Stat Comput (2014) 24:521–529
DOI 10.1007/s11222-013-9385-5

Coordinate descent algorithm for covariance graphical lasso

Hao Wang

Received: 11 July 2012 / Accepted: 12 February 2013 / Published online: 23 February 2013

Abstract Bien and Tibshirani (Biometrika, 98(4):807–820, 2011) have proposed a covariance graphical lasso method that applies a lasso penalty on the elements of the covariance matrix. This method is definitely useful because it not only produces sparse and positive definite estimates of the covariance matrix but also discovers marginal independence structures by generating exact zeros in the estimated covariance matrix. However, the objective function is not convex, making the optimization challenging. Bien and Tibshirani (Biometrika, 98(4):807–820, 2011) described a majorize-minimize approach to optimize it. We develop a new optimization method based on coordinate descent. We discuss the convergence property of the algorithm. Through simulation experiments, we show that the new algorithm has a number of advantages over the majorize-minimize approach, including its simplicity, computing speed and numerical stability. Finally, we show that the cyclic version of the coordinate descent algorithm is more efficient than the greedy version.

Keywords Coordinate descent · Covariance graphical lasso · Covariance matrix estimation · L_1 penalty · MM algorithm · Marginal independence · Regularization · Shrinkage · Sparsity

1 Introduction

Bien and Tibshirani (2011) proposed a covariance graphical lasso procedure for simultaneously estimating covariance matrix and marginal dependence structures.[1] Let $\mathbf{S}$ be the sample covariance matrix such that $\mathbf{S} = \mathbf{Y}'\mathbf{Y}/n$ where $\mathbf{Y}(n \times p)$ is the data matrix of p variables and n samples. A basic version of their covariance graphical lasso problem is to minimize the following objective function:

$$g(\mathbf{\Sigma}) = \log(\det \mathbf{\Sigma}) + \operatorname{tr}\left(\mathbf{S}\mathbf{\Sigma}^{-1}\right) + \rho\|\mathbf{\Sigma}\|_1, \tag{1}$$

over the space of positive definite matrices M^+ with $\rho \geq 0$ being the shrinkage parameter. Here, $\mathbf{\Sigma} = (\sigma_{ij})$ is the $p \times p$ covariance matrix and $\|\mathbf{\Sigma}\|_1 = \sum_{1\leq i,j\leq p} |\sigma_{ij}|$ is the L_1-norm of $\mathbf{\Sigma}$. A general version of the covariance graphical lasso in Bien and Tibshirani (2011) allows different shrinkage parameters for different elements in $\mathbf{\Sigma}$. To ease exposition, we describe our methods in the context of one common shrinkage parameter as in (1). All of our results can be extended to the general version of different shrinkage parameters with little difficulty.

Because of the L_1-norm term, the covariance graphical lasso is able to set some of the off-diagonal elements of $\mathbf{\Sigma}$ exactly equal to zero in its minimum point of (1). Zeros in $\mathbf{\Sigma}$ encode marginal independence structures among the components of a multivariate normal random vector with covariance matrix $\mathbf{\Sigma}$. It is distinctly different from the concentration graphical models (also referred to as covariance selection models due to Dempster 1972) where zeros are in the concentration matrix $\mathbf{\Sigma}^{-1}$ and are associated with conditional independence.

The objective function (1) is not convex, imposing computational challenges for minimizing it. Bien and Tibshirani (2011) proposed a majorize-minimize approach to approxi-

H. Wang (✉)
Department of Statistics, University of South Carolina, Columbia, SC 29208, USA
e-mail: haowang@sc.edu

[1] An unpublished Ph.D. dissertation Lin (2010) may consider the covariance graphical lasso method earlier than Bien and Tibshirani (2011).

mately minimize (1). In this paper, we develop the coordinate descent algorithm for minimizing (1). We discuss the convergence property and investigate its computational efficiency through simulation studies. In comparison with Bien and Tibshirani (2011)'s algorithm, the coordinate descent algorithm is vastly simpler to implement, substantially faster to run and numerically more stable in our tests.

Coordinate descent is not new to the model fitting for regularized problems. It is shown to be very competitive for solving convex and some non-convex penalized regression models (Fu 1998; Sardy et al. 2000; Friedman et al. 2007; Wu and Lange 2008; Breheny and Huang 2011) as well as the concentration graphical models (Friedman et al. 2008). However, the development of this general algorithm to covariance graphical lasso models is new and unexplored before. In this sense, our work also contributes to the literature by documenting the usefulness of this important algorithm for the class of covariance graphical lasso models.

Finally, we investigate if the proposed coordinate descent can be further improved by comparing it with two additional competitors: a greedy coordinate descent algorithm and a new majorize-minimize algorithm. We show that the cyclic coordinate descent algorithm remains to be superior to these competitors in efficiency.

2 Coordinate descent algorithm

2.1 Algorithm description

To minimize (1) based on the simple idea of coordinate descent, we show how to update $\mathbf{\Sigma}$ one column and row at a time while holding all of the rest elements in $\mathbf{\Sigma}$ fixed. Without loss of generality, we focus on the last column and row. Partition $\mathbf{\Sigma}$ and $\mathbf{S}$ as follows:

$$\mathbf{\Sigma} = \begin{pmatrix} \mathbf{\Sigma}_{11}, \boldsymbol{\sigma}_{12} \\ \boldsymbol{\sigma}'_{12}, \sigma_{22} \end{pmatrix}, \qquad \mathbf{S} = \begin{pmatrix} \mathbf{S}_{11}, \mathbf{s}_{12} \\ \mathbf{s}'_{12}, s_{22} \end{pmatrix}, \tag{2}$$

where (a) $\mathbf{\Sigma}_{11}$ and $\mathbf{S}_{11}$ are the covariance matrix and the sample covariance matrix of the first $p-1$ variables, respectively; (b) $\boldsymbol{\sigma}_{12}$ and $\mathbf{s}_{12}$ are the covariances and the sample covariances between the first $p-1$ variables and the last variable, respectively; and (c) σ_{22} and $\mathbf{s}_{22}$ are the variance and the sample variance of the last variable, respectively.

Let

$$\boldsymbol{\beta} = \boldsymbol{\sigma}_{12}, \qquad \gamma = \sigma_{22} - \boldsymbol{\sigma}'_{12}\mathbf{\Sigma}_{11}^{-1}\boldsymbol{\sigma}_{12},$$

and apply the block matrix inversion to $\mathbf{\Sigma}$ using blocks $(\mathbf{\Sigma}_{11}, \boldsymbol{\beta}, \gamma)$:

$$\mathbf{\Sigma}^{-1} = \begin{pmatrix} \mathbf{\Sigma}_{11}^{-1} + \mathbf{\Sigma}_{11}^{-1}\boldsymbol{\beta}\boldsymbol{\beta}'\mathbf{\Sigma}_{11}^{-1}\gamma^{-1}, & -\mathbf{\Sigma}_{11}^{-1}\boldsymbol{\beta}\gamma^{-1} \\ -\boldsymbol{\beta}'\mathbf{\Sigma}_{11}^{-1}\gamma^{-1}, & \gamma^{-1} \end{pmatrix}. \tag{3}$$

The three terms in (1) can be expressed as a function of $(\boldsymbol{\beta}, \gamma)$:

$$\log(\det \mathbf{\Sigma}) = \log(\gamma) + c_1,$$

$$\operatorname{tr}\big(\mathbf{S}\mathbf{\Sigma}^{-1}\big) = \boldsymbol{\beta}'\mathbf{\Sigma}_{11}^{-1}\mathbf{S}_{11}\mathbf{\Sigma}_{11}^{-1}\boldsymbol{\beta}\gamma^{-1} - 2\mathbf{s}'_{12}\mathbf{\Sigma}_{11}^{-1}\boldsymbol{\beta}\gamma^{-1} + s_{22}\gamma^{-1} + c_2,$$

$$\rho\|\mathbf{\Sigma}\|_1 = 2\rho\|\boldsymbol{\beta}\|_1 + \rho\big(\boldsymbol{\beta}'\mathbf{\Sigma}_{11}^{-1}\boldsymbol{\beta} + \gamma\big) + c_3,$$

where c_1,c_2 and c_3 are constants not involving $(\boldsymbol{\beta}, \gamma)$. Dropping off c_1,c_2 and c_3 from (1), we have the following objective function with respect to $(\boldsymbol{\beta}, \gamma)$:

$$\min_{\boldsymbol{\beta},\gamma}\big\{\log(\gamma) + \boldsymbol{\beta}'\mathbf{\Sigma}_{11}^{-1}\mathbf{S}_{11}\mathbf{\Sigma}_{11}^{-1}\boldsymbol{\beta}\gamma^{-1} - 2\mathbf{s}'_{12}\mathbf{\Sigma}_{11}^{-1}\boldsymbol{\beta}\gamma^{-1} + s_{22}\gamma^{-1} + 2\rho\|\boldsymbol{\beta}\|_1 + \rho\boldsymbol{\beta}'\mathbf{\Sigma}_{11}^{-1}\boldsymbol{\beta} + \rho\gamma\big\}. \tag{4}$$

For γ, removing terms in (4) that do not depend on γ gives

$$\min_{\gamma}\big\{\log(\gamma) + a\gamma^{-1} + \rho\gamma\big\},$$

where $a = \boldsymbol{\beta}'\mathbf{\Sigma}_{11}^{-1}\mathbf{S}_{11}\mathbf{\Sigma}_{11}^{-1}\boldsymbol{\beta} - 2\mathbf{s}_{12}\mathbf{\Sigma}_{11}^{-1}\boldsymbol{\beta} + s_{22}$. Clearly, it is solved by:

$$\hat{\gamma} = \begin{cases} a & \text{if } \rho = 0, \\ (-1 + \sqrt{1 + 4a\rho})/(2\rho) & \text{if } \rho \neq 0. \end{cases} \tag{5}$$

For $\boldsymbol{\beta}$, removing terms in (4) that do not depend on $\boldsymbol{\beta}$ gives

$$\min_{\boldsymbol{\beta}}\big\{\boldsymbol{\beta}'\mathbf{V}\boldsymbol{\beta} - 2\mathbf{u}'\boldsymbol{\beta} + 2\rho\|\boldsymbol{\beta}\|_1\big\}, \tag{6}$$

where $\mathbf{V} = (v_{ij}) = \mathbf{\Sigma}_{11}^{-1}\mathbf{S}_{11}\mathbf{\Sigma}_{11}^{-1}\gamma^{-1} + \rho\mathbf{\Sigma}_{11}^{-1}$, $\mathbf{u} = \mathbf{\Sigma}_{11}^{-1}\mathbf{s}_{12}\gamma^{-1}$. The problem in (6) is a lasso problem and can be efficiently solved by coordinate descent algorithms (Friedman et al. 2007; Wu and Lange 2008). Specifically, for $j \in \{1, \ldots, p-1\}$, the minimum point of (6) along the coordinate direction in which β_j varies is:

$$\hat{\beta}_j = \mathcal{S}\Big(u_j - \sum_{k \neq j} v_{kj}\hat{\beta}_k, \rho\Big)/v_{jj}, \tag{7}$$

where $\mathcal{S}$ is the soft-threshold operator:

$$\mathcal{S}(x, t) = \operatorname{sign}(x)\big(|x| - t\big)_+.$$

The update (7) is iterated for $j = 1, \ldots, p-1, 1, 2, \ldots,$ until convergence. We then update the column as $(\boldsymbol{\sigma}_{12} = \boldsymbol{\beta}, \sigma_{22} = \gamma + \boldsymbol{\beta}'\mathbf{\Sigma}_{11}^{-1}\boldsymbol{\beta})$ followed by cycling through all columns until convergence. This algorithm can been viewed as a block coordinate descent method with p blocks of $\boldsymbol{\beta}$'s and another p blocks of γ's. The algorithm is summarized as follows:

Coordinate descent algorithm Given input $(\mathbf{S}, \rho)$, start with $\boldsymbol{\Sigma}^{(0)}$, and at the $(k+1)$th iteration $(k = 0, 1, \ldots)$

1. Let $\boldsymbol{\Sigma}^{(k+1)} = \boldsymbol{\Sigma}^{(k)}$.
2. For $i = 1, \ldots, p$,
   (a) Partition $\boldsymbol{\Sigma}^{(k+1)}$ and $\mathbf{S}$ as in (2).
   (b) Compute γ as in (5).
   (c) Solve the lasso problem (6) by repeating (7) until convergence.
   (d) Update $\boldsymbol{\sigma}_{12}^{(k+1)} = \boldsymbol{\beta}, \boldsymbol{\sigma}_{21}^{(k+1)} = \boldsymbol{\beta}'$, $\sigma_{22}^{(k+1)} = \gamma + \boldsymbol{\beta}' \boldsymbol{\Sigma}_{11}^{-1} \boldsymbol{\beta}$.
3. Let $k = k + 1$ and repeat (1)–(3) until convergence.

2.2 Algorithm convergence

The convergence of the proposed block coordinate descent algorithm to a stationary point can be addressed by the theoretical results for block coordinate descent methods for non-differentiable minimization by Tseng (2001). The key to applying the general theory there to our algorithm is the separability of the non-differentiable penalty terms in (1). First, from (5) and (6), the objective function g has a unique minimum point in each coordinate block. This satisfies the conditions of Part (c) of Theorem 4.1 in Tseng (2001) and hence implies that the algorithm converges to a coordinatewise minimum point. Second, because all directional derivatives exist, by Lemma 3.1 of Tseng (2001), each coordinatewise minimum point is a stationary point. A similar argument has been given by Breheny and Huang (2011) to show the convergence of coordinate decent algorithm to a stationary point for nonconvex penalized regression models.

3 Comparison of algorithms

We conduct a simulation experiment to compare the performance of the proposed coordinate descent algorithm with Bien and Tibshirani (2011)'s algorithm. We consider two configurations of $\boldsymbol{\Sigma}$:

- A *sparse* model taken from Bien and Tibshirani (2011) with $\sigma_{i,i+1} = \sigma_{i,i-1} = 0.4, \sigma_{ii} = \delta$ and zero otherwise. Here, δ is chosen such that the condition number of $\boldsymbol{\Sigma}$ is p.
- A *dense* model with $\sigma_{ii} = 2$ and $\sigma_{ij} = 1$ for $i \neq j$.

The algorithm of Bien and Tibshirani (2011) is coded in R with its built-in functions. To be comparable to it, we implement the coordinate descent algorithm in R without writing any functions in a compiled language. All computations are performed on a Intel Xeon X5680 3.33 GHz processor.

For either the sparse model or the dense model, we consider problem sizes of $(p, n) = (100, 200)$ and $(p, n) = (200, 400)$, thus a total of four scenarios of model and size combinations. For each scenario, we generate 20 datasets and apply the two algorithms to each of them under a range of ρ values. All computations are initialized at the sample covariance matrix, i.e., $\boldsymbol{\Sigma}^{(0)} = \mathbf{S}$. For Bien and Tibshirani (2011)'s algorithm, we follow the default setting of tuning parameters provided by the "spcov" package (http://cran.r-project.org/web/packages/spcov/index.html). For the coordinate descent algorithm, we use the same criterion as Bien and Tibshirani (2011)'s algorithm to stop the iterations: The procedure stops when the change of the objective function is less than 10^{-3}.

First, we compare the computing speed. The four panels in Fig. 1 display the CPU time under the four scenarios, respectively. In each panel, CPU time in seconds of the two algorithms for each of the 20 datasets is plotted against the shrinkage parameter ρ which is set at five different values that result in a wide range of sparsity levels in the estimated $\boldsymbol{\Sigma}$. As can be seen, the coordinate descent algorithm is in general substantially faster than Bien and Tibshirani (2011)'s algorithm except when a tiny shrinkage parameter is applied to the dense model, i.e., $\rho = 0.01$ in Panel (c) and (d). Moreover, the coordinate descent algorithm seems to be particularly attractive for sparser models as its run time generally decreases when the sparsity level increases. In contrast, the computing time of Bien and Tibshirani (2011)'s algorithm significantly increases as the sparsity level increases under the two dense scenarios. Finally, the computing time of the coordinate descent algorithm appears to have less variability across multiple replications than that of Bien and Tibshirani (2011)'s algorithm, particularly when the estimated $\boldsymbol{\Sigma}$ is sparse. This suggests that the coordinate descent algorithm has more consistent computing time performance.

Next, we examine the ability of the algorithms to find minimum points. To do so, we compute the minimum values of the objective functions achieved by each algorithm. For each dataset and each ρ, We calculate the relative minimum values of the objective function defined as:

$$g(\hat{\boldsymbol{\Sigma}}_{CD}) - g(\hat{\boldsymbol{\Sigma}}_{BT}), \tag{8}$$

where $\hat{\boldsymbol{\Sigma}}_{BT}$ and $\hat{\boldsymbol{\Sigma}}_{CD}$ are the minimum points found by Bien and Tibshirani (2011)'s algorithm and the coordinate descent algorithm, respectively. Thus, a negative value of (8) indicates that the coordinate descent algorithm finds better points than Bien and Tibshirani (2011)'s algorithm, and a smaller relative minimum value indicates a better performance of the coordinate descent algorithm. The four panels in Fig. 2 display the relative minimum values of (8) as functions of the shrinkage parameter ρ for the four scenarios, respectively. As can be seen, the coordinate descent algorithm tends to outperform Bien and Tibshirani (2011)'s

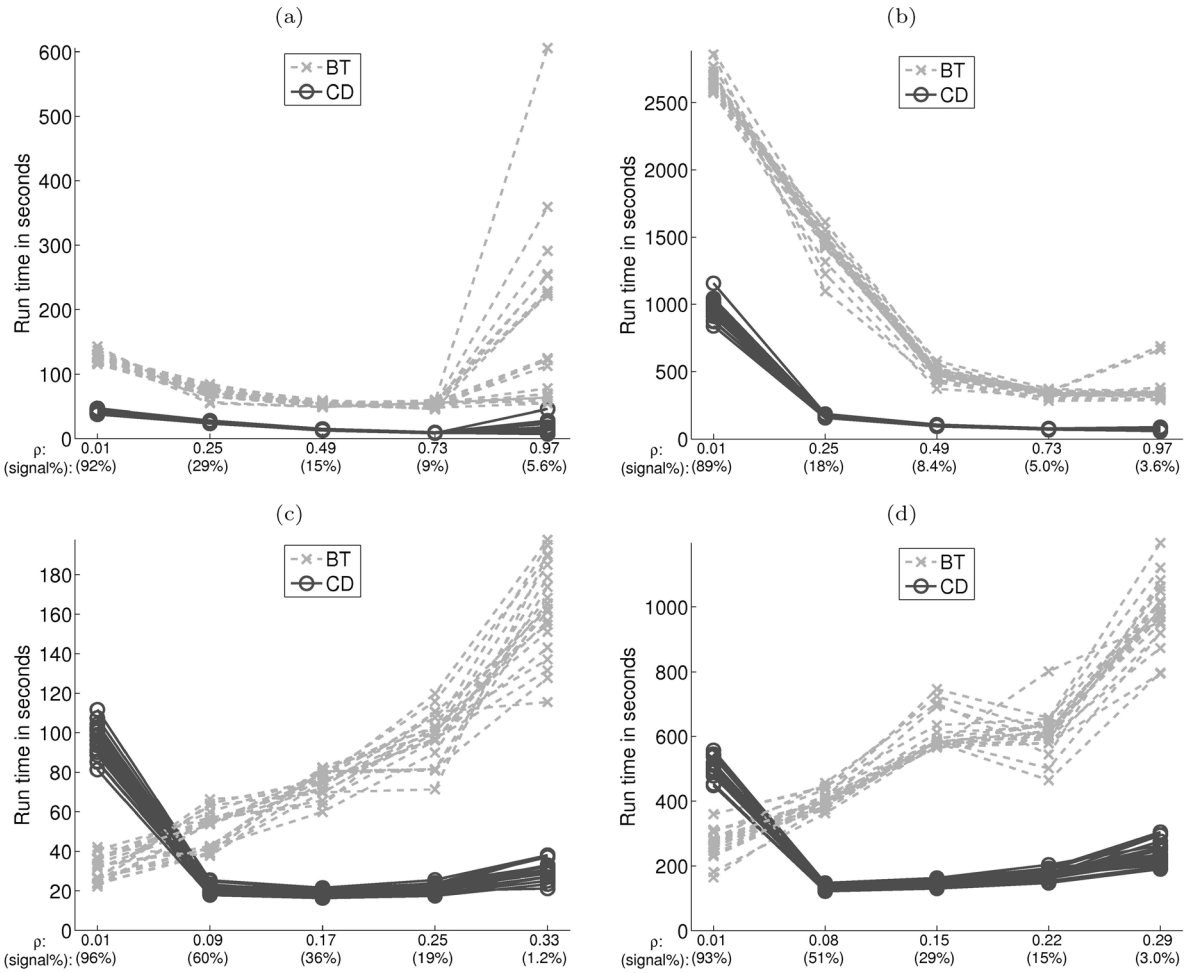

Fig. 1 CPU time is plotted against the shrinkage parameter ρ under four different scenarios for two algorithms. The x axis shows both the shrinkage parameter ρ and the average percentage of non-zero off-diagonal elements across the replicates at that ρ (in parentheses). The y axis shows the computing time in seconds. The four scenarios are: sparse $\mathbf{\Sigma}$ and $(p, n) = (100, 200)$ (**a**); sparse $\mathbf{\Sigma}$ and $(p, n) = (200, 400)$ (**b**); dense $\mathbf{\Sigma}$ and $(p, n) = (100, 200)$ (**c**), and dense $\mathbf{\Sigma}$ and $(p, n) = (200, 400)$ (**d**). The two algorithms are: Bien and Tibshirani (2011) (BT, *dashed gray line*), and coordinate descent (CD, *solid black line*). Each line represents results of one of the 20 replicates. Each computation is initialized at the sample covariance matrix $\mathbf{\Sigma}^{(0)} = \mathbf{S}$

algorithm, as the value of (8) tends to be negative. The only exceptions occur when the shrinkage parameter is tiny (i.e., $\rho = 0.01$) and the estimated $\mathbf{\Sigma}$ has a high percentage of non-zero elements (i.e., about 90%). When the estimated $\mathbf{\Sigma}$ is highly sparse, the coordinate descent algorithm consistently finds points that are more optimal than Bien and Tibshirani (2011)'s algorithm, as is evident from the negative values at the right endpoints of the lines in each panel.

Finally, it is known that, for nonconvex problems, any optimization algorithms are not guaranteed to converge to a global minimum. It is often recommended to run algorithms at multiple initial values. Thus we wish to compare the performance of the algorithms under different initial values. In the previous experiments, all computations are initialized at the full sample covariance matrix $\mathbf{\Sigma}^{(0)} = \mathbf{S}$. To be different, it is natural to initialize them at the other extreme case in which $\mathbf{\Sigma}^{(0)} = \text{diag}(s_{11}, \ldots, s_{pp})$. For each of the four scenarios, we select three different values of ρ such that they represent low-, medium- and high-levels of sparsity, respectively. We repeat the previous experiment at the new initial value of a diagonal matrix $\mathbf{\Sigma}^{(0)} = \text{diag}(s_{11}, \ldots, s_{pp})$.

We record the CPU time, sparsity of the minimum points and the minimum value of the objective function. Table 1 summarizes the observed values of these measures based on 20 replicates by sample mean and sample standard deviation. Three things are worth noting. First, Bien and Tibshirani (2011)'s algorithm seems to get stuck at the initial value of a diagonal matrix in all cases. In contrast, the proposed algorithms work fine and find reasonable minimum points of $\mathbf{\Sigma}$, because the minimum values of the objective function and the level of sparsity are quite close to those obtained from starting at the full sample covariance matrix. We have also tried to initialize Bien and Tibshirani (2011)'s al-

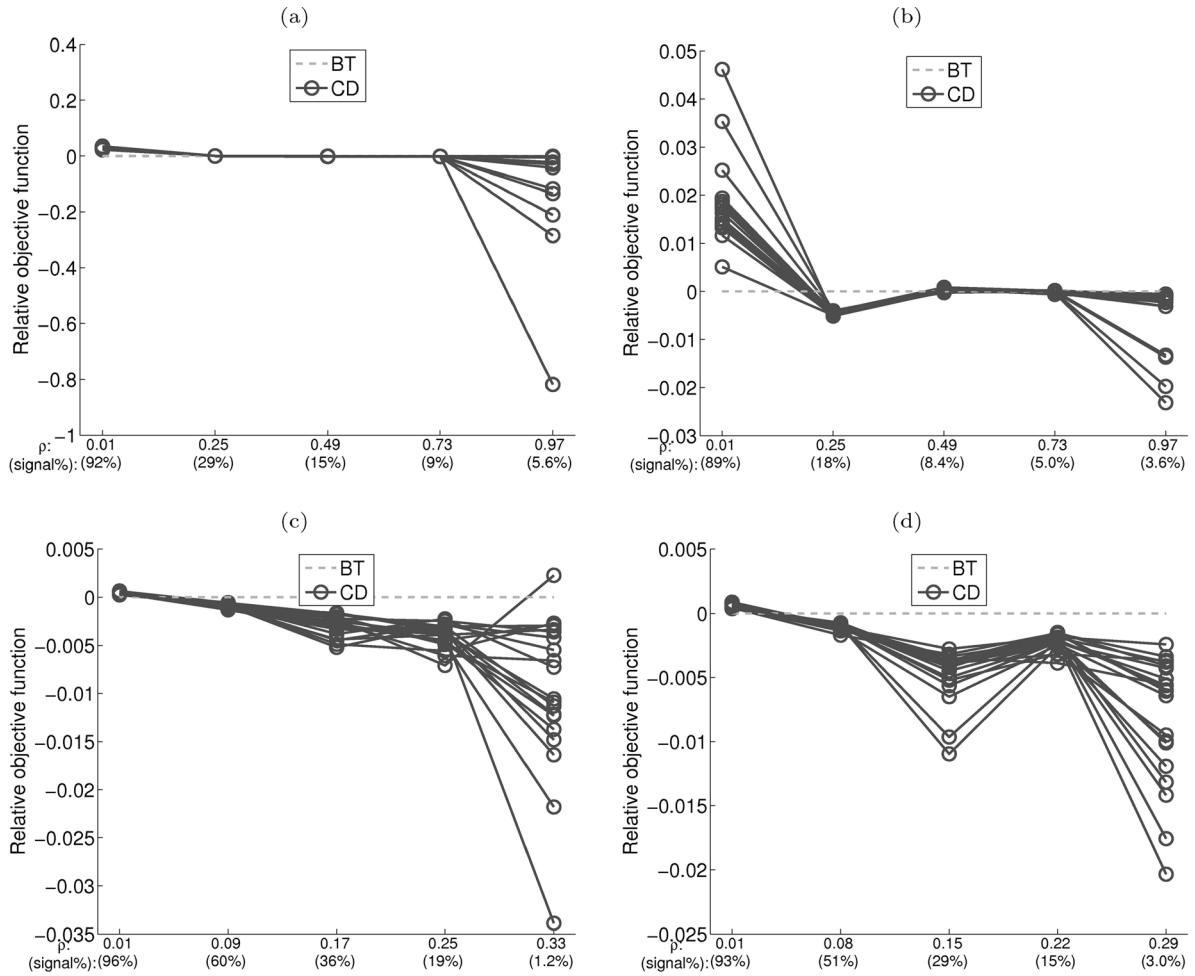

Fig. 2 Relative minimum value of the objective function, defined in (8), is plotted against the shrinkage parameter ρ under four different scenarios for two algorithms. The x axis shows both the shrinkage parameter ρ and the average percentage of non-zero off-diagonal elements across the replicates at that ρ (in parentheses). The y axis shows the relative minimum value of the objective function. The four scenarios are: sparse $\mathbf{\Sigma}$ and $(p, n) = (100, 200)$ (**a**); sparse $\mathbf{\Sigma}$ and $(p, n) = (200, 400)$ (**b**); dense $\mathbf{\Sigma}$ and $(p, n) = (100, 200)$ (**c**), and dense $\mathbf{\Sigma}$ and $(p, n) = (200, 400)$ (**d**). The two algorithms are: Bien and Tibshirani (2011) (BT, *dashed gray line*), and coordinate descent (CD, *solid black line*). Each line represents results of one of the 20 replicates. Each computation is initialized at the sample covariance matrix $\mathbf{\Sigma}^{(0)} = \mathbf{S}$

gorithm at $\mathbf{\Sigma}_{BT}^{(0)} = \text{diag}(s_{11}, \ldots, s_{pp}) + 10^{-3}$, but found that it still gets stuck after a few iterations. Although it may be possible to alleviate this issue by adjusting some of Bien and Tibshirani (2011)'s algorithm's tuning parameters, it may be safe to conclude that Bien and Tibshirani (2011)'s algorithm requires either very careful tuning or performs badly at this important initial value. Second, initial values indeed matter. Comparing the results between full and diagonal initial values, we see substantial differences in all three measures. For example, the limiting points from the diagonal initial matrices are sparser than those from the full initial matrices. This is not surprising because of the drastic difference in sparsity between these two starting points. Third, comparing the minimum values of the objective function achieved by the two algorithms (last two columns), we see that coordinate descent often finds the smaller minimum values than Bien and Tibshirani (2011)'s algorithms. The few exceptions seem to be the cases when ρ is small and the fraction of the number of non-zero elements is large.

4 Alternative algorithms

The coordinate descent algorithm described in Sect. 2 is indeed a cyclic algorithm because it systematically cycles through coordinate directions to minimize the lasso objection function (6). Although we have demonstrated that it outperforms Bien and Tibshirani (2011)'s algorithm, it is of interest to investigate whether this cyclic coordinate descent algorithm can be further improved by alternative algorithms. We propose and explore two additional competitors: a greedy coordinate descent algorithm and a majorize-

Table 1 Performance of the two algorithms starting at two different initial values: "Full", $\mathbf{\Sigma}^{(0)} = \mathbf{S}$; and "Diag", $\mathbf{\Sigma}^{(0)} = \text{diag}(s_{11}, \ldots, s_{pp})$. The "CPU Time" *columns* present the CPU run time in seconds; the "% Nonzero" *columns* present the percentage of nonzero elements in the minimum points; the "Objective Func" *columns* present the minimum value of the objective function. The two algorithms are: Bien and Tibshirani (2011) (BT) and coordinate descent (CD) of Sect. 2. For each measure, we report sample mean and sample standard deviation (in parentheses) based on 20 replicates

Model	p	ρ	Method	CPU Time		% Nonzero		Objective Function	
				Full	Diag	Full	Diag	Full	Diag
Sparse	100	0.01	BT	126	30	0.916	0.000	2.916	79.251
				(7)	(1)	(0.00)	(0.00)	(0.97)	(1.20)
			CD	42	126	0.922	0.924	2.945	2.917
				(3)	(10)	(0.00)	(0.00)	(0.97)	(0.97)
Sparse	100	0.49	BT	53	27	0.148	0.000	57.961	79.251
				(3)	(1)	(0.01)	(0.00)	(1.19)	(1.20)
			CD	14	13	0.145	0.145	57.961	57.961
				(1)	(1)	(0.01)	(0.01)	(1.19)	(1.19)
Sparse	100	0.97	BT	172	27	0.058	0.000	82.890	79.251
				(138)	(1)	(0.01)	(0.00)	(1.37)	(1.20)
			CD	16	0	0.056	0.000	82.805	79.251
				(10)	(0)	(0.01)	(0.00)	(1.34)	(1.20)
Sparse	200	0.01	BT	2677	253	0.885	0.000	−4.089	156.586
				(88)	(10)	(0.00)	(0.00)	(1.18)	(0.97)
			CD	967	942	0.891	0.896	−4.071	−4.101
				(72)	(60)	(0.00)	(0.00)	(1.18)	(1.18)
Sparse	200	0.49	BT	483	173	0.084	0.000	106.455	156.586
				(45)	(5)	(0.00)	(0.00)	(1.15)	(0.97)
			CD	101	115	0.084	0.084	106.456	106.456
				(4)	(4)	(0.00)	(0.00)	(1.15)	(1.15)
Sparse	200	0.97	BT	365	174	0.037	0.000	157.203	156.586
				(108)	(4)	(0.00)	(0.00)	(1.26)	(0.97)
			CD	62	4	0.036	0.000	157.199	156.586
				(8)	(0)	(0.00)	(0.00)	(1.26)	(0.97)
Dense	100	0.01	BT	30	24	0.963	0.000	90.048	169.474
				(6)	(1)	(0.00)	(0.00)	(1.23)	(4.05)
			CD	97	52	0.962	0.961	90.048	90.048
				(8)	(3)	(0.00)	(0.00)	(1.23)	(1.23)
Dense	100	0.17	BT	74	28	0.361	0.000	151.377	169.474
				(6)	(3)	(0.01)	(0.00)	(2.75)	(4.05)
			CD	19	18	0.359	0.358	151.374	151.374
				(2)	(1)	(0.02)	(0.02)	(2.75)	(2.75)
Dense	100	0.33	BT	162	23	0.013	0.000	169.372	169.474
				(23)	(1)	(0.00)	(0.00)	(4.06)	(4.05)
			CD	31	5	0.012	0.002	169.362	169.397
				(5)	(3)	(0.00)	(0.00)	(4.06)	(4.06)
Dense	200	0.01	BT	269	166	0.930	0.000	180.248	340.339
				(49)	(14)	(0.00)	(0.00)	(1.36)	(5.44)
			CD	503	291	0.930	0.929	180.248	180.248
				(28)	(9)	(0.00)	(0.00)	(1.36)	(1.36)
Dense	200	0.15	BT	610	161	0.287	0.000	302.595	340.339
				(58)	(24)	(0.01)	(0.00)	(3.64)	(5.44)
			CD	145	128	0.287	0.286	302.590	302.590
				(9)	(8)	(0.01)	(0.01)	(3.63)	(3.63)
Dense	200	0.29	BT	983	139	0.031	0.000	339.466	340.339
				(98)	(3)	(0.00)	(0.00)	(5.38)	(5.44)
			CD	241	200	0.030	0.022	339.457	339.473
				(33)	(43)	(0.00)	(0.00)	(5.38)	(5.39)

Table 2 Performance of alternative algorithms under four scenarios in Sect. 3. The four scenarios are: sparse $\mathbf{\Sigma}$ and $p = 100$ (*Panel a*); sparse $\mathbf{\Sigma}$ and $p = 200$ (*Panel b*); dense $\mathbf{\Sigma}$ and $p = 100$ (*Panel c*); and dense $\mathbf{\Sigma}$ and $p = 200$ (*Panel d*). The three algorithms are: the cyclic coordinate descent (CD) of Sect. 2, the greedy coordinate descent algorithm (GD) and the majorize-minimize MM algorithm (MM) of Sect. 4. The "Time" *columns* present the CPU run time in seconds; the "% Nonzero" *columns* present the percentage of nonzero elements in the minimum points; the "Objective Func" *columns* present the minimum value of the objective function. For each measure, we report sample mean and sample standard deviation (in parentheses) based on 20 replicates

ρ	Method	Time	% Nonzero	Obj Func
		Panel a: Sparse $p = 100$		
0.01	CD	42	0.924	2.917
		(3)	(0.00)	(0.97)
	MM	16	0.923	2.906
		(1)	(0.00)	(0.97)
	GD	313	0.917	2.908
		(22)	(0.00)	(0.97)
0.49	CD	14	0.145	57.961
		(1)	(0.01)	(1.19)
	MM	24	0.179	58.333
		(1)	(0.01)	(1.19)
	GD	52	0.145	57.961
		(4)	(0.01)	(1.19)
0.97	CD	16	0.000	79.251
		(10)	(0.00)	(1.20)
	MM	68	0.073	83.678
		(53)	(0.01)	(1.36)
	GD	24	0.056	82.805
		(9)	(0.01)	(1.34)
		Panel c: Dense $p = 100$		
0.01	CD	97	0.961	90.048
		(8)	(0.00)	(1.23)
	MM	19	0.967	90.049
		(1)	(0.00)	(1.23)
	GD	266	0.963	90.055
		(20)	(0.00)	(1.23)
0.17	CD	19	0.358	151.374
		(2)	(0.02)	(2.75)
	MM	38	0.431	151.491
		(1)	(0.01)	(2.75)
	GD	45	0.360	151.375
		(3)	(0.01)	(2.75)
0.33	CD	31	0.002	169.397
		(5)	(0.00)	(4.06)
	MM	86	0.058	169.783
		(10)	(0.01)	(4.06)
	GD	32	0.012	169.361
		(5)	(0.00)	(4.05)

ρ	Method	Time	% Nonzero	Obj Func
		Panel b: Sparse $p = 200$		
0.01	CD	967	0.896	−4.101
		(72)	(0.00)	(1.18)
	MM	306	0.887	−4.139
		(7)	(0.00)	(1.18)
	GD	2959	0.885	−4.134
		(159)	(0.00)	(1.18)
0.49	CD	101	0.084	106.456
		(4)	(0.00)	(1.15)
	MM	358	0.086	107.231
		(11)	(0.00)	(1.15)
	GD	383	0.084	106.455
		(20)	(0.00)	(1.15)
0.97	CD	62	0.000	156.586
		(8)	(0.00)	(0.97)
	MM	369	0.036	158.669
		(52)	(0.00)	(1.25)
	GD	122	0.036	157.199
		(8)	(0.00)	(1.26)
		Panel d: Dense $p = 200$		
0.01	CD	503	0.929	180.248
		(28)	(0.00)	(1.36)
	MM	257	0.933	180.250
		(9)	(0.00)	(1.36)
	GD	1566	0.932	180.263
		(86)	(0.00)	(1.36)
0.15	CD	145	0.286	302.590
		(9)	(0.01)	(3.63)
	MM	583	0.326	302.823
		(14)	(0.01)	(3.63)
	GD	305	0.287	302.592
		(16)	(0.01)	(3.63)
0.29	CD	241	0.022	339.473
		(33)	(0.00)	(5.39)
	MM	1104	0.066	340.081
		(57)	(0.01)	(5.38)
	GD	240	0.030	339.459
		(27)	(0.00)	(5.38)

minimize MM algorithm. We briefly describe these two algorithms below.

The greedy coordinate descent algorithm updates along the coordinate direction that gives the largest gradient descent. It has been implemented and explored, for example, in nonparametric wavelet denoising models (Sardy et al. 2000) and in l_1 and l_2 regressions (Wu and Lange 2008). In the covariance graphical lasso model, we implement a version of the greedy coordinate descent algorithm based on the theoretical results of directional derivatives (Wu and Lange 2008) for identifying the deepest gradient descent coordinate in each iteration for solving (6).

The majorize-minimize MM algorithm is a general optimization algorithm that creates a surrogate function that is easier to minimize than the original objection function (Hunter and Lange 2004). The design of the surrogate function is key to its efficiency. For the covariance graphical lasso model, Bien and Tibshirani (2011)'s MM algorithm uses a surrogate function that is minimized by elementwise soft-thresholding. We consider an alternative surrogate function that can be minimized by updating one column and one row at a time. The details of our new MM algorithm is provided in the Appendix.

We use the greedy coordinate descent algorithm and the new MM algorithm for fitting the covariance graphical lasso model in the same experiment in Table 1. The results are presented in Table 2. For easy comparison, the results of the cyclic coordinate descent algorithm in Table 1 is also shown. Only results from runs initialized at the sample covariance matrix are reported. The relative performance of computing time from runs initialized at the diagonal covariance matrix is similar. Comparing algorithms at a fixed ρ within each scenario, we observe that the cyclic coordinate descent algorithm is the fastest except when the shrinkage parameter is tiny (i.e., $\rho = 0.01$) and the resulted estimate of $\mathbf{\Sigma}$ has a high percentage of non-zero elements (i.e., about 90%). In these exceptions, the new MM algorithm seems to be the fastest. Since sparse estimates of $\mathbf{\Sigma}$ is more interesting than dense estimates, the cyclic coordinate descent algorithm is perhaps the most desirable algorithm among the three. The greedy coordinate descent algorithm is substantially slower than the cyclic coordinate descent algorithm. This is consistent with Wu and Lange (2008) which report a better performance of the cyclic coordinate descent algorithm over the greedy coordinate descent algorithm for fitting l_2 regressions under lasso penalty, and may indicate that the extra computational cost of finding the deepest gradient descent is not compensated by the reduced number of iterations until convergence.

5 Discussion

We have developed a coordinate descent algorithm for fitting sparse covariance graphical lasso models. The new algorithm is shown to be much easier to implement, significantly faster to run and numerically more stable than the algorithm of Bien and Tibshirani (2011). Both MATLAB and R software packages implementing the new algorithms for solving covariance graphical models are freely available from the author's website of the paper.

Appendix: Details of the majorize-minimize MM algorithm in Sect. 4

Consider $\sqrt{\sigma_{ij}^2 + \epsilon}$ as an approximation to $|\sigma_{ij}|$ for a small $\epsilon > 0$. Consequently, the original objective function (1) can be approximated by

$$\log(\det \mathbf{\Sigma}) + \operatorname{tr}\big(\mathbf{S}\mathbf{\Sigma}^{-1}\big) + 2\rho \sum_{i<j} \sqrt{\sigma_{ij}^2 + \epsilon} + \rho \sum_i \sigma_{ii}. \quad (9)$$

Note the inequality

$$\sqrt{\sigma_{ij}^2 + \epsilon} \le \sqrt{\big(\sigma_{ij}^{(k)}\big)^2 + \epsilon} + \frac{\sigma_{ij}^2 - (\sigma_{ij}^{(k)})^2}{2\sqrt{(\sigma_{ij}^{(k)})^2 + \epsilon}},$$

for a fixed $\sigma_{ij}^{(k)}$ and all σ_{ij}. Then (9) is majorized by

$$Q\big(\mathbf{\Sigma} \mid \mathbf{\Sigma}^{(k)}\big) = \log(\det \mathbf{\Sigma}) + \operatorname{tr}\big(\mathbf{S}\mathbf{\Sigma}^{-1}\big) + \rho \sum_{i<j} \frac{\sigma_{ij}^2}{\sqrt{(\sigma_{ij}^{(k)})^2 + \epsilon}} + \rho \sum_i \sigma_{ii}. \quad (10)$$

The minimize-step in MM then minimizes (10) along each column (row) of $\mathbf{\Sigma}$. Without loss of generality, consider the last column and row. Partition $\mathbf{\Sigma}$ and $\mathbf{S}$ as in (2) and consider the same transformation from $(\boldsymbol{\sigma}_{12}, \sigma_{22})$ to $(\boldsymbol{\beta} = \boldsymbol{\sigma}_{12}, \gamma = \sigma_{22} - \boldsymbol{\sigma}_{12}' \mathbf{\Sigma}_{11}^{-1} \boldsymbol{\sigma}_{12})$. The four terms in (10) can be written as functions of $(\boldsymbol{\beta}, \gamma)$

$$\log(\det \mathbf{\Sigma}) = \log(\gamma) + c_1,$$

$$\operatorname{tr}\big(\mathbf{S}\mathbf{\Sigma}^{-1}\big) = \boldsymbol{\beta}' \mathbf{\Sigma}_{11}^{-1} \mathbf{S}_{11} \mathbf{\Sigma}_{11}^{-1} \boldsymbol{\beta} \gamma^{-1} - 2\mathbf{s}_{12}' \mathbf{\Sigma}_{11}^{-1} \boldsymbol{\beta} \gamma^{-1} + s_{22} \gamma^{-1} + c_2,$$

$$\rho \sum_{i<j} \frac{\sigma_{ij}^2}{\sqrt{(\sigma_{ij}^{(k)})^2 + \epsilon}} = \rho \boldsymbol{\beta}' \mathbf{D}^{-1} \boldsymbol{\beta},$$

$$\mathbf{D} = \operatorname{diag}\Big(\sqrt{\big(\sigma_{ij}^{(k)}\big)^2 + \epsilon}\Big),$$

$$\rho \sum_i \sigma_{ii} = \rho\big(\boldsymbol{\beta}' \mathbf{\Sigma}_{11}^{-1} \boldsymbol{\beta} + \gamma\big) + c_3,$$

where c_1, c_2 and c_3 are constants not involving $(\boldsymbol{\beta}, \gamma)$. Dropping off c_1,c_2 and c_3 from (10), we only have to minimize

$$Q\big(\boldsymbol{\beta}, \gamma \mid \mathbf{\Sigma}^{(k)}\big) = \log(\gamma) + \boldsymbol{\beta}' \mathbf{\Sigma}_{11}^{-1} \mathbf{S}_{11} \mathbf{\Sigma}_{11}^{-1} \boldsymbol{\beta} \gamma^{-1} - 2\mathbf{s}_{12}' \mathbf{\Sigma}_{11}^{-1} \boldsymbol{\beta} \gamma^{-1} + s_{22} \gamma^{-1} + \rho \boldsymbol{\beta}' \mathbf{D}^{-1} \boldsymbol{\beta} + \rho \boldsymbol{\beta}' \mathbf{\Sigma}_{11}^{-1} \boldsymbol{\beta} + \rho\gamma. \quad (11)$$

For γ, it is easy to derive from (11) that the conditional minimum point given $\boldsymbol{\beta}$ is the same as in (5). For $\boldsymbol{\beta}$, (11) can be written as a function of $\boldsymbol{\beta}$,

$$Q\big(\boldsymbol{\beta} \mid \gamma, \Sigma^{(k)}\big) = \boldsymbol{\beta}' \big(\mathbf{V} + \rho \mathbf{D}^{-1}\big) \boldsymbol{\beta} - 2\mathbf{u}' \boldsymbol{\beta},$$

where $\mathbf{V}$ and $\mathbf{u}$ are defined in (6). This implies that the conditional minimum point of $\boldsymbol{\beta}$ is $\boldsymbol{\beta} = (\mathbf{V} + \rho \mathbf{D}^{-1})^{-1} \mathbf{u}$. Cycling

through every column always drives down the approximated objective function (9). In our implementation, the value of ϵ is chosen as follows. The approximation error of (9) to (1) is

$$2\rho\sum_{i<j}\left(\sqrt{\sigma_{ij}^2+\epsilon}-|\sigma_{ij}|\right)=2\rho\sum_{i<j}\frac{\epsilon}{\sqrt{\sigma_{ij}^2+\epsilon}+|\sigma_{ij}|}$$
$$<2\rho\sum_{i<j}\frac{\epsilon}{\sqrt{\epsilon}}=\rho p(p-1)\sqrt{\epsilon}.$$

Note that the algorithm stops when the change of the objective function is less than 10^{-3}. We choose ϵ such that $\rho p(p-1)\sqrt{\epsilon}=0.001$, i.e., $\epsilon=(0.001/(\rho(p-1)p))^2$, to ensure that the choice of ϵ has no more influence on the estimated $\mathbf{\Sigma}$ than the stopping rule.

References

Bien, J., Tibshirani, R.J.: Sparse estimation of a covariance matrix. Biometrika **98**(4), 807–820 (2011). doi:10.1093/biomet/asr054

Breheny, P., Huang, J.: Coordinate descent algorithms for nonconvex penalized regression, with applications to biological feature selection. Ann. Appl. Stat. **1**, 232–253 (2011)

Dempster, A.: Covariance selection. Biometrics **28**, 157–175 (1972)

Friedman, J., Hastie, T., Höfling, H., Tibshirani, R.: Pathwise coordinate optimization. Ann. Appl. Stat. **1**(2), 302–332 (2007)

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

Fu, W.J.: Penalized regressions: the bridge versus the lasso. J. Comput. Graph. Stat. **7**(3), 397–416 (1998)

Hunter, D.R., Lange, K.: A tutorial on MM algorithms. Am. Stat. **58**(1) (2004). doi:10.2307/27643496

Lin, N.: A penalized likelihood approach in covariance graphical model selection. Ph.D. Thesis, National University of Singapore (2010)

Sardy, S., Bruce, A.G., Tseng, P.: Block coordinate relaxation methods for nonparametric wavelet denoising. J. Comput. Graph. Stat. **9**(2), 361–379 (2000)

Tseng, P.: Convergence of a block coordinate descent method for nondifferentiable minimization. J. Optim. Theory Appl. **109**, 475–494 (2001)

Wu, T.T., Lange, K.: Coordinate descent algorithms for lasso penalized regression. Ann. Appl. Stat. **2**(1), 224–244 (2008)