An Efficient Parallel Block Coordinate Descent Algorithm for Large-scale Precision Matrix Estimation using Graphics Processing Units

Seunghwan Lee

Department of Statistics
Inha University

May 28, 2021

Joint work with Prof. Choi, Y.-G. and Prof. Yu, D.

Outline

- Precision Matrix Estimation
- Convex correlation selection method (CONCORD)
- Edge-Coloring in Graph Theory
- Parallel Coordinate Descent algorithm for CONCORD using Graphics Processing Units
- Numerical study
- Summary

Estimation of Precision Matrix Ω

- Observed data ($n \times p$ dimensional matrix):

$$\mathbf{X} = (X^1, X^2, \ldots, X^n)^T,$$

where $X^k = (X_1^k, \cdots, X_p^k)^T \sim \mathcal{N}(\mathbf{0}, \Sigma)$.

- *Conditional Independence*:

$$\omega_{ij} = 0 \iff X_i \perp\!\!\!\perp X_j \mid \{X_k : k \neq i, j\},$$

where ω_{ij} is the (i, j)-th element of the *precision matrix* $\Omega \equiv \Sigma^{-1}$.

Estimation of Precision Matrix Ω

- Loglikelihood of Ω

$$\log L(\Omega) = \frac{n}{2} \log \det \Omega - \frac{1}{2} \sum_{k=1}^{n} (X^k)^{\mathrm{T}} \Omega \, X^k - \frac{np}{2} \log 2\pi$$

- Maximum likelihood estimator (MLE) of Ω

$$\widehat{\Omega} = S^{-1}, \text{ where } S = \frac{1}{n} \sum_{k=1}^{n} X^k (X^k)^T$$

- In high-dimensional case $p > n$, S is ill-conditioned and singular.

  $\Rightarrow S^{-1}$ can not be obtained from S.

ℓ_1-regularization Methods

- Maximum likelihood approach

$$\widehat{\Omega} = \underset{\omega^{ij}, 1 \le i,j \le p}{\operatorname{argmin}} \left\{ -\log\det(\Omega) + \operatorname{tr}(S\Omega) + \lambda \sum_{\substack{i \neq j \\ (\text{or } 1 \le i,j \le p)}} |\omega_{ij}| \right\}$$

subject to Ω is positive definite and symmetric,
where $S = \frac{1}{n} \sum_{k=1}^{n} X^k (X^k)^T$ and $\lambda \ge 0$.

- Existing methods:
Yuan and Lin (2007);
Graphical lasso (Friedman et al., 2008; Witten et al., 2012;
Mazumerder and Hastie, 2012).

ℓ_1-regularization Methods

- ℓ_1-minimization approach

$$\widehat{\Omega} = \underset{\Omega}{\operatorname{argmin}} \; \|\Omega\|_1 \text{ subject to } \|S\Omega - I_p\|_\infty \leq \lambda,$$

  where $S = \frac{1}{n}\sum_{k=1}^{n} X^k (X^k)^T$, $\lambda \geq 0$, $\|A\|_1 = \sum_{i=1}^{n}\sum_{j=1}^{p} |a_{ij}|$ and $\|A\|_\infty = \max_{1\leq i\leq n, 1\leq j\leq p} |a_{ij}|$ for $A = (a_{ij}) \in \mathbb{R}^{n\times p}$.

- Existing methods:
  CLIME (Cai et al., 2011);
  ACLIME (Cai et al., 2016).

ℓ_1-regularization Methods

- Regression approach

$$X_i = \sum_{j \neq i} \beta_{ij} X_j + \epsilon_i = \sum_{j \neq i} \rho_{ij} \sqrt{\frac{\omega_{jj}}{\omega_{ii}}} X_j + \epsilon_i,$$

where $\beta_{ij} = -\omega_{ij}/\omega_{ii} = \rho_{ij}\sqrt{\omega_{jj}/\omega_{ii}}$, and ϵ_i is uncorrelated with $X_{-[i]} = \{X_j \mid j \neq i,\ 1 \leq j \leq p\}$ and has mean 0 and variance $1/\omega_{ii}$.

$(\omega_{ij} = 0 \iff \beta_{ij} = 0 \iff \rho_{ij} = 0)$

- Existing methods:

  Neighborhood selection (Meinshausen and Bühlmann, 2006);
  SPACE (Peng et al., 2009);
  Matrix inversion with scaled lasso (Sun and Zhang, 2013);
  CONCORD (Khare et al., 2015).

CONvex CORrelation selection methoD (CONCORD)

- Consider a log-pseudolikelihood function $L(\Omega;\lambda)$

$$L(\Omega;\lambda) = -\sum_{i=1}^{p} n\log\omega_{ii} + \frac{1}{2}\sum_{i=1}^{p}\sum_{k=1}^{n}\Big(\omega_{ii}X_i^k + \sum_{j\neq i}\omega_{ij}X_j^k\Big)^2 + \lambda\sum_{i<j}|\omega_{ij}|$$

- $L(\Omega;\lambda) = L(\omega_D, \omega_{-D};\lambda)$ is convex for (ω_D, ω_{-D}), where $\omega_D = (\omega_{11}, \omega_{22}, \ldots, \omega_{pp})^T$ and $\omega_{-D} = (\omega_{12}, \omega_{13}, \ldots, \omega_{(p-1)p})^T$.
- It is shown that the cyclic coordinatewise minimization guarantees the convergence to a global minimum of $L(\Omega;\lambda)$ if all diagonal elements of the sample covariance matrix are positive.

Coordinate descent (CD) for CONCORD

- Cyclic coordinatewise minimization

1. Given the current solutions $\hat{\omega}_D^{(k)} = (\hat{\omega}_{ii}^{(k)}, 1 \le i \le p)$ at k-th iteration and the current updated solutions $(\hat{\omega}_{st})_{(s,t)\neq(i,j)}$,

$$\hat{\omega}_{ij}^{(k+1)} = \operatorname*{argmin}_{\omega_{ij}} L(\omega_{ij}; \hat{\omega}_D^{(k)}, (\hat{\omega}_{st})_{(s,t)\neq(i,j)}, \lambda)$$

2. Given the current solution $\hat{\omega}_{-D}^{(k+1)} = (\hat{\omega}_{ij}^{(k+1)}, i < j)$,

$$\hat{\omega}_D^{(k+1)} = \operatorname*{argmin}_{\omega_D} L(\omega_D; \hat{\omega}_{-D}^{(k+1)}, \lambda)$$

3. Repeat Steps 1 and 2 until convergence occurs.

Coordinate descent (CD) for CONCORD

- Given the current solution $\hat{\omega}_{-D}^{(k)} = (\hat{\omega}_{ij}^{(k)}, i < j)$ at k-th iteration,

$$\hat{\omega}_{ii}^{(k+1)} = \frac{-\sum_{j\neq i} \hat{\omega}_{ij}^{(k)} T_{ij} + \sqrt{\left(\sum_{j\neq i} \hat{\omega}_{ij}^{(k)} T_{ij}\right)^2 + 4nT_{ii}}}{2T_{ii}} \text{ for } i = 1, 2, \ldots, p, \tag{1}$$

  where T_{ij} denote the (i, j)th element of $\mathbf{X}^T\mathbf{X}$.

- Given the current solution $\hat{\omega}_{D}^{(k)} = (\hat{\omega}_{ii}^{(k)}, 1 \leq i \leq p)$ at k-th iteration,

$$\hat{\omega}_{ij}^{(k+1)} = \frac{\text{Soft}_{\lambda}(-\sum_{j'\neq j} \tilde{\omega}_{ij'} T_{jj'} - \sum_{i'\neq i} \tilde{\omega}_{i'j} T_{ii'})}{T_{ii} + T_{jj}} \text{ for } 1 \leq i < j \leq p, \tag{2}$$

  where $\tilde{\omega}_{st}$ denotes the current iterative solution of ω_{st}, T_{ij} denotes the (i, j)th element of $\mathbf{X}^T\mathbf{X}$, $\text{Soft}_{\tau}(x) = \text{sign}(x)(|x| - \tau)_+$, and $(x)_+ = \max(0, x)$.

Motivation: Parallelizable updating equations

- For the updating equations of $\hat{\omega}_D$, the current updated solution $\hat{\omega}_{ii}$ does not affect the other updating equations of $\hat{\omega}_{jj}$ for $j \neq i$.
- For the updating equations of $\hat{\omega}_{-D}$, $\hat{\omega}_{ij}$ depends on the current iterative solutions $(\hat{\omega}_{i1}, \ldots, \hat{\omega}_{i(j-1)}, \hat{\omega}_{i(j+1)}, \ldots, \hat{\omega}_{ip})$ and $(\hat{\omega}_{1j}, \ldots, \hat{\omega}_{(i-1)j}, \hat{\omega}_{(i+1)j}, \ldots, \hat{\omega}_{pj})$. Thus, the current updated solution $\hat{\omega}_{ij}$ does not affect other updating equations of $\hat{\omega}_{st}$ such that $s \neq i$ and $t \neq j$.

Edge-Coloring

- Edge-Coloring:

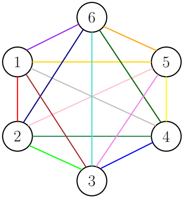
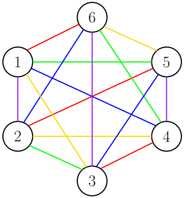
  Edge-coloring is defined as an assignment of *colors to the edges* of a graph, in a way that any pair of adjacent edges (edges sharing at least one vertices) is colored in different colors.

- A special case of Baranyai's Theorem with a complete graph $\mathcal{K}_p$

Theorem (Baranyai's Theorem)

Suppose that $\mathcal{K}_p$ is an undirected complete graph with p nodes. If p is even, $\mathcal{K}_p$ is edge-colorable with at least $p-1$ colors. If p is odd, then $\mathcal{K}_p$ is edge-colorable with at least p colors.

Example of Edge-Coloring

Coloring scheme	with mutually distinct colors	with minimal number of colors
$\mathcal{K}_p$ with $p = 6$		
# of colors	$p(p-1)/2 = 15$	$p-1 = 5$
# of edge(s) for each color	1	$p/2 = 3$
Collections of edges of the same colors ($(i,j) \to ij$)	$\{12\}, \{13\}, \{14\}, \{15\}, \{16\},$ $\{23\}, \{24\}, \{25\}, \{26\}, \{34\},$ $\{35\}, \{36\}, \{45\}, \{46\}, \{56\}$	$\{16, 25, 34\}, \{15, 23, 46\},$ $\{14, 26, 35\},$ $\{13, 24, 56\}, \{12, 36, 45\}$

Analogy between edge-coloring and update ordering

- Analogy between edge-coloring and update ordering:

  (A) Associate the edge-coloring of edge (i, j) by k-th color
  with the update of $\hat{\omega}_{ij}$ as in (2) at the k-th step.

- Edge-coloring and CD (serial update):

  Coloring all edges by colors 1 through $p(p-1)/2$

- Edge-coloring and Parallel CD:

  Coloring multiple edges (i, j) with the same k-th color, which means that the associated ω_{ij}'s are simultaneously updated given the same current iterate.

Circle method for edge-coloring in $\mathcal{K}_p$

- Partitioning an edge set of $\mathcal{K}_p$ as $p-1$ (p) *equi-color subsets* in which edges have same color when p is even (odd).

- To handle the differences between even p and odd p, we consider a variable p_{even}, which is defined as $p_{even} = p$ if p is even and $p_{even} = p + 1$ if p is odd.

Circle method for edge-coloring in $\mathcal{K}_p$

(i) Clockwise rotation for the round-robin table with the $(1, 1)$ element is fixed:

1	2	3	$\cdots$	$p_{even}/2$
p_{even}	$p_{even}-1$	$p_{even}-2$	$\cdots$	$p_{even}/2+1$

$\rightarrow$

1	p_{even}	2	$\cdots$	$p_{even}/2-1$
$p_{even}-1$	$p_{even}-2$	$p_{even}-3$	$\cdots$	$p_{even}/2$

$\rightarrow \cdots \rightarrow$

1	3	4	$\cdots$	$p_{even}/2+1$
2	p_{even}	$p_{even}-1$	$\cdots$	$p_{even}/2+2$

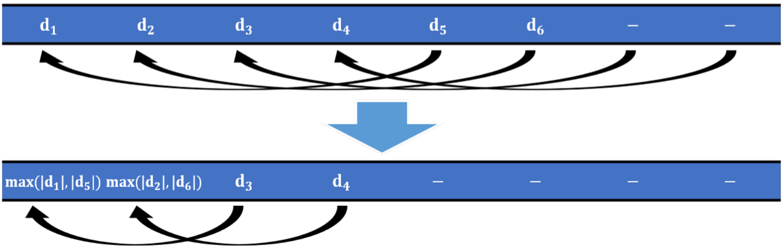
(ii) Define matching pairs: for each table in (i), two indices in each columns of the given table defines a matching pair. For example, in the first table in (i), the matching pairs are
$\{(1, p_{even}), (2, p_{even}-1), \ldots, (p_{even}/2, p_{even}/2+1)\}$.

(iii) Discard a pair containing (p_{even}) index in (ii) if p is odd.

Cyclic Reduction for $\|\Omega^{(k)} - \hat{\Omega}\|_\infty$

- $\|A\|_\infty = \max_{1\le i\le n, 1\le j\le p} |a_{ij}|$ for $A = (a_{ij}) \in \mathbb{R}^{n\times p}$.
- Graphical representation of cyclic reduction:

$$\begin{pmatrix} \hat{\omega}_{11}^{(k+1)} - \hat{\omega}_{11}^{(k)} & \hat{\omega}_{12}^{(k+1)} - \hat{\omega}_{12}^{(k)} & \hat{\omega}_{13}^{(k+1)} - \hat{\omega}_{13}^{(k)} \\ & \hat{\omega}_{22}^{(k+1)} - \hat{\omega}_{22}^{(k)} & \hat{\omega}_{23}^{(k+1)} - \hat{\omega}_{23}^{(k)} \\ & & \hat{\omega}_{33}^{(k+1)} - \hat{\omega}_{33}^{(k)} \end{pmatrix} \Rightarrow (d_1, d_2, d_3, d_4, d_5, d_6)$$



Comparison of Computation times

- Dimension (# of nodes): 500, 1000, 2500, 5000
- Sample size (# of samples): 500, 1000, 2000
- Tuning parameter: $\lambda^* = \lambda/n = 0.1, 0.3$
- Network structure: AR(2) and Scale-free networks
- Generate samples
$$\mathbf{X}^1, \mathbf{X}^2, \ldots, \mathbf{X}^n \sim N(0, \Sigma),$$
where $\Sigma = \Omega^{-1}$ and Ω is from a network structure.
- We generate 10 data sets and report the averages of computation times ($\delta_{tol} = 10^{-5}$).
- System specs : Intel Xeon(R) W-2175 CPU (2.50GHz, Max Turbo (4.30GHz)) and 128 GB RAM with NVIDIA GeForce GTX 1080 Ti

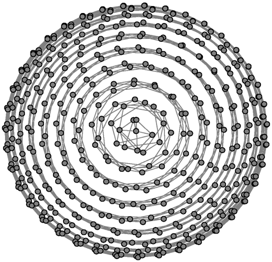
Comparison of Computation times

- Implemented algorithms
  - CD-BLAS:
    Implemented CD algorithm using BLAS (`gconcord`)
  - CD-NAIVE:
    Implemented CD algorithm using C language only.
  - PCD-CPU:
    Implemented PCD algorithm without CUDA C (running on CPU)
  - PCD-GPU:
    Implemented PCD algorithm with CUDA C (running on GPU)

AR(2) Network

AR(2) network has edges between j-th node and $(j-1)$-th node for $j = 2, 3, \ldots, p$ and edges between j-th node and $(j-2)$-th node for $j = 3, 4, \ldots, p$.

$$\Omega = (\omega_{ij})_{1 \le i,j \le p} = \begin{cases} 1 & \text{if } i = j \\ 0.45 & \text{if } |i-j| = 1 \\ 0.4 & \text{if } |i-j| = 2 \\ 0 & \text{otherwise} \end{cases}$$



Scale-free Network

Degrees of nodes follow power law distribution having a form $P(k) \propto k^{-\alpha}$, where $P(k)$ is a fraction of nodes having k connections and α is a preferential attachment parameter. We set $\alpha = 2.3$ and generate a scale-free network by using the Barabasi and Albert (BA) model.

$$(i)\ \tilde{\Omega} = (\tilde{\omega}_{ij})_{1 \le i,j \le p} = \begin{cases} 1 & \text{if } i = j \\ U & \text{if } (i,j) \in E \\ 0 & \text{otherwise} \end{cases}$$

$$U \sim \mathcal{U}([-1, -0.5] \cup [0.5, 1]).$$

$$(ii)\ \Omega = (\omega_{ij})_{1 \le i,j \le p} = \frac{\tilde{\omega}_{ij}}{1.25 \sum_{k \ne i} |\tilde{\omega}_{ik}|}$$

$$(iii)\ \Omega = (\Omega + \Omega^T)/2$$

$$(iv)\ \omega_{ii} = 1 \text{ for } i = 1, 2, \ldots, p$$

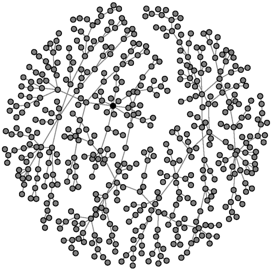


Table 1: Summary for AR(2) network with $\lambda^* = 0.1$

λ^*	n	p	Computation time (sec.)				Iteration		$\lvert\hat{E}\rvert$	
			CD-BLAS	CD-NAIVE	PCD-CPU	PCD-GPU	CD	PCD	CD	PCD
0.1	500	500	3.22	3.18	3.31	0.89	26.40	26.10	1976.70	1976.70
			(0.02)	(0.02)	(0.02)	(0.01)	(0.16)	(0.18)	(9.52)	(9.52)
		1000	13.09	26.28	28.66	4.87	26.90	26.80	5078.90	5078.90
			(0.09)	(0.17)	(0.18)	(0.04)	(0.18)	(0.20)	(14.19)	(14.19)
		2500	86.01	451.03	513.18	56.72	27.30	27.10	20327.30	20327.50
			(0.83)	(4.30)	(3.28)	(0.38)	(0.26)	(0.18)	(45.87)	(45.90)
		5000	378.04	3646.43	4149.79	404.75	27.70	27.30	64307.20	64307.40
			(2.94)	(19.49)	(51.93)	(2.29)	(0.15)	(0.15)	(69.26)	(69.13)
	1000	500	2.07	3.16	3.29	0.88	25.70	25.50	1407.20	1407.20
			(0.01)	(0.02)	(0.02)	(0.01)	(0.15)	(0.17)	(5.05)	(5.05)
		1000	25.90	25.84	28.27	4.76	26.20	26.10	2825.20	2825.20
			(0.14)	(0.13)	(0.22)	(0.03)	(0.13)	(0.18)	(4.50)	(4.50)
		2500	167.90	428.93	490.24	54.91	26.20	26.20	7216.80	7216.80
			(1.60)	(3.38)	(4.18)	(0.28)	(0.13)	(0.13)	(6.92)	(6.92)
		5000	694.62	3419.50	3862.95	385.66	26.20	26.00	14806.20	14806.20
			(3.87)	(17.43)	(17.96)	(0.05)	(0.13)	(0.00)	(14.28)	(14.28)
	2000	500	2.12	3.19	3.36	0.85	25.00	25.10	1393.10	1393.10
			(0.00)	(0.00)	(0.01)	(0.00)	(0.00)	(0.10)	(3.74)	(3.74)
		1000	21.81	25.76	27.88	4.65	25.70	25.40	2803.10	2803.10
			(0.39)	(0.16)	(0.26)	(0.03)	(0.15)	(0.16)	(4.70)	(4.70)
		2500	342.84	433.35	495.12	54.54	25.90	25.90	7006.90	7006.90
			(1.26)	(1.65)	(1.93)	(0.21)	(0.10)	(0.10)	(9.79)	(9.79)
		5000	1389.95	3440.95	3929.09	386.23	26.00	26.00	14009.30	14009.40
			(6.07)	(13.14)	(11.22)	(0.12)	(0.00)	(0.00)	(9.38)	(9.35)

Table 2: Summary for AR(2) network with $\lambda^* = 0.3$

λ^*	n	p	Computation time (sec.)				Iteration		$\lvert\hat{E}\rvert$	
			CD-BLAS	CD-NAIVE	PCD-CPU	PCD-GPU	CD	PCD	CD	PCD
0.3	500	500	1.69 (0.06)	1.70 (0.05)	1.73 (0.06)	0.50 (0.02)	13.90 (0.46)	13.40 (0.52)	859.50 (5.00)	859.50 (5.00)
		1000	7.00 (0.13)	14.19 (0.26)	14.80 (0.45)	2.55 (0.07)	14.40 (0.27)	13.70 (0.40)	1722.20 (5.87)	1722.20 (5.87)
		2500	45.52 (0.72)	240.36 (3.57)	267.74 (3.37)	29.61 (0.38)	14.50 (0.22)	14.10 (0.18)	4297.80 (7.12)	4297.80 (7.12)
		5000	210.20 (4.05)	1937.84 (38.32)	2289.35 (23.56)	215.59 (2.47)	14.80 (0.29)	14.50 (0.17)	8624.10 (17.08)	8624.10 (17.08)
	1000	500	1.06 (0.02)	1.59 (0.03)	1.62 (0.03)	0.46 (0.01)	12.40 (0.22)	12.10 (0.23)	853.60 (4.66)	853.60 (4.66)
		1000	12.06 (0.13)	12.31 (0.13)	13.05 (0.15)	2.22 (0.02)	12.20 (0.13)	11.80 (0.13)	1698.00 (5.80)	1698.00 (5.80)
		2500	78.69 (1.56)	203.18 (3.53)	225.83 (4.36)	25.51 (0.48)	12.50 (0.22)	12.10 (0.23)	4268.10 (11.70)	4268.10 (11.70)
		5000	350.79 (7.29)	1718.20 (35.40)	1901.72 (39.33)	182.99 (3.18)	13.00 (0.30)	12.30 (0.21)	8521.50 (11.65)	8521.50 (11.65)
	2000	500	1.12 (0.01)	1.64 (0.02)	1.62 (0.00)	0.45 (0.01)	11.80 (0.13)	11.00 (0.00)	854.20 (3.14)	854.20 (3.14)
		1000	10.81 (0.21)	12.22 (0.16)	12.76 (0.11)	2.11 (0.02)	11.60 (0.16)	11.10 (0.10)	1717.00 (4.96)	1717.00 (4.96)
		2500	159.15 (0.09)	204.24 (0.27)	216.04 (1.86)	24.00 (0.32)	12.00 (0.00)	11.30 (0.15)	4291.10 (5.94)	4291.10 (5.94)
		5000	637.67 (5.89)	1597.47 (15.28)	1796.61 (32.73)	171.11 (2.46)	12.10 (0.10)	11.50 (0.17)	8578.20 (14.48)	8578.30 (14.51)

Numerical study

Table 3: Summary for scale-free network with $\lambda^* = 0.1$

λ^*	n	p	Computation time (sec.)				Iteration		$\lvert\hat{E}\rvert$	
			CD-BLAS	CD-NAIVE	PCD-CPU	PCD-GPU	CD	PCD	CD	PCD
0.1	500	500	1.33 (0.02)	1.35 (0.02)	1.53 (0.04)	0.46 (0.01)	11.30 (0.15)	12.20 (0.29)	2348.30 (13.88)	2348.30 (13.88)
		1000	5.64 (0.15)	11.41 (0.30)	13.18 (0.28)	2.34 (0.05)	11.90 (0.31)	12.60 (0.27)	8090.90 (22.41)	8090.90 (22.41)
		2500	43.59 (1.59)	228.30 (8.18)	269.74 (9.14)	30.84 (1.05)	14.20 (0.51)	14.60 (0.50)	42601.50 (34.34)	42601.50 (34.34)
		5000	193.67 (3.84)	1872.72 (37.21)	2345.42 (39.78)	230.10 (3.98)	14.10 (0.28)	15.50 (0.27)	144508.00 (58.82)	144507.70 (58.88)
	1000	500	0.96 (0.02)	1.44 (0.03)	1.61 (0.03)	0.48 (0.01)	11.60 (0.27)	12.50 (0.27)	598.20 (5.46)	598.20 (5.46)
		1000	10.76 (0.28)	11.00 (0.27)	12.49 (0.22)	2.19 (0.04)	11.20 (0.29)	11.70 (0.21)	1340.00 (4.94)	1340.00 (4.94)
		2500	92.85 (2.22)	235.54 (5.80)	269.49 (3.32)	29.01 (0.32)	13.90 (0.31)	13.70 (0.15)	4471.70 (18.16)	4471.70 (18.16)
		5000	369.05 (7.76)	1829.25 (38.16)	2121.88 (58.87)	209.48 (6.02)	13.80 (0.29)	14.10 (0.41)	12557.50 (21.90)	12557.60 (21.94)
	2000	500	1.09 (0.01)	1.60 (0.02)	1.77 (0.02)	0.49 (0.01)	11.90 (0.18)	12.80 (0.20)	506.50 (0.50)	506.50 (0.50)
		1000	10.62 (0.25)	11.97 (0.20)	12.90 (0.32)	2.18 (0.05)	11.70 (0.21)	11.60 (0.31)	1011.00 (1.02)	1011.00 (1.02)
		2500	187.80 (5.47)	239.21 (6.88)	257.65 (4.20)	28.62 (0.47)	14.50 (0.43)	13.50 (0.22)	2517.50 (1.56)	2517.50 (1.56)
		5000	680.01 (9.11)	1705.53 (23.24)	2078.65 (26.84)	209.39 (2.66)	13.10 (0.18)	14.10 (0.18)	5045.90 (2.74)	5045.90 (2.74)

Table 4: Summary for scale-free network with $\lambda^* = 0.3$

λ^*	n	p	Computation time (sec.)				Iteration		$\lvert\hat{E}\rvert$	
			CD-BLAS	CD-NAIVE	PCD-CPU	PCD-GPU	CD	PCD	CD	PCD
0.3	500	500	1.03	1.06	1.14	0.37	8.80	9.00	364.70	364.70
			(0.02)	(0.02)	(0.02)	(0.00)	(0.13)	(0.15)	(1.69)	(1.69)
		1000	4.43	9.06	10.14	1.80	9.40	9.60	713.30	713.30
			(0.08)	(0.16)	(0.22)	(0.04)	(0.16)	(0.22)	(2.13)	(2.13)
		2500	34.08	176.28	208.13	22.30	10.50	10.50	1755.40	1755.40
			(0.78)	(3.97)	(3.55)	(0.46)	(0.27)	(0.22)	(5.31)	(5.31)
		5000	138.34	1343.89	1568.08	157.45	10.40	10.60	3569.40	3569.40
			(2.87)	(27.90)	(43.39)	(4.52)	(0.22)	(0.31)	(6.12)	(6.12)
	1000	500	0.77	1.15	1.23	0.37	9.00	9.30	367.80	367.80
			(0.00)	(0.00)	(0.02)	(0.00)	(0.00)	(0.15)	(1.50)	(1.50)
		1000	8.65	8.94	9.75	1.71	9.00	9.00	715.00	715.00
			(0.15)	(0.14)	(0.16)	(0.03)	(0.15)	(0.15)	(2.09)	(2.09)
		2500	68.93	176.52	198.19	21.84	10.50	10.30	1758.80	1758.80
			(1.09)	(2.82)	(3.00)	(0.32)	(0.17)	(0.15)	(2.63)	(2.63)
		5000	267.18	1323.10	1530.87	150.17	9.90	10.10	3582.60	3582.60
			(2.69)	(13.37)	(15.77)	(1.48)	(0.10)	(0.10)	(3.96)	(3.96)
	2000	500	0.87	1.26	1.33	0.38	9.00	9.10	367.30	367.30
			(0.00)	(0.00)	(0.01)	(0.00)	(0.00)	(0.10)	(1.24)	(1.24)
		1000	8.21	9.53	10.43	1.75	9.10	9.20	712.70	712.70
			(0.07)	(0.09)	(0.14)	(0.02)	(0.10)	(0.13)	(1.73)	(1.73)
		2500	134.24	173.16	199.99	22.10	10.40	10.40	1760.00	1760.00
			(2.07)	(2.64)	(3.10)	(0.34)	(0.16)	(0.16)	(3.50)	(3.50)
		5000	513.17	1294.45	1482.85	148.71	9.90	10.00	3585.10	3585.10
			(5.14)	(13.08)	(1.64)	(0.00)	(0.10)	(0.00)	(4.13)	(4.13)

Summary

- We proposed the efficient parallel coordinate descent algorithm for CONCORD that simultaneously updates $p_{even}/2$ off-diagonal elements, which is $p/2$ for an even p and $(p-1)/2$ for an odd p.
- We also showed that $p_{even}/2$ is the maximum of the number of simultaneously updatable elements in the CONCORD-CD algorithm by applying the theoretical results in the edge-coloring.
- Comprehensive numerical studies show that the proposed CONCORD-PCD algorithm is adequate for GPU-parallel computation and more efficient than the original CONCORD-CD algorithm for large datasets.

References.

- Cai, T. T., Liu, W., and Luo, X. (2011). A constrained l1 minimization approach to sparse precision matrix estimation. *Journal of the American Statistical Association*, **106**(494), 594-607.
- Cai, T. T., Liu, W., and Zhou, H. H. (2016). Estimating sparse precision matrix: Optimal rates of convergence and adaptive estimation. *The Annals of Statistics*, **44**(2), 455-488.
- Friedman, J., Hastie, T., and Tibshirani, R. (2008). Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*, **9** (3), 432-441.
- Khare, K., Oh, S.-Y. and Rajaratnam, B. (2015). A convex pseudolikelihood framework for high dimensional partial correlation estimation with convergence guarantees. *Journal of the Royal Statistical Society:Series B*, **77**, 803-825.
- Mazumder, R. and Hastie, T. (2012). The graphical lasso: New insights and alternatives. *Electronic Journal of Statistics*, **6**(August), 2125-2149.
- Meinshausen, N. and Bühlmann, P. (2006). High-dimensional graphs and variable selection with the Lasso. *Annals of Statistics*, **34**(3), 1436-1462.
- Peng, J., Wang, P., Zhou, N., and Zhu, J. (2009). Partial correlation estimation by joint sparse regression models. *Journal of the American Statistical Association*, **104**(486), 735-746.
- Sun, T. and Zhang, C. H. (2013). Sparse matrix inversion with scaled lasso. *Journal of Machine Learning Research*, **14**, 3385-3418.
- Witten, D. M., Friedman, J. H., and Simon, N. (2011). New insights and faster computations for the graphical lasso. *Journal of Computational and Graphical Statistics*, **20**(4), 892-900.
- Yuan, M., and Lin, Y. (2007). Model Selection and Estimation in the Gaussian Graphical Model. *Biometrika*, **94**(1), 19-35.

Cyclic Reduction for $\|\Omega^{(k)} - \hat{\Omega}\|_\infty$

- Cyclic reduction

  Let $\theta = (\theta_1, \ldots, \theta_m) = \mathrm{vech}(\Omega)$, which is a half-vectorization for the parameter Ω, and $\mathbf{d} = (d_j)_{1 \le j \le m} = \theta^{(k)} - \hat{\theta}$.

  - Initialization:

    for $q = z - 1$,

    $$d_j \leftarrow \begin{cases} \max(|d_j|, |d_{j+2^q}|) & \text{if } j + 2^q \le m \\ d_j & \text{if } j + 2^q > m \text{ for } j = 1, \ldots, 2^q, \end{cases}$$

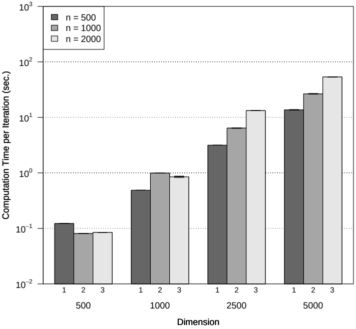
    where $z = \lceil \log_2(m) \rceil$, where $\lceil x \rceil$ is the smallest integer greater than or equal to x.

  - Parallel Update:

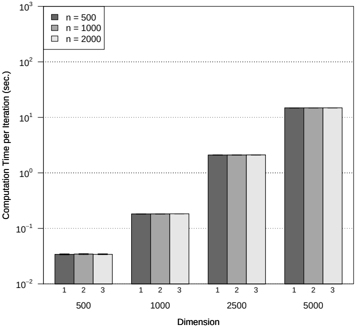
    for $q = z - 2, \ldots, 0$,

    $$d_j \leftarrow \max(|d_j|, |d_{j+2^q}|) \text{ for } j = 1, \ldots, 2^q,$$

Computation times per iteration for AR(2)

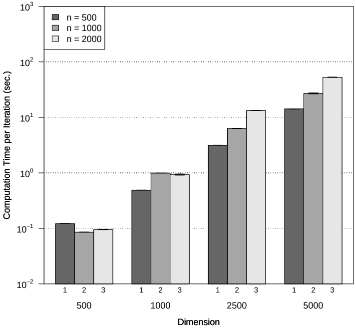


(a) CD-BLAS, $\lambda^* = 0.1$

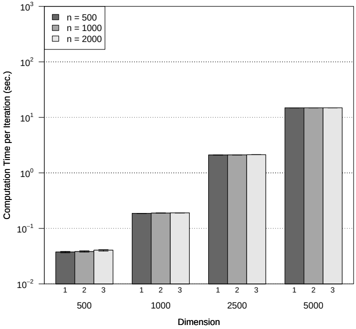


(b) PCD-GPU, $\lambda^* = 0.1$

Computation times per iteration for AR(2)

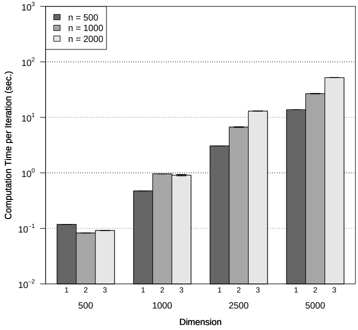


(c) CD-BLAS, $\lambda^* = 0.3$

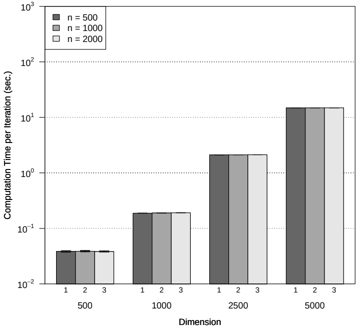


(d) PCD-GPU, $\lambda^* = 0.3$

Computation times per iteration for Scale-free

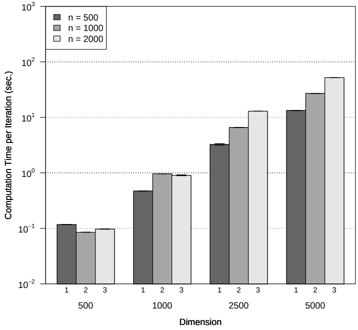


(a) CD-BLAS, $\lambda^* = 0.1$

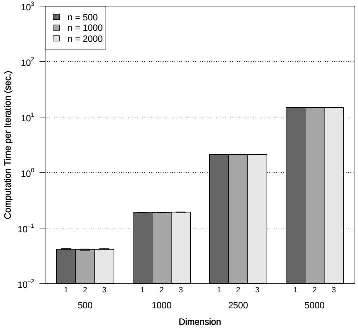


(b) PCD-GPU, $\lambda^* = 0.1$

Computation times per iteration for Scale-free



(c) CD-BLAS, $\lambda^* = 0.3$



(d) PCD-GPU, $\lambda^* = 0.3$