independence screening (ISIS) algorithm (Fan et al., 2009), and then selected from the remaining set by maximizing (9) with their respective penalties. The regularization parameters were determined according to the BIC criterion. To measure the performance of these methods in variable selection, we calculate the false and negative selection rates. Let $\boldsymbol{s}_*$ denote the set of true variables, and let $\hat{\boldsymbol{s}}$ denote the set of selected variables. Define

$$\text{fsr} = \frac{|\hat{\boldsymbol{s}} \setminus \boldsymbol{s}_*|}{|\hat{\boldsymbol{s}}|}, \quad \text{nsr} = \frac{|\boldsymbol{s}_* \setminus \hat{\boldsymbol{s}}|}{|\boldsymbol{s}_*|}, \tag{12}$$

where $|\cdot|$ denotes the set cardinality. The smaller the values of fsr and nsr are, the better the performance of the method is. The numerical results are summarized in Table 1. It is easy to see that for all the methods, the performance deteriorates very quickly: All the values of fsr, nsr and parameter estimation error increase with p. These methods have also been tried with the tuning parameters determined via cross-validation. The results are even worse than those reported in Table 1.

Table 1: Performance of the regularization methods for the simulated data with different values of p: $|\hat{\boldsymbol{s}}|_{avg}$ denotes the average number of variables selected for 10 datasets, fsr and nsr are calculated in (12), $\|\hat{\boldsymbol{\theta}} - \boldsymbol{\theta}_*\| = \sqrt{\sum_{j=1}^{p}(\hat{\theta}_j - \theta_{*j})^2}$ measures the parameter estimation error, and the numbers in parenthesis denote the standard deviations of the corresponding estimators.

		Lasso	SCAD	MCP
p=500	$\|\hat{\boldsymbol{s}}\|_{avg}$	20.4 (0.60)	16.6 (1.80)	15.7 (1.16)
	$\|\hat{\boldsymbol{s}} \cap \boldsymbol{s}_*\|_{avg}$	9.8 (0.20)	10 (0)	10 (0)
	$\|\hat{\boldsymbol{\theta}} - \boldsymbol{\theta}_*\|_{avg}$	0.86 (0.103)	0.75 (0.116)	0.82 (0.080)
	fsr	0.51 (0.022)	0.31 (0.086)	0.32 (0.059)
	nsr	0.02 (0.020)	0 (0)	0 (0)
p=1000	$\|\hat{\boldsymbol{s}}\|_{avg}$	21 (0)	19.6 (1.11)	18.2 (0.65)
	$\|\hat{\boldsymbol{s}} \cap \boldsymbol{s}_*\|_{avg}$	7.8 (0.85)	8.9 (0.67)	9 (0.73)
	$\|\hat{\boldsymbol{\theta}} - \boldsymbol{\theta}_*\|_{avg}$	1.73 (0.323)	1.44 (0.314)	1.41 (0.325)
	fsr	0.63 (0.041)	0.52 (0.067)	0.49 (0.050)
	nsr	0.22 (0.085)	0.11 (0.067)	0.10 (0.073)
p=2000	$\|\hat{\boldsymbol{s}}\|_{avg}$	21 (0)	20.9 (0.10)	20.4 (0.43)
	$\|\hat{\boldsymbol{s}} \cap \boldsymbol{s}_*\|_{avg}$	6.1 (0.92)	6.6 (0.85)	6.7 (0.86)
	$\|\hat{\boldsymbol{\theta}} - \boldsymbol{\theta}_*\|_{avg}$	2.42 (0.345)	2.50 (0.333)	2.51 (0.340)
	fsr	0.71 (0.044)	0.68 (0.041)	0.66 (0.049)
	nsr	0.39 (0.092)	0.34 (0.085)	0.33 (0.086)
p=5000	$\|\hat{\boldsymbol{s}}\|_{avg}$	21 (0)	19.9 (1.10)	20.3 (0.70)
	$\|\hat{\boldsymbol{s}} \cap \boldsymbol{s}_*\|_{avg}$	3.7 (0.79)	5.4 (0.93)	5.2 (0.95)
	$\|\hat{\boldsymbol{\theta}} - \boldsymbol{\theta}_*\|_{avg}$	3.31 (0.235)	2.96 (0.414)	3.01 (0.390)
	fsr	0.82 (0.037)	0.69 (0.085)	0.73 (0.062)
	nsr	0.63 (0.079)	0.46 (0.093)	0.48 (0.095)

Note that for this example, all the three methods selected about 21 variables, which is approximately equal to $n/log(n)$, the upper bound of the model size set by ISIS. Without this upper bound, more variables will be selected. For example, we have also implemented Lasso using the R-package *glmnet* with the regularization parameter tuned via cross-validation. The average number of variables selected by Lasso for the datasets with $p = 5000$ is 72.7 with standard deviation 3.05.

The difficulty suffered by the regularization methods, whose performance deteriorates with dimensions, can be alleviated using the BwC method. BwC decomposes the high-dimensional variable selection problem to a series of lower dimensional variable selection problems, which, as a return, increases the accuracy of variable selection and parameter estimation.

3.1 An Illustrative Example

To illustrate the performance of BwC, we use the datasets of $p = 5000$ generated above. We first consider a naive version of BwC before giving a sophisticated one. For this naive version of BwC, we had the datasets slightly modified: we exchanged the positions of some variables in the dataset such that the 10 true variables are positioned as $\{1, 2, 1001, 1002, 2001, 2002, 3001, 3002, 4001, 4002\}$. Then BwC was implemented as follows:

1. Split the predictors into 5 blocks: $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{1000}\}$, $\{\boldsymbol{x}_{1001}, \ldots, \boldsymbol{x}_{2000}\}$, $\{\boldsymbol{x}_{2001}, \ldots, \boldsymbol{x}_{3000}\}$, $\{\boldsymbol{x}_{3001}, \ldots, \boldsymbol{x}_{4000}\}$, and $\{\boldsymbol{x}_{4001}, \ldots, \boldsymbol{x}_{5000}\}$.

2. Conduct variable selection using Lasso for each block independently, and combine the selected predictors to get an initial estimate of $\boldsymbol{\theta}$.

3. Conduct blockwise conditional variable selection using MCP for 25 sweeps. Here a sweep refers to a cycle of updates for all blocks.

In this implementation, MCP is used as the block consistent estimator because MCP itself is consistent for high dimensional variable selection. For MCP, we let the regularization parameter be determined according to the BIC criterion, and other parameters be set to the default value as given in the *R*-package *SIS*. To save computational time, we set the number of iterations in

the sure independence screening (SIS) step to be 1. However, multiple iterations will not hurt the performance of BwC except for the CPU time. Figure 1 shows the convergence path of BwC for one dataset. For this dataset, BwC converges to a model with 11 variables, including 10 true variables and one false variable, and the estimation error of $\boldsymbol{\theta}$ is about 0.5. BwC can converge very fast, usually within 5 sweeps. Other results are summarized in Table 2.

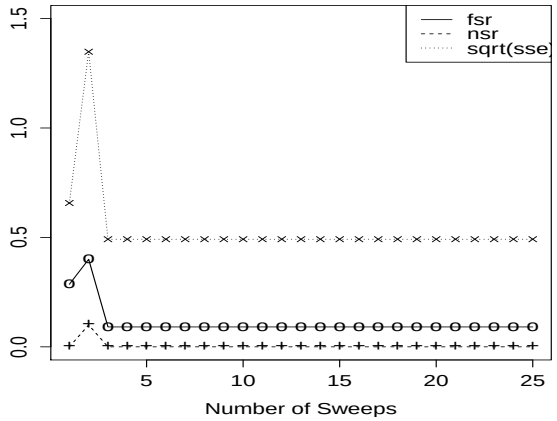

Figure 1: Convergence path of BwC for one simulated dataset with $n = 100$ and $p = 5000$, where fsr, nsr, and sqrt(sse) denote the false selection rate, negative selection rate, and parameter estimation error $\|\hat{\boldsymbol{\theta}} - \boldsymbol{\theta}_*\|$, respectively.

Table 2: Comparison of BwC, Lasso, SCAD and MCP for the simulated example with $p = 5000$. Refer to Table 1 for the notation.

	Lasso	SCAD	MCP	BwC
$\lvert\hat{\boldsymbol{s}}\rvert_{avg}$	21(0.0)	19.9(1.10)	20.3(0.70)	12.8(0.36)
$\lvert\hat{\boldsymbol{s}} \cap \boldsymbol{s}_*\rvert_{avg}$	3.7 (0.79)	5.4 (0.93)	5.2 (0.95)	10(0)
$\|\hat{\boldsymbol{\theta}} - \boldsymbol{\theta}_*\|_{avg}$	3.31 (0.24)	2.96 (0.41)	3.01 (0.39)	0.655 (0.05)
fsr	0.824	0.729	0.744	0.219
nsr	0.63	0.46	0.48	0

For comparison, the results of Lasso, SCAD and MCP were also included in the table. The comparison shows that BwC has made a drastic improvement over Lasso, SCAD and MCP in both variable selection and parameter estimation. On average, BwC selected only 12.8 variables for each dataset, without missing any true variables. Recall that the number of true variables is 10 and all the variables are highly correlated with a mutual correlation coefficient of 0.5. SCAD and MCP selected about 20 variables for each dataset, missing about half of the true variables. Lasso performs even worse than SCAD and MCP, it missed over 60% of the true variables. In different runs of BwC, SCAD and Lasso were used as the block consistent estimator. SCAD produced almost the same results as MCP, while Lasso did not. This is because Lasso is not consistent for variable selection unless the strongly irrepresentable condition holds (Zhao and Yu, 2006; Zou, 2006), and such a condition is hard to be satisfied by this example due to the high correlation between the predictors.

BwC works extremely well for this example, which has been designed for an ideal scenario that each block contains some significant variables. Here the significant variables can be understood as the true variables or the variables that are significantly correlated with the response variable. Since variable selection is to distinguish significant variables from other variables, the existence of significant variables in each block can reasonably accelerate the convergence of BwC by avoiding selecting too many noisy variables from empty blocks. A block is called empty if it contains no significant variables. To achieve such an ideal scenario, we propose a balanced data partitioning scheme as described in the next subsection. Along with the description, we discussed how to choose the number of blocks.

3.2 BwC with a Balanced Data Partitioning Scheme

In the initial step of BwC, variable selection was done for each block independently. Since for each block, the model can be misspecified with some true variables missed from the set of available variables. As shown by Song and Liang (2015b), for each block, the variables found in this initial step include the true variables as well as the surrogate variables for the missed true variables in the block. Hence, we can expect that the total number of variables selected in the initial step is

large. Then, via conditional updating, some surrogate variables can be deleted. Therefore, the number of selected variables has a trend to decrease with sweeps. To achieve the goal of balanced data partitioning, we propose to re-distribute the selected variables to different blocks in a balanced manner, i.e., each block is assigned about the same number of selected variables. For convenience, we call a stage of the algorithm for each re-distribution of selected variables. To accommodate the decreasing trend of selected variables, we let the number of blocks be diminished with stages. In summary, we have the following algorithm:

Algorithm 3.1 *BwC for High-Dimensional Regression*

1. *Split the predictors into K blocks, conduct variable selection using Lasso for each block independently, and combine the selected predictors to get an initial estimate of $\boldsymbol{\theta}$.*

2. *Conduct blockwise conditional variable selection using MCP for m sweeps.*

3. *Reduce the number of blocks K by s, which is called the block diminishing size, and re-distribute the variables selected in step 2 to each block in a balanced manner.*

4. *Go to step 2 unless the number of blocks has been small enough, say 2.*

5. *Output the final model according to a pre-specified criterion such as sparsity, prediction error, BIC or EBIC (Chen and Chen, 2008).*

In practice, we often set $K = 10$ or larger, with 10 as the default value. Since the performance of the regularization methods can deteriorate with the dimension, see Table 1, a small initial block size is generally preferred. On the other hand, an extremely small initial block size can result in a large set of significant variables identified in the initial step, which, in turn, slows down the convergence of BwC. As a trade-off, we suggest to set the initial block size to be 1000 or less. Since BwC can converge pretty fast for a given partition, we set $m = 25$ as the default number of sweeps, which has been extremely conservative according to our experience. Also, we point out that the MCP algorithm, implemented in the package *SIS*, assumes that significant variables can be found for each regression. We modified the code slightly such that it allows for the return of null models.

For the block diminishing size s, we set 2 as the default value, but 1 and 3 are also often used. In distributing selected variables to different blocks, we often have the unselected variables shuffled and redistributed, though this is unnecessary. The shuffling and redistributing step helps to maintain the same block sizes as stages proceed. For BwC, we often output the sparsest model as the final model, which is set as the default criterion for model selection. Other criteria, such as prediction error, BIC and EBIC, can also be used as the criterion for final model selection. In what follows, BwC will be run with default settings unless stated otherwise.

To illustrate the performance of BwC with the balanced data partitioning scheme, we considered a modified example from the one used in Section 3.1. Here we considered four settings of the true model with the true model size $|\boldsymbol{s}_*| = 5$, 10, 15 and 20, respectively. Under each setting, the positions of the true variables are uniformly distributed among the first 3500 variables out of $p = 5000$ in total. This mimics the situation that empty blocks exist in the early stages of BwC. To accommodate the large true model size, we increase the sample size n from 100 to 200. For each true model size, we generated 10 datasets independently. To measure the prediction performance of the selected model, we generated additional 1000 observations as the test set for each of the 40 datasets.

BwC was applied to these datasets under the default setting. The results are summarized in Figure 2 and Table 3. Figure 2 shows the number of selected variables and the sum of squared prediction errors produced by BwC for 4 datasets, each with different true model size, along with the number of blocks. For the datasets with $|\boldsymbol{s}_*| = 5$, 10, 15 and 20, the sparsest models were produced at $K = 2$, 4, 6 and 6, respectively. In terms of variable selection, the results are surprisingly good: The sparsest models produced by BwC are almost identical to the true models. It is interesting to see that the model sizes produced by BwC with block numbers form a U-shape curve, where the sparsest model (usually the true model) is located at the bottom of the curve. For the dataset with $|\boldsymbol{s}_*| = 5$, although the size of the models produced by BwC tends to decrease with K, it still forms a U-curve if the non-blocking case (i.e., $K = 1$) is included. From this U-curve, the optimal block number can be automatically determined. For these datasets, it is easy to see that the curves of prediction error match well with the curves of model size.

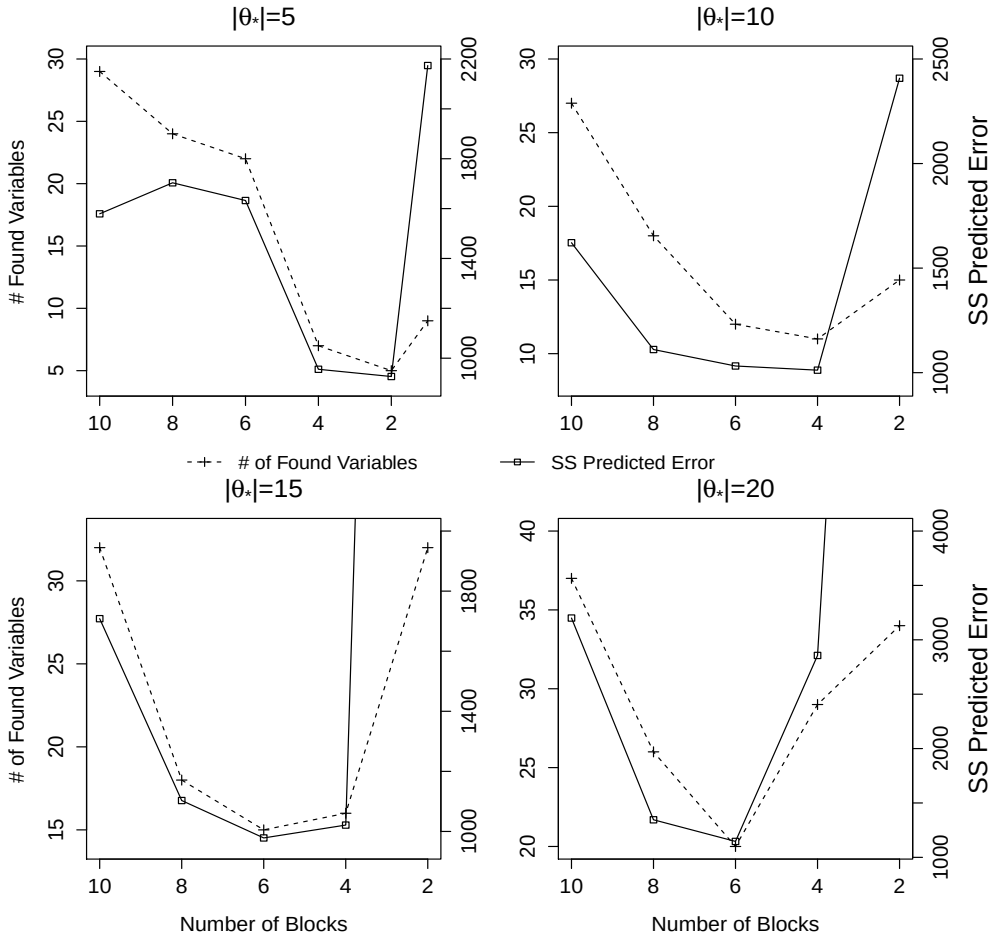

Figure 2: The number of selected variables and the sum of squared predicted errors along with the diminishing number of blocks

Table 3 summarizes the results of BwC for the 40 datasets. For comparison, the regularization methods, Lasso, SCAD and MCP, were also applied to these datasets. As for the previous examples, these methods were run with the package *SIS* under their default settings except that the regularization parameters were determined according to the BIC criterion. The comparison indicates that BwC is superior to these regularization methods in both variable selection and parameter estimation for all the four cases. The regularization methods tend to select more false variables and miss more true variables as $|\boldsymbol{s}_*|$ increases, while BwC can consistently identify almost all true variables.

Table 3 also reports the CPU time (in seconds) cost by each method for a single dataset. The comparison indicates that BwC does not cost much more CPU time than SCAD and MCP. Note that in the package *SIS*, all the methods, Lasso, SCAD and MCP were implemented with a variable screening step. Under their default setting, the variable screening step is done with the ISIS algorithm, which may take quite a few iterations to converge for some datasets. For this reason, SCAD, MCP and BwC take comparable CPU time. Compared to SCAD and MCP, Lasso tends to select more variables at each iteration. As a consequence, it often takes less iterations to converge and thus costs less CPU time.

For the model with $|\boldsymbol{s}_*| = 10$, we have considered two sample sizes $n = 100$ and 200 with the results reported in Table 2 and Table 3, respectively. A comparison of the two tables show that the performance of all methods has been improved as the sample size increases. For BwC, the average model size $|\hat{\boldsymbol{s}}|_{avg}$ is closer to the true model size and the value of fsr is further reduced as the sample size increases. For the regularization methods, both fsr and nsr have been reduced, although the average model size is still very large. This comparison indicates that the performance of BwC is much less dependent on the sample size than the regularization method.

For a thorough comparison of BwC with the regularized methods, we also consider the scenario where the mutual correlation coefficient between different predictors is low. For this purpose, we generated another 40 datasets according to (10) and (11) with the variance of $\boldsymbol{e}$ tuned such that the mutual correlation coefficient is equal to 0.2. The results are summarized in Table 4. BwC also significantly outperforms the regularization methods under this scenario.

Table 3: Numerical results of BwC, Lasso, SCAD and MCP for the simulated data with the mutual correlation coefficient $\rho = 0.5$: The CPU time (in seconds) is measured for a single run on an Intel Core i7-4790@3.60GHz Quad-Core desktop. Refer to Table 1 for other notations.

		Lasso	SCAD	MCP	BwC
$\lvert\boldsymbol{s}_*\rvert = 5$	$\lvert\hat{\boldsymbol{s}}\rvert_{avg}$	18.9 (3.07)	20.6 (4.37)	21.5 (0.77)	6.7 (0.30)
	$\lvert\hat{\boldsymbol{s}} \cap \boldsymbol{s}_*\rvert_{avg}$	5 (0)	5 (0)	5 (0)	5 (0)
	$\lVert\hat{\boldsymbol{\theta}} - \boldsymbol{\theta}_*\rVert_{avg}$	0.50 (0.065)	0.70 (0.136)	0.94 (0.033)	0.33 (0.029)
	fsr	0.69 (0.032)	0.50 (0.137)	0.75 (0.010)	0.24 (0.036)
	nsr	0 (0)	0 (0)	0 (0)	0 (0)
	CPU time	5.41 (1.26)	18.94 (4.43)	20.53 (1.60)	30.79 (0.41)
$\lvert\boldsymbol{s}_*\rvert = 10$	$\lvert\hat{\boldsymbol{s}}\rvert_{avg}$	35.3 (1.70)	28.4 (4.04)	25.4 (1.27)	12.7 (0.56)
	$\lvert\hat{\boldsymbol{s}} \cap \boldsymbol{s}_*\rvert_{avg}$	10 (0)	10 (0)	10 (0)	9.9 (0.10)
	$\lVert\hat{\boldsymbol{\theta}} - \boldsymbol{\theta}_*\rVert_{avg}$	0.75 (0.048)	0.81 (0.128)	0.98 (0.056)	0.50 (0.077)
	fsr	0.71 (0.023)	0.51 (0.111)	0.60 (0.022)	0.21 (0.038)
	nsr	0 (0)	0 (0)	0 (0)	0.01 (0.01)
	CPU time	10.14 (0.89)	23.28 (3.50)	26.66 (1.34)	31.43 (0.36)
$\lvert\boldsymbol{s}_*\rvert = 15$	$\lvert\hat{\boldsymbol{s}}\rvert_{avg}$	37 (0)	34.3 (2.20)	30.3 (0.75)	17.5 (0.62)
	$\lvert\hat{\boldsymbol{s}} \cap \boldsymbol{s}_*\rvert_{avg}$	12.5 (1.21)	14.5 (0.50)	15 (0)	15 (0)
	$\lVert\hat{\boldsymbol{\theta}} - \boldsymbol{\theta}_*\rVert_{avg}$	1.70 (0.430)	1.12 (0.258)	1.00 (0.039)	0.46 (0.024)
	fsr	0.66 (0.032)	0.54 (0.062)	0.50 (0.012)	0.13 (0.028)
	nsr	0.17 (0.081)	0.03 (0.033)	0 (0)	0 (0)
	CPU time	16.97 (2.81)	24.46 (2.23)	32.78 (0.31)	31.98 (0.11)
$\lvert\boldsymbol{s}_*\rvert = 20$	$\lvert\hat{\boldsymbol{s}}\rvert_{avg}$	37 (0)	37 (0)	36.5 (0.34)	24 (0.71)
	$\lvert\hat{\boldsymbol{s}} \cap \boldsymbol{s}_*\rvert_{avg}$	9 (1.42)	11.6 (1.69)	12.7 (1.69)	19.9 (0.10)
	$\lVert\hat{\boldsymbol{\theta}} - \boldsymbol{\theta}_*\rVert_{avg}$	4.04 (0.404)	3.62 (0.493)	3.36 (0.527)	0.65 (0.106)
	fsr	0.76 (0.038)	0.69 (0.046)	0.65 (0.050)	0.16 (0.026)
	nsr	0.55 (0.071)	0.42 (0.085)	0.37 (0.084)	0.01 (0.005)
	CPU time	13.49 (1.96)	17.37(1.59)	24.45 (2.57)	32.64(0.52)

Table 4: Numerical results of BwC, Lasso, SCAD and MCP for the simulated data with the mutual correlation coefficient $\rho = 0.2$. Refer to Table 3 for the notation.

		Lasso	SCAD	MCP	BwC
$\lvert \boldsymbol{s}_* \rvert = 5$	$\lvert \hat{\boldsymbol{s}} \rvert_{avg}$	22.6 (3.97)	11.5 (4.25)	17.8 (4.34)	5.6 (0.31)
	$\lvert \hat{\boldsymbol{s}} \cap \boldsymbol{s}_* \rvert_{avg}$	5 (0)	5 (0)	5 (0)	5 (0)
	$\lVert \hat{\boldsymbol{\theta}} - \boldsymbol{\theta}_* \rVert_{avg}$	0.58 (0.059)	0.34 (0.096)	0.60 (0.126)	0.24 (0.031)
	fsr	0.71 (0.048)	0.19 (0.113)	0.43 (0.134)	0.09 (0.040)
	nsr	0 (0)	0 (0)	0 (0)	0 (0)
$\lvert \boldsymbol{s}_* \rvert = 10$	$\lvert \hat{\boldsymbol{s}} \rvert_{avg}$	37 (0)	12.8 (2.69)	28.7 (4.02)	11.4 (0.43)
	$\lvert \hat{\boldsymbol{s}} \cap \boldsymbol{s}_* \rvert_{avg}$	9.9 (0.1)	10 (0)	10 (0)	10 (0)
	$\lVert \hat{\boldsymbol{\theta}} - \boldsymbol{\theta}_* \rVert_{avg}$	0.90 (0.075)	0.32 (0.063)	0.82 (0.120)	0.34 (0.034)
	fsr	0.73 (0.003)	0.08 (0.073)	0.52 (0.107)	0.11 (0.031)
	nsr	0.01 (0)	0.01 (0)	0 (0)	0 (0)
$\lvert \boldsymbol{s}_* \rvert = 15$	$\lvert \hat{\boldsymbol{s}} \rvert_{avg}$	37 (0)	30.4 (3.36)	32.6 (2.93)	17.2 (0.53)
	$\lvert \hat{\boldsymbol{s}} \cap \boldsymbol{s}_* \rvert_{avg}$	15 (0)	14.6 (0.40)	14.6 (0.40)	15 (0)
	$\lVert \hat{\boldsymbol{\theta}} - \boldsymbol{\theta}_* \rVert_{avg}$	0.83 (0.030)	0.88 (0.262)	1.05 (0.248)	0.45 (0.024)
	fsr	0.59 (0)	0.43 (0.094)	0.49 (0.082)	0.12 (0.026)
	nsr	0 (0)	0.03 (0.027)	0.03 (0.027)	0 (0)
$\lvert \boldsymbol{s}_* \rvert = 20$	$\lvert \hat{\boldsymbol{s}} \rvert_{avg}$	37 (0)	33.6 (2.27)	37 (0)	23.8 (1.44)
	$\lvert \hat{\boldsymbol{s}} \cap \boldsymbol{s}_* \rvert_{avg}$	10.8 (1.44)	14.6 (1.92)	14.6 (1.81)	19.9 (0.10)
	$\lVert \hat{\boldsymbol{\theta}} - \boldsymbol{\theta}_* \rVert_{avg}$	3.57 (0.377)	2.49 (0.612)	2.87 (0.570)	0.63 (0.134)
	fsr	0.71 (0.039)	0.51 (0.097)	0.62 (0.049)	0.14 (0.044)
	nsr	0.46 (0.072)	0.27 (0.096)	0.30 (0.091)	0.01 (0.005)

3.3 A Real Data Study: Biomarker Discovery for Anticancer Drug Sensitivity

Complex diseases such as cancer often have significant heterogeneity in response to treatments. Hence, individualized treatment based on the patient's prognostic or genomic data rather than a "one treatment fits all" approach could lead to significant improvement of patient care. The success of personalized medicine decisively depends on the accuracy of the diagnosis and outcome prediction (Hamburg and Collins, 2010), and thus the discovery of precise biomarkers for disease is essential. Recent advances in high-throughput biotechnologies, such as microarray, sequencing technologies and mass spectrometry, have provided an unprecedented opportunity for biomarker discovery. Given the high dimensionality of the omics data, biomarker discovery is best cast as variable selection from a statistical point of view. In this study, we consider the problem of identifying the genes that predict the anticancer drug sensitivity for different cell lines, toward the ultimate goal of selecting right drugs for individual patients.

We considered a dataset extracted from the Cancer Cell Line Encyclopedia (CCLE) database. The dataset include the sensitivity data of 474 cell lines to the drug topotecan, as well as the expression data of 18,926 genes for each cell line. Topotecan is a chemotherapeutic agent that is a topoisomerase inhibitor, and it has been used to treat lung cancer, ovarian cancer, and other types cancer. The drug sensitivity was measured using the area under dose-response curves, which is termed as activity area in (Barretina et al., 2012). For this dataset, BwC was run with the initial block number $K = 20$ and the block diminishing size $s = 3$. To have a thorough exploration for this dataset, BwC was run 25 times. Note that the models selected by BwC may be different in different runs due to its stochastic nature in redistributing significant genes to different blocks. The average number of genes found by BwC is 9.88 with standard deviation 0.40. Table 5 lists the top 10 genes identified by BwC in the order of frequencies appeared in the 25 final models. The appearance frequency provides a simple measure for the importance of each gene to prediction of drug sensitivity by integrating over multiple models.

Our result is consistent with the current biological knowledge. For example, the first gene

Table 5: Top 10 genes selected by BwC for the drug topotecan based on 25 independent runs.

No.	Gene	Frequency	No.	Gene	Frequency
1	SLFN11	25	6	MFAP2	6
2	HSPB8	11	7	RFXAP	6
3	C14orf93	8	8	ANO10	5
4	ILF3	7	9	AGPAT5	5
5	C15orf57	6	10	RPL3	5

SLFN11 is a helicase that breaks apart annealed nucleic acid strands. Helicases are known to work hand in hand with topoisomerase at many major DNA events (Duguet, 1997). It was also found by Barretina et al. (2012) that SLFN11 is predictive of treatment response for Topotecan, as well as irinotecan, another inhibitor of TOP-I. It is known that the gene HSPB8 interacts with the gene HSPB1, and the recent study (Li et al., 2016) shows that HSPB1 polymorphisms might be associated with radiation-induced damage risk in lung cancer patients treated with radiotherapy. The gene ILF3 is also related to lung cancer, which shows correlated mRNA and protein overexpression in lung cancer development and progression (Guo et al., 2008). The relationship of other genes to the drug sensitivity will be further explored elsewhere.

For comparison, Lasso, SCAD and MCP have also been applied to this dataset. Similar to simulated examples, they tend to select a larger model than BwC. Lasso selected 18 genes, MCP selected 21 genes, and SCAD selected 52 genes, which are all significantly larger than the average model size produced by BwC.

4 BwC for High Dimensional Multivariate Regression

Multivariate regression is a generalized regression model of regressing $q > 1$ responses on p predictors. Applications of this generalized model often arise in biomedicine, psychometrics, and many other quantitative disciplines. Let $\boldsymbol{X} \in \mathbb{R}^{n\times p}$ denote the design matrix, let $\boldsymbol{Y} \in \mathbb{R}^{n\times q}$ denote the

response matrix, let $\boldsymbol{B} \in \mathbb{R}^{p\times q}$ denote the regression coefficient matrix, and let $\boldsymbol{E} \in \mathbb{R}^{n\times q}$ denote the random error matrix. Then the model can be written as

$$\boldsymbol{Y} = \boldsymbol{X}\boldsymbol{B} + \boldsymbol{E}. \tag{13}$$

For convenience, we let $\boldsymbol{y}_i$ and $\boldsymbol{y}_{(i)}$ denote the ith column and ith row of $\boldsymbol{Y}$, respectively. Likewise, we let $\boldsymbol{\epsilon}_i$ and $\boldsymbol{\epsilon}_{(i)}$ denote the ith column and ith row of $\boldsymbol{E}$, respectively. In general, it is assumed that $\boldsymbol{\epsilon}'_{(i)}$ follows a multivariate normal distribution $N(0, \Sigma)$, while $\boldsymbol{\epsilon}'_{(1)}, \ldots, \boldsymbol{\epsilon}'_{(n)}$ are mutually independent. We are interested in jointly estimating the regression coefficient matrix $\boldsymbol{B}$ and the precision matrix $\boldsymbol{\Omega} = \Sigma^{-1}$, in particular, when q and/or p are greater than n.

In the literature, there are several papers on variable selection for multivariate linear regression, see e.g., Turlach et al. (2005) and Peng et al. (2010), under the assumption that the response variables are mutually independent. There are also several methods for precision matrix estimation, such as gLasso (Yuan and Lin, 2007; Friedman et al., 2008), nodewise regression (Meinshausen and Bühlmann, 2006), and ψ-learning (Liang et al., 2015). However, the papers which address simultaneous variable selection and precision matrix estimation in the context of multiple regression are sparse. Three exceptions are Rothman et al. (2010), Bhadra and Mallick (2013) and Sofer et al. (2014). Rothman et al. (2010) proposed an iterative approach that alternatively estimates the precision matrix and regression coefficients under a l_1-penalty, but without any theoretical guarantees provided for the convergence of their algorithm. Bhadra and Mallick (2013) proposed a Bayesian approach to the problem, but again no convergence results were proved. Sofer et al. (2014) proves the convergence of the iterative approach of Rothman et al. (2010) under very restrictive conditions, such as $pq/n \to 0$ and $q^2/n \to 0$, which require both p and q to be much smaller than n.

The BwC method can be naturally applied to this problem by treating $\boldsymbol{B}$ and $\boldsymbol{\Omega}$ as two separated parameters blocks. It is easy to verify that the conditions (A) and (B) are satisfied for this problem. Therefore, BwC provides a theoretical guarantee for the convergence of the iterative approach used in Rothman et al. (2010) and Sofer et al. (2014). More importantly, the theory of BwC allows both q and p to be greater than the sample size n. For the problems for which p is extremely large, BwC

can be run in a nested manner: conditioned on the estimate of $\boldsymbol{\Omega}$, BwC can again be used as a subroutine for estimation of $\boldsymbol{B}$. In what follows we first give a brief review of the iterative approach used in Rothman et al. (2010) in order to distinguish it from BwC.

4.1 A Brief Review of the Existing Iterative Approach

Consider the multivariate regression model (13). With the regularization methods, a natural way is to minimize the objective function

$$Q(\boldsymbol{B}, \boldsymbol{\Omega}) = -\ln|\boldsymbol{\Omega}| + trace\left[\frac{1}{n}(\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{B})^T(\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{B})\boldsymbol{\Omega}\right] + P_\lambda(\boldsymbol{B}) + P_\gamma(\boldsymbol{\Omega}), \tag{14}$$

where $P_\lambda(\cdot)$ and $P_\gamma(\cdot)$ are the penalty functions for $\boldsymbol{B}$ and $\boldsymbol{\Omega}$, respectively; and λ and γ are regularization parameters. In practice, $P_\lambda(\cdot)$ can be set to a standard penalty function, such as the one used in Lasso, SCAD or MCP; and $P_\gamma(\cdot)$ can be set to the l_1-penalty as the one used in graphical Lasso (Friedman et al., 2008).

To minimize $Q(\boldsymbol{B}, \boldsymbol{\Omega})$ simultaneously over $\boldsymbol{B}$ and $\boldsymbol{\Omega}$, Rothman et al. (2010) proposed an iterative approach, the so-called multivariate regression with covariance estimation (MRCE) algorithm, which is actually a block coordinate descent algorithm. Let $\hat{\boldsymbol{\Omega}}$ and $\hat{\boldsymbol{B}}$ denote the current estimators of $\boldsymbol{\Omega}$ and $\boldsymbol{B}$, respectively. One iteration of the MRCE algorithm consist of two general steps:

Algorithm 4.1 *MRCE Algorithm*

(i) Estimate $\boldsymbol{B}$ by minimizing $Q(\boldsymbol{B}|\hat{\boldsymbol{\Omega}})$ conditional on the current estimator of $\boldsymbol{\Omega}$, where

$$Q(\boldsymbol{B}|\hat{\boldsymbol{\Omega}}) = trace\left[\frac{1}{n}(\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{B})^T(\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{B})\hat{\boldsymbol{\Omega}}\right] + P_\lambda(\boldsymbol{B}). \tag{15}$$

(ii) Estimate $\boldsymbol{\Omega}$ by minimizing $Q(\boldsymbol{\Omega}|\hat{\boldsymbol{B}})$ conditional on the current estimator of $\boldsymbol{B}$, where

$$Q(\boldsymbol{\Omega}|\hat{\boldsymbol{B}}) = -\ln|\boldsymbol{\Omega}| + trace\left[\frac{1}{n}(\boldsymbol{Y} - \boldsymbol{X}\hat{\boldsymbol{B}})^T(\boldsymbol{Y} - \boldsymbol{X}\hat{\boldsymbol{B}})\boldsymbol{\Omega}\right] + P_\gamma(\boldsymbol{\Omega}). \tag{16}$$

Step (ii) can be accomplished using the graphical Lasso algorithm. To accomplish step (i), Rothman et al. (2010) suggested to set a Lasso penalty for $\boldsymbol{B}$, i.e., set

$$P_\lambda(\boldsymbol{B}) = \lambda \sum_{i=1}^{q}\sum_{j=1}^{p}|\beta_{ji}|,$$

and then apply the cyclical coordinate descent algorithm to solve (15) for $\boldsymbol{B}$. Let $\boldsymbol{S} = \boldsymbol{X}^T\boldsymbol{X}$ and $\boldsymbol{H} = \boldsymbol{X}^T\boldsymbol{Y}\boldsymbol{\Omega}$, let $\hat{\boldsymbol{B}}^{(m)}$ denote the estimate of $\boldsymbol{B}$ at the m-th iteration, and let $\boldsymbol{B}^{RIDGE} = (\boldsymbol{X}^T\boldsymbol{X}+\lambda I)^{-1}\boldsymbol{X}^T\boldsymbol{Y}$ denote the ridge penalized least-square estimator of $\boldsymbol{B}$. The cyclical coordinate descent algorithm is described as follows:

Algorithm 4.2 *Cyclical coordinate descent algorithm for multivariate regression*

1. *Set* $\hat{\boldsymbol{B}}^{(m)} \leftarrow \hat{\boldsymbol{B}}^{(m-1)}$*, visit all entries of* $\hat{\boldsymbol{B}}^{(m)}$ *in some sequence and for entry* (r, c) *update* $\hat{\beta}_{rc}^{(m)}$ *with the minimizer of the objective function along its coordinate direction given by*

$$\hat{\beta}_{rc}^{(m)} \leftarrow sign\left(\hat{\beta}_{rc}^{(m)} + \frac{h_{rc} - u_{rc}}{s_{rr}\omega_{rr}}\right)\left(\left|\hat{\beta}_{rc}^{(m)} + \frac{h_{rc} - u_{rc}}{s_{rr}\omega_{cc}}\right| - \frac{n\lambda}{s_{rr}\omega_{cc}}\right)_+,$$

   where $u_{rc} = \sum_{j=1}^{p}\sum_{k=1}^{q}\hat{\beta}_{jk}^{(m)}s_{rj}\omega_{kc}$.

2. *If* $\sum_{j,k}|\hat{\beta}_{jk}^{(m)} - \hat{\beta}_{jk}^{(m-1)}| < \epsilon\sum_{j,k}|\hat{\beta}_{jk}^{RIDGE}|$ *then stop, otherwise go to Step 1, where* ϵ *is a pre-specified small number, say* 10^{-4}.

Following from the convergence theory of Tseng (2001), this algorithm is guaranteed to converge to the global minimizer if the given precision matrix $\hat{\boldsymbol{\Omega}}$ is non-negative definite. In this case, the trace term in the objective function (15) is convex and differentiable and the penalty term decomposes into a sum of convex functions of individual parameters. As analyzed in Rothman et al. (2010), the computational complexity of this algorithm is $O(p^2q^2)$ per iteration, as it needs to cycle through pq parameters, and each calculation of u_{rc} costs $O(pq)$ flops.

For high dimensional problems, the iterative approach often performs less well. Firstly, the cyclical coordinate descent algorithm, which is a simple adoption of the Lasso algorithm for multivariate regression, can deteriorate its performance as the dimension increases. Secondly, the errors introduced in one step can adversely affect the performance of the approach in the other step. Also, it often takes a large number of iterations to converge as noted by Rothman et al. (2010). To address these issues, Rothman et al. (2010) proposed the so-called approximate MRCE (AMRCE) algorithm, which is described as follows.

Algorithm 4.3 Approximate MRCE (AMRCE) Algorithm

1. *Perform q separate lasso regressions each with the same optimal tuning parameter* $\hat{\lambda}_0$ *via cross validation. Let* $\hat{\boldsymbol{B}}^{lasso}_{\hat{\lambda}_0}$ *denote the solution.*

2. *Compute* $\hat{\boldsymbol{\Omega}} = \hat{\boldsymbol{\Omega}}(\hat{\boldsymbol{B}}^{lasso}_{\hat{\lambda}_0})$ *through the gLasso algorithm.*

3. *Compute* $\hat{\boldsymbol{B}} = \hat{\boldsymbol{B}}(\hat{\boldsymbol{\Omega}})$ *through the cyclical coordinate descent algorithm.*

The AMRCE algorithm basically output the result from the first sweep of the iterative approach, except that the initial value for the parameter matrix $\hat{\boldsymbol{B}}$ is chosen by implementing q separate Lasso regression each with the same optimal tuning parameter $\hat{\lambda}_0$ selected via cross validation. In this paper, we implemented step 2 using the R-package *gLasso* (Friedman et al., 2015), and implemented step 3 using the R-package *MRCE* (Rothman, 2015). The two-stage algorithm proposed by Sofer et al. (2014) is essentially the same with the AMRCE algorithm, where one stage is to estimate $\boldsymbol{B}$ conditional on the estimate of $\boldsymbol{\Omega}$, and the other stage is to estimate $\boldsymbol{\Omega}$ conditional on the estimate of $\boldsymbol{B}$. The major difference between the two algorithms is on how to set the initial estimators and how to tune the regularization parameters.

4.2 The BwC Approach

Consider the following transformation for the model (13) for a given precision matrix $\boldsymbol{\Omega}$:

$$\tilde{\boldsymbol{Y}} = \boldsymbol{X}\tilde{\boldsymbol{B}} + \tilde{\boldsymbol{E}}, \tag{17}$$

where $\tilde{\boldsymbol{Y}} = \boldsymbol{Y}\boldsymbol{\Omega}^{\frac{1}{2}}$, $\tilde{\boldsymbol{B}} = \boldsymbol{B}\boldsymbol{\Omega}^{\frac{1}{2}}$ and $\tilde{\boldsymbol{E}} = \boldsymbol{E}\boldsymbol{\Omega}^{\frac{1}{2}}$. For this transformed model, the elements of $\tilde{\boldsymbol{E}}$ are i.i.d. Gaussian $N(0,1)$. If we partition the matrices $\tilde{\boldsymbol{Y}}$, $\tilde{\boldsymbol{B}}$ and $\tilde{\boldsymbol{E}}$ columnwisely, i.e., letting $\tilde{\boldsymbol{Y}} = (\tilde{\boldsymbol{y}}_1, \tilde{\boldsymbol{y}}_2, \cdots, \tilde{\boldsymbol{y}}_q)$, $\tilde{\boldsymbol{B}} = (\tilde{\boldsymbol{\beta}}_1, \cdots, \tilde{\boldsymbol{\beta}}_q)$, and $\tilde{\boldsymbol{E}} = (\tilde{\boldsymbol{\epsilon}}_1, \tilde{\boldsymbol{\epsilon}}_2, \cdots, \tilde{\boldsymbol{\epsilon}}_q)$, then the model (17) can be expressed as q independent regressions:

$$\tilde{\boldsymbol{y}}_i = \boldsymbol{X}\tilde{\boldsymbol{\beta}}_i + \tilde{\boldsymbol{\epsilon}}_i, \qquad i = 1, \cdots, q, \tag{18}$$

where $\tilde{\boldsymbol{\epsilon}}_i$ follows a multivariate Gaussian distribution $N_n(\boldsymbol{0}, I)$. Based on this transformation, the BwC approach can be applied to iteratively estimate the coefficient matrix $\tilde{\boldsymbol{B}}$ and the precision matrix $\boldsymbol{\Omega}$. One sweep of the approach consists of the following steps:

Algorithm 4.4 BwC for Transformed Multivariate Regression

1. *Conditioned on the current estimator of* $\boldsymbol{\Omega}$*, estimate* $\tilde{\boldsymbol{\beta}}_1, \ldots, \tilde{\boldsymbol{\beta}}_q$ *independently according to (18) using the BwC algorithm 3.1.*

2. *Conditioned on the current estimator of* $\tilde{\boldsymbol{B}}$*, estimate the adjacency structure of the precision matrix* $\boldsymbol{\Omega}$ *from the residual* $\boldsymbol{Y} - \boldsymbol{X}\hat{\tilde{\boldsymbol{B}}}\Sigma^{1/2}$ *using the* ψ*-learning algorithm (Liang et al., 2015), where* $\hat{\tilde{\boldsymbol{B}}}$ *denote the current estimate of* $\tilde{\boldsymbol{B}}$*.*

3. *Based on the adjacency structure obtained in step 2, recover the precision matrix* $\boldsymbol{\Omega}$ *using the algorithm given in Hastie et al. (2009)(p.634).*

Note that Algorithm 4.4 is a nested BwC algorithm, where $\tilde{\boldsymbol{B}}$ is also estimated using the BwC algorithm. Given an estimate of $\tilde{\boldsymbol{B}}$, one may get an estimate of $\boldsymbol{B}$ via the transformation $\hat{\boldsymbol{B}} = \hat{\tilde{\boldsymbol{B}}}\hat{\boldsymbol{\Sigma}}^{\frac{1}{2}}$. Note that the estimate $\hat{\boldsymbol{B}}$ obtained in this way might not be very sparse (although still sparse in rows), as $\hat{\Sigma}^{\frac{1}{2}}$ can be a dense matrix. However, since a zero-row of $\tilde{\boldsymbol{B}}$ implies a zero-row of $\boldsymbol{B}$, $\hat{\tilde{\boldsymbol{B}}}$ can be used for variable screening, i.e., removing those variables in $\boldsymbol{X}$ which correspond a zero-row of $\hat{\tilde{\boldsymbol{B}}}$. After variable screening, the remaining rows of $\boldsymbol{B}$ can be easily estimated using Algorithm 4.2 based on the estimate of $\boldsymbol{\Omega}$. In summary, the proposed approach consists of the following steps:

Algorithm 4.5 BwC for Multivariate Regression

1. *(Initialization) Use the* ψ*-learning algorithm to estimate the adjacency structure of* $\boldsymbol{\Omega}$ *from the response variables* $\boldsymbol{Y}$*, and recover the working precision matrix* $\hat{\boldsymbol{\Omega}}$ *from this structure using the algorithm given in Hastie et al. (2009)(p.634).*

2. *(*$\tilde{\boldsymbol{B}}$ *and* $\boldsymbol{\Omega}$*-estimation) Conditioned on the working precision matrix* $\hat{\boldsymbol{\Omega}}$*, estimate* $\tilde{\boldsymbol{B}}$ *and* $\boldsymbol{\Omega}$ *using Algorithm 4.4.*

3. *(Variable screening) Remove all the variables in* $\boldsymbol{X}$ *which correspond to the zero-rows of* $\hat{\tilde{\boldsymbol{B}}}$*. Denote the reduced design matrix as* $\boldsymbol{X}_{red}$*.*

4. *(*$\boldsymbol{B}$*-estimation) Conditioned on the working precision matrix* $\hat{\boldsymbol{\Omega}}$*, estimate the remaining remaining rows of* $\boldsymbol{B}$ *using Algorithm 4.2 from the reduced model* $\boldsymbol{Y} = \boldsymbol{X}_{red}\boldsymbol{B}_{red} + \boldsymbol{E}$*.*

Regarding this algorithm, we have a few remarks.

1. Like the AMRCE algorithm, we find that Algorithm 4.4 used in estimation of $\tilde{\boldsymbol{B}}$ and $\boldsymbol{\Omega}$ only needs to iterate for a very few sweeps, say, 1 or 2. A large number of sweeps will not hurt the performance of the algorithm, but not necessary. In general, the algorithm can produce very stable estimates (with most of the significantly non-zero elements in $\tilde{\boldsymbol{B}}$ and $\boldsymbol{\Omega}$ being identified) within a very few sweeps, but will not converge to some fixed solution. This is because the execution of Algorithm 4.4 involves a lot of randomness, such as redistribution of selected variables, shuffling of unselected variables, and stochastic approximation estimation involved in the ψ-learning algorithm in recovering the adjacency structure of the precision matrix.

2. Due to the sparsity of $\tilde{\boldsymbol{B}}$, $\boldsymbol{X}$ can be much reduced in the step of variable screening. Since the initial estimate $\hat{\boldsymbol{B}}_{red}^{(0)} = \hat{\tilde{\boldsymbol{B}}}_{red}\hat{\Sigma}^{1/2}$ can be quite close to the true value of $\hat{\boldsymbol{B}}_{red}$, the $\boldsymbol{B}$-estimation step can usually converge very fast.

3. The BwC approach estimates $\tilde{\boldsymbol{\beta}}_i$, $i = 1, \ldots, q$, independently. For each i, the data size it deals with is $n \times p$, and the computational complexity of the BwC subroutine is $O(k'np)$, where k' denotes the number of sweeps performed in the subroutine. Recall that for each conditioned step of the subroutine, the SIS-MCP algorithm is implemented for variable selection, whose computational complexity is $O(nm)$ and m denotes the number of variables included in the subset data. Hence, the total computational complexity for q regressions is $O(k'nqp)$, which can be much better than $O(p^2q^2)$, the computational complexity of the MRCE algorithm.

4. The BwC approach adopted the ψ-learning algorithm (Liang et al., 2015) for estimating the adjacency structure of the precision matrix. The ψ-learning algorithm works based on a new measure of partial correlation coefficients, the so-called ψ-partial correlation coefficient, which is evaluated with a reduced conditional set and thus feasible for high-dimensional problems. Under the Markov property and adjacency faithfulness conditions, it can be shown that the ψ-partial correlation coefficient is equivalent to the true partial correlation coefficient in construction of Gaussian graphical networks, and the networks constructed based on which are

asymptotically consistent with the true network. The numerical results reported in Liang et al. (2015) indicate that the ψ-learning algorithm can produce more accurate networks than gLasso. In terms of computational complexity, the ψ-learning algorithm is also better than gLasso. The ψ-learning algorithm has a computational complexity of nearly $O(q^2)$, while gLasso has a computational complexity of $O(q^3)$. Even in its fast implementation (Witten et al., 2011; Mazumder and Hastie, 2012), which makes use of the block diagonal structure in the graphical Lasso solution, gLasso still has a computational complexity of $O(q^{2+\nu})$, where $0 < \nu \leq 1$ may vary with the block diagonal structure it used.

Finally, we would point out that a fundamental difference between the MRCE approach and the BwC approach is that MRCE works with an explicit objective function as specified in (14), while BwC does not. BwC requires only an asymptotic objective function for each block; that is, the estimator obtained for each block must be consistent, while the joint finite-sample objective function for all blocks might not exist in an explicit form. For the ψ-learning algorithm, which is developed based on the theory of graphical models (under neither regularization nor Bayesian frameworks), an explicit finite-sample objective function does not exist though a consistent estimator can still be produced. From this perspective, we can see that the BwC approach is flexible. It provides a general framework for combining different methods for parameter estimation of complex models.

4.3 A Simulated Example

Consider the multivariate regression (13) with $n = 200$, $q = 100$, and $p = 3000$, 5000 and 10000. The explanatory variables are generated by (10) and (11) except that the shared error term $\boldsymbol{e}$ is divided by a random number generated from Uniform[1,10]. That is, some of the explanatory variables are still highly correlated. The full regression coefficient matrix $\boldsymbol{B}$ is a $p \times q$ matrix, which contains 300,000, 500,000 and 1,000,000 elements for $p = 3000$, 5000 and 10000, respectively. To set the matrix $\boldsymbol{B}$, we randomly selected 200 elements to set them to 1 and set the non-selected

elements to 0. The precision matrix $\mathbf{\Omega} = (\omega_{ij})$ is given by

$$\omega_{ij} = \begin{cases} 1, & \text{if } i = j, \\ 0.5, & \text{if } |i-j| = 1, \\ 0.25, & \text{if } |i-j| = 2, \\ 0, & \text{otherwise.} \end{cases} \tag{19}$$

Such a precision matrix has been used by Yuan and Lin (2007), Mazumder and Hastie (2012) and Liang et al. (2015) to illustrate their respective algorithms for high dimensional Gaussian graphical models. For each value of p, we generated 10 independent datasets, and the numerical results reported below are averaged over the 10 datasets.

The BwC approach was first applied to this example. In the $\tilde{B}$- and $\mathbf{\Omega}$-estimation step, we set the initial block number $K = 10$, the number of sweeps $m = 25$, and the block diminishing size $s = 2$, and adopted the SIS-MCP algorithm for variable selection with the regularization parameter determined by the BIC criterion. The ψ-learning algorithm used in precision matrix estimation contains two free parameters, α_1 and α_2, which represent the significant levels of the screening tests for the correlation coefficients and ψ-partial correlation coefficients, respectively. The correlation screening step produces conditional sets for evaluating the ψ-partial correlation coefficients. As mentioned in Liang et al. (2015), the ψ-learning algorithm is pretty robust to the choice of α_1, and a large α_1 is usually preferred, which produces large conditional sets and thus reduces the risk of missing important connections in the resulting Gaussian graphical network. Here a connection also refers to a nonzero off-diagonal element of the precision matrix. For this example, we set $\alpha_1 = 0.25$. For α_2, it is ideal to decrease its value with sweeps. A large value of α_2 in the early sweeps helps to keep important connections in the network, as variable selection in these sweeps might still be pre-mature. A small value of α_2 in the late sweeps helps to reduce the false selection of network connections. For simplicity, we set $\alpha_2 = 0.05$ in all sweeps. In the variable screening step, the number of variables in $\boldsymbol{X}$ can be largely reduced. For the datasets with $p = 3000$, on average $\boldsymbol{X}_{red}$ contains only 201.6 variables with a standard deviation of 1.9; for $p = 5000$, on average $\boldsymbol{X}_{red}$ contains only 210.1 variables with a standard deviation of 2.3; and for $p = 10000$, on average $\boldsymbol{X}_{red}$

contains only 218.5 variables with a standard deviation of 2.8. After obtaining the estimate of $\boldsymbol{\Omega}$, the cyclical coordinate descent algorithm was applied to estimate $\boldsymbol{B}_{red}$ based on the reduced set of variables. For this step, the regularization parameter was tuned according to the EBIC criterion. For comparison, the AMRCE algorithm was also applied to this example, for which the regularization parameter was tuned according to the BIC criterion.

Table 6 and Table 7 summarize the results for precision matrix estimation and variable selection, respectively. The comparison shows that BwC significantly outperforms AMRCE for this example. As indicated by Table 6, AMRCE mis-identified almost all true connections, while BwC can correctly identified over 80% of them. For variable selection, BwC tends to have a lower false selection rate than AMRCE, although they both are able to identify all true variables. It is remarkable that in terms of mean squared errors (MSE) of the estimate of $\boldsymbol{B}$, BwC works much better than AMRCE. To have this issue further explored, we plot in Figure 3 the histograms of the non-zero elements of $\hat{\boldsymbol{B}}$ obtained for three datasets. The histograms show that for the true variables, the regression coefficient estimates produced by BwC are closer to their true value 1 than those produced by AMRCE. Also, the true variables and the falsely selected variables are more separable for the BwC estimates than for the AMRCE estimates. In fact, with a finely tuned threshold, the true variables can be accurately identified with the BwC approach.

Finally, we would like to report that the BwC approach took a little longer CPU time than AMRCE for this example. This is because BwC took multiple sweeps with diminishing block sizes in estimation of $\tilde{\boldsymbol{B}}$, although it is cheaper than AMRCE for each sweep. On average, BwC took 26, 36 and 52 CPU minutes on an Intel Core i7-4790@3.60GHz Quad-Core desktop for each dataset with $p = 3000$, 5000 and 10000, respectively. On average, AMRCE took 5, 9 and 20 CPU minutes for these datasets, respectively, on the same computer.

4.4 A Real Data Study: eQTL Analysis

This dataset includes 60 unrelated individuals of Northern and Western European ancestry from Utah (CEU), whose genotypes are available from the International Hapmap project and publicly downloadable via the hapmart interface (http://hapmart.hapmap.org). The genotype is coded

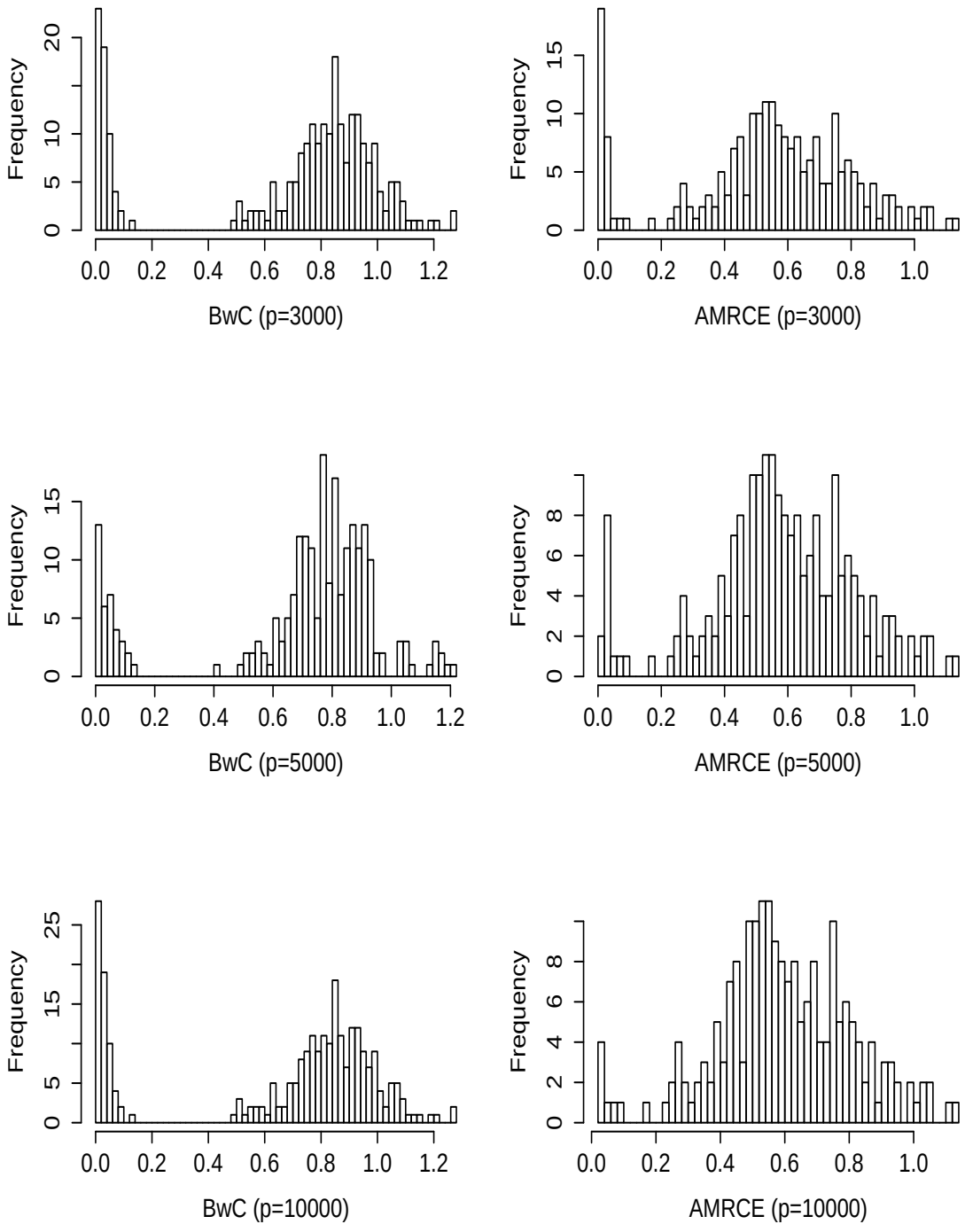

Figure 3: Histograms of non-zero elements of $\hat{\boldsymbol{B}}$ obtained by BwC (left panel) and AMRCE (right panel) for three datasets with $p = 3000$ (upper panel), $p = 5000$ (middle panel), and $p = 10000$, respectively.

Table 6: Results comparison for precision matrix estimation: the results are averages over 10 independent datasets with the standard deviation given in the parenthesis, where $|\hat{\boldsymbol{s}}|_{avg}$ denotes the number of connections selected by the method, and $|\hat{\boldsymbol{s}} \cap \boldsymbol{s}_*|_{avg}$ denotes the number of true connections selected by the method. The true number of connections is 197.

p	method	$\lvert\hat{\boldsymbol{s}}\rvert_{avg}$	$\lvert\hat{\boldsymbol{s}} \cap \boldsymbol{s}_*\rvert_{avg}$	fsr	nsr
3000	BwC	171.1 (2.28)	162.0 (1.75)	0.05 (0.008)	0.18 (0.009)
	AMRCE	70.7 (5.92)	1.1 (0.33)	0.99 (0.004)	0.99 (0.003)
5000	BwC	169.8 (3.51)	161.3 (2.64)	0.05 (0.006)	0.18 (0.013)
	AMRCE	68.35 (10.14)	0.65 (0.20)	0.99 (0.003)	0.99 (0.002)
10000	BwC	165.6 (1.57)	156.2 (1.54)	0.06 (0.006)	0.21 (0.008)
	AMRCE	65.35 (13.58)	1.55 (0.52)	0.98 (0.006)	0.98 (0.005)

as 0, 1 and 2 for homozygous rare, heterozygous and homozygous common alleles, respectively. Our study focuses on the SNPs found in the $5'$ UTR (untranslated region) of mRNA (messenger RNA) with a minor allele frequency of 0.1 or more. The UTR possibly has an important role in the regulation of gene expression and has previously been subject to investigation by Chen et al. (2008). The gene expression data of these individuals were analyzed by Stranger et al. (2007). There were four replicates for each individual. The raw data were background corrected and then quantile normalized across replicates of a single individual and then median normalized across all individuals. The gene expression data are again publicly available from the Sanger Institute website (ftp://ftp.sanger.ac.uk/pub/genevar). Out of the 47293 total available probes corresponding to different Illumina TargetID, we selected the 100 most variable probes with each corresponding to a different transcript. Further, we removed the "duplicated" SNPs. Two SNPs are said "duplicated" if they have the same genotypes across all individuals. After these reductions, we reached a dataset with $n = 60$, $p = 3005$ (for SNPs), and $q = 100$ (for transcripts). The same dataset has been analyzed by Bhadra and Mallick (2013).

To validate the Gaussian assumption for the response variables of the model (13), the *non-paranormal* transformation proposed by Liu et al. (2009) has been applied to the gene expression

Table 7: Results comparison for variable selection: the results are averages over 10 independent datasets with the standard deviation given in the parenthesis, where $|\hat{\boldsymbol{s}}|_{avg}$ denotes the number of variables selected by the method, and $|\hat{\boldsymbol{s}} \cap \boldsymbol{s}_*|_{avg}$ denotes the number of true variables selected by the method. The true number of variables is 200.

p	method	$\lvert\hat{\boldsymbol{s}}\rvert_{avg}$	$\lvert\hat{\boldsymbol{s}} \cap \boldsymbol{s}_*\rvert_{avg}$	fsr	nsr	MSE
3000	BwC	248.3 (3.23)	200 (0)	0.19 (0.010)	0 (0)	0.05 (0.003)
	AMRCE	268.5 (5.03)	199.9 (0.10)	0.25 (0.014)	0.001 (0.001)	0.22 (0.009)
5000	BwC	251.8 (3.91)	200 (0)	0.20 (0.012)	0 (0)	0.05 (0.002)
	AMRCE	265.9 (12.96)	199.4 (0.40)	0.23 (0.039)	0.003 (0.002)	0.25 (0.019)
10000	BwC	255.9 (3.96)	200 (0)	0.22 (0.012)	0 (0)	0.06 (0.003)
	AMRCE	233.3 (7.20)	198.5 (0.50)	0.14 (0.021)	0.008 (0.003)	0.30 (0.011)

data. The *nonparanormal* transformation is a semiparametric transformation, which converts a continuous variable to a Gaussian variable while leaving the structure of the underlying graphical model unchanged. Prior to the analysis, we centered both the response variable (Gene expression data) and the predictors (SNP). Each of the predictors has also been normalized to have a standard deviation of 1. The BwC approach was applied to this example. In the step of $\tilde{\boldsymbol{B}}$ and $\boldsymbol{\Omega}$-estimation, we set the initial block number $K = 10$, the number of sweeps $m = 25$, and the block diminishing size $s = 2$ for the sub-BwC run. For the ψ-learning algorithm, we set $\alpha_1 = 0.25$ and $\alpha_2 = 0.05$. The BwC approach was independently run for 25 times with the results summarized in Table 8 and Table 9.

Table 8 reports 8 pairs of associated transcripts that were identified by BwC in all 25 runs. Among them, there are two transcripts from the gene FXYD2, two transcripts from the gene KIAA0125, and two transcripts from the gene PIP3-E. Also, the associated gene pair, CYorf15B and SMCY, found in this study are known to be related with gender (Weickert, 2009). These results indicate the validity of the BwC approach. Due to the general weak relations between SNPs and transcripts, the null model tends to be produced in the $\boldsymbol{B}$-estimation step with the cyclical coordinate descent algorithm. Instead, we explored the estimator $\hat{\boldsymbol{B}} = \hat{\tilde{\boldsymbol{B}}}\hat{\boldsymbol{\Sigma}}^{\frac{1}{2}}$ for this example.

Table 8: Pairs of associated transcripts identified by BwC in all 25 runs.

Correlated Transcripts	Related Gene
GI_14211892-S : GI_33356559-S	CYorf15B : SMCY
GI_20302136-S : GI_7661757-S	KIAA0125 : KIAA0125
hmm3574-S : Hs.406489-S	hmm3574 : Hs.406489
Hs.449602-S : Hs.449584-S	Hs.449602 : Hs.449584
Hs.512137-S : GI_41190507-S	Hs.512137 : LOC388978
Hs.449602-S : Hs.449584-S	Hs.449602 : Hs.449584
GI_27754767-A : GI_27754767-I	FXYD2 : FXYD2
GI_40354211-S : Hs.185140-S	PIP3-E : PIP3-E

For this estimator, there were 256 non-zero elements identified by BwC in all 25 runs. Among the top 100 elements (in magnitudes), we found 5 SNPs which are associated with more than one transcripts. Table 9 shows these SNPs and the associated multiple transcripts.

Table 9: SNPs and their associated multiple transcripts identified by BwC.

SNP	Transcript
rs3784932	GI_24308084-S, GI_21389558-S
rs1044369	GI_37546969-S, hmm3587S
rs1944926	GI_21464138-A , GI_22027487-S, GI_41350202-S, hmm9615-S
rs2241649	hmm9615S, GI_7019408-S
rs12892419	GI_33356162-S, GI_13514808-S

5 Conclusion

We have proposed the *blockwise consistency* (BwC) method as a potential solution to the problem of parameter estimation for complex models that are often encountered in big data analysis. The BwC method decomposes the high dimensional parameter estimation problem to a series of lower

dimensional parameter estimation problems which often have much simpler structures than the original high-dimensional problem, and thus can be easily solved. Moreover, under the framework provided by BwC, a variety of methods, such as Bayesian and frequentist methods, can be jointly used to achieve a consistent estimator for the original high-dimensional complex model. The BwC approach has been illustrated using two examples, high dimensional variable selection and high-dimensional multivariate regression. Both examples show that the BwC method can provide a drastic improvement over the existing methods. Extension of the BwC method to other high-dimensional problems, such as variable selection for high-dimensional generalized linear models, is straightforward.

The BwC method works in a similar way to the block coordinate descent algorithm. As mentioned previously, the fundamental difference between the two algorithms is that the block coordinate descent algorithm works with an explicit finite-sample objective function for each block, while BwC does not. BwC requires only a consistent estimator for each block, while the joint finite-sample objective function for all blocks might not exist in an explicit form. This allows us to combine the best methods to each sub-problem to solve the original complex problems.

The BwC method can also be used under the Bayesian framework to find the maximum *a posteriori* (MAP) estimator for a complex model, e.g., hierarchical model, for which the conjuacy holds for the parameters. We note that these models are traditionally treated with the Gibbs sampler under the Bayesian framework. However, the feasibility of the Gibbs sampler (Geman and Geman, 1984) is being challenged in the era of big data, as which typically requires a large number of iterations and multiple scans of the full dataset for each sweep. Compared to the Gibbs sampler, BwC can converge much faster, usually within tens of iterations. For many problems, typically, if the Gibbs sampler can be implemented, so can BwC but with substantially less work and more straightforward convergence properties. The uncertainty of the BwC estimator can be assessed using the bootstrap method (Efron and Tibshirani, 1993).

References

Barretina, J., Caponigro, G., Stransky, N., Venkatesan, K., Margolin, A., Kim, S., Wilson, C., Lehar, J., Kryukov, G., Sonkin, D., Reddy, A., Liu, M., Murray, L., Berger, M., Monahan, J., Morais, P., Meltzer, J., Korejwa, A., Jane-Valbuena, J. andMapa, F., Thibault, J., Bric-Furlong, E., Raman, P., Shipway, A., Engels, I., and et al. (2012), "The Cancer cell line encyclopedia enables predictive modeling of anticancer drug sensitivity," *Nature*, 483, 603–607.

Bhadra, A. and Mallick, B. (2013), "Joint High-Dimensional Bayesian Variable amd Covariance Selection with an Application to eQTL Analysis," *Biometrics*, 69, 447–457.

Breheny, P. and Huang, J. (2011), "Coordinate descent algorithms for nonconvex penalized regression, with applications to biological feature selection," *Annals of Applied Statistics*, 5, 232–252.

Chen, J. and Chen, Z. (2008), "Extended Bayesian information criterion for model selection with large model space," *Biometrika*, 94, 759–771.

Duguet, M. (1997), "When helicase and topoisomerase meet!" *J. Cell Sci.*, 110, 1345–1350.

Efron, B. and Tibshirani, R. (1993), *An Introduction to the Bootstrap*, Boca Raton, FL: Chapman & Hall/CRC.

Fan, J., Feng, Y., Saldana, D. F., Samworth, R., and Wu, Y. (2015), "Sure Independence Screening," *CRAN R Package*.

Fan, J. and Li, R. (2001), "Variable selection via nonconcave penalized likelihood and its oracle properties," *Journal of the American Statistical Association*, 96, 1348–1360.

Fan, J., Samworth, R., and Wu, Y. (2009), "Ultrahigh dimensional feature selection: Beyond the linear model," *Journal of Machine Learning Research*, 10, 1829–1853.

Friedman, J., Hastie, T., and Tibshirani, R. (2008), "Sparse inverse covariance estimation with the graphical lasso," *Biostatistics*, 9, 432–441.

— (2010), "Regularization paths for generalized linear models via coordinate descent," *Journal of Statistical Software*, 33, 1–22.

— (2015), "GLASSO: Graphical lasso- estimation of Gaussian graphical models," *CRAN R-Package*.

Geman, S. and Geman, D. (1984), "Stochastic relaxation, Gibbs distributions and the Bayesian restoration of images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6, 721–741.

Guo, N., Wan, Y., Tosun, K., Lin, H., Msiska, Z., Flynn, D., Remick, S., Vallyathan, V., Dowlati, A., Shi, X., Castranova, V., Beer, D., and Qian, Y. (2008), "Confirmation of gene expression-based prediction of survival in non-small cell lung cancer," *Clin. Cancer Res.*, 14, 8213–8220.

Hamburg, M. and Collins, F. (2010), "The path to personalized medicine," *New Engl. J. Med.*, 363, 301–304.

Hastie, T., Tibshirani, R., and Friedman, J. (2009), *The Elements of Statistical Learning*, Springer.

Li, X., Xu, S., Cheng, Y., and Shu, J. (2016), "HSPB1 polymorphisms might be associated with radiation-induced damage risk in lung cancer patients treated with radiotherapy," *Tumour Biol.*, to appear.

Liang, F., Song, Q., and Qiu, P. (2015), "An Equivalent Measure of Partial Correlation Coefficients for High Dimensional Gaussian Graphical Models," *Journal of the American Statistical Association*, 110, 1248–1265.

Liu, H., Lafferty, J., and Wasserman, L. (2009), "The Nonparanormal: Semiparametric Estimation of High Dimensional Undirected Graphs," *Journal of Machine Learning Research*, 10, 2295–2328.

Mazumder, R., Friedman, J., and Hastie, T. (2011), "SparseNet: Coordinate descent with nonconvex penalties," *Journal of the American Statistical Association*, 106, 1125–1138.

Mazumder, R. and Hastie, T. (2012), "The graphical Lasso: New insights and alternatives," *Electronic Journal of Statistics*, 6, 2125–2149.

Meinshausen, N. and Bühlmann, P. (2006), "High-dimensional graphs and variable selection with the Lasso," *Annals of Statistics*, 34, 1436–1462.

Peng, J., Zhu, J., Bergamaschi, A., Han, W., Noh, D.-Y., Pollack, J. R., and Wang, P. (2010), "Regularized Multivariate Regression for Identifying Master Predictors with Application to Integrative Genomics Study of Breast Cancer," *Annals of Applied Statistics*, 4, 53–77.

Rothman, A. (2015), "MRCE: Multivariate regression with covariance estimation," *CRAN R-Package*.

Rothman, A., Levina, E., and Zhu, J. (2010), "Sparse multivariate regression with covariance estimation," *Journal of Computational and Graphical Statistics*, 19, 947–962.

Sofer, T., Dicker, L., and Lin, X. (2014), "Variable selection for high dimensional multivariate outcomes," *Statistica Sinica*, 22, 1633–1654.

Song, Q. and Liang, F. (2015a), "High Dimensional Variable Selection with Reciprocal L1-Regularization," *Journal of the American Statistical Association*, 110, 1607–1620.

— (2015b), "A Split-and-Merge Bayesian Variable Selection Approach for Ultra-high dimensional Regression," *Journal of the Royal Statistical Society, Series B*, 77, 947–972.

Tibshirani, R. (1996), "Regression shrinkage and selection via the LASSO," *Journal of the Royal Statistical Society, Series B*, 58, 267–288.

Tseng, P. (2001), "Convergence of a block coordinate descent method for nondifferentiable minimization," *Journal of Optimization Theory and Applications*, 109, 475–494.

Tseng, P. and Yun, S. (2009), "A coordinate gradient descent method for nonsmooth separable minimization," *Mathematical Programming, Series B*, 117, 387–423.

Turlach, B., Venables, W., and Wright, S. (2005), "Simultaneous variable selection," *Technometrics*, 47, 349–363.

Weickert, C. e. a. (2009), "Transcriptome analysis of male female differences in prefrontal cortical development," *iMolecular Psychiatry*, 14, 558–561.

Witten, D., Friedman, J., and Simon, N. (2011), "New insights and faster computations for the graphical Lasso," *Journal of Computational and Graphical Statistics*, 20, 892–900.

Yuan, M. and Lin, Y. (2007), "Model selection and estimation in the Gaussian graphical model," *Biometrika*, 95, 19–35.

Zhang, C.-H. (2010), "Nearly unbiased variable selection under minimax concave penalty," *Annals of Statistics*, 38, 894–942.

Zhao, P. and Yu, B. (2006), "On model selection consistency of Lasso," *Journal of Machine Learning Research*, 7, 2541–2563.

Zou, H. (2006), "The adptive lasso and its oracle properties," *Annals of Statistics*, 38, 894–942.