Communication-Avoiding Optimization Methods for Distributed Massive-Scale Sparse Inverse Covariance Estimation

Penporn Koanantakool[1,2,*] **Alnur Ali**[3] **Ariful Azad**[2] **Aydın Buluç**[2,1]
Dmitriy Morozov[2,5] **Leonid Oliker**[2] **Katherine Yelick**[1,2] **Sang-Yun Oh**[4,2]
[1]Department of Electrical Engineering and Computer Sciences, UC Berkeley
[2]Computational Research Division, Lawrence Berkeley National Laboratory
[3]Machine Learning Department, Carnegie Mellon University
[4]Department of Statistics and Applied Probability, UC Santa Barbara
[5]Berkeley Institute for Data Science, UC Berkeley

Abstract

Across a variety of scientific disciplines, sparse inverse covariance estimation is a popular tool for capturing the underlying dependency relationships in multivariate data. Unfortunately, most estimators are not scalable enough to handle the sizes of modern high-dimensional data sets (often on the order of terabytes), and assume Gaussian samples. To address these deficiencies, we introduce HP-CONCORD, a highly scalable optimization method for estimating a sparse inverse covariance matrix based on a regularized *pseudolikelihood* framework, without assuming Gaussianity. Our parallel proximal gradient method uses a novel *communication-avoiding* linear algebra algorithm and runs across a multi-node cluster with up to 1k nodes (24k cores), achieving parallel scalability on problems with up to ≈819 billion parameters (1.28 million dimensions); even on a single node, HP-CONCORD demonstrates scalability, outperforming a state-of-the-art method. We also use HP-CONCORD to estimate the underlying dependency structure of the brain from fMRI data, and use the result to identify functional regions automatically. The results show good agreement with a clustering from the neuroscience literature.

[*] Now at Google Brain.

Proceedings of the 21^{st} International Conference on Artificial Intelligence and Statistics (AISTATS) 2018, Lanzarote, Spain. PMLR: Volume 84.

1 INTRODUCTION

Suppose we observe n samples of a p-dimensional random vector $(X_1, \ldots, X_p)$, drawn independently and identically from an unknown distribution, which we assume without loss of generality has mean zero and covariance matrix $\Sigma^0 \in \mathbf{S}^p_{++}$, where $\mathbf{S}^p_{++}$ denotes the space of $(p \times p)$-dimensional positive definite matrices. In this paper, we are interested in obtaining a sparse estimate of the underlying inverse covariance matrix $\Omega^0 = (\Sigma^0)^{-1}$ associated with the p random variables, even in a high-dimensional setup, where it may be the case that $p \gg n$. This problem, known as sparse inverse covariance estimation, is an important one in statistics, and plays a role in a wide variety of real-world applications, including finance (*e.g.*, [30, 28, 45, 23, 3]), biology (*e.g.*, [19, 23, 34]), and sustainability (*e.g.*, [46, 3, 4]).

There are at least two reasons for the popularity of sparse inverse covariance estimation. First, it is well-known that the sparsity pattern of the underlying inverse covariance matrix Ω^0 gives rise to a *partial correlation graph* associated with the random variables $X_1, \ldots, X_p$; thus, an estimate of Ω^0 can reveal the statistical relationships between the variables. To be more concrete: we construct an undirected graph, where the vertices correspond to the variables $X_1, \ldots, X_p$, and we put an edge between two vertices if and only if their (estimated) partial correlation coefficient is nonzero. Under the assumption that the underlying data-generating distribution is multivariate normal, the partial correlation graph is precisely the conditional independence graph associated with the variables [27, 26, 7]. It also turns out that many downstream applications readily use an estimate of the inverse covariance matrix, not just its sparsity pattern.

The literature on sparse inverse covariance estima-

tion is vast, so we cannot possibly give a complete coverage of it here; some influential papers include [32, 48, 19, 36, 9]. On the computational side, a lot of the recent work has looked at developing scalable algorithms, specifically for a shared memory environment. However, in a "massive-scale" setting, where p, n are so large that using a single machine/node is infeasible, it may be more suitable to consider a distributed memory approach; *e.g.*, functional magnetic resonance imaging (fMRI) data sets easily run into the hundreds of gigabytes and require fitting billions of parameters. In these cases, the literature is somewhat lacking, with a few exceptions that we return to later [42, 22].

With this motivation in mind, we propose a new, highly scalable parallel proximal gradient method, for obtaining a sparse estimate of the inverse covariance matrix, in shared and/or distributed memory settings. The method, called HP-CONCORD ("HP" stands for "high-performance"), builds on the recently introduced CONCORD [23, 34] and PseudoNet [3] estimators, and explicitly minimizes the communication costs between nodes by leveraging ideas from the literature on *communication-avoiding algorithms* [14]. Highlighting some of our findings: on a single node, HP-CONCORD is about an order of magnitude faster at fitting ≈800 million parameters than BigQUIC, a well-known method for scalable sparse inverse covariance estimation [21], and also demonstrates good scalability on a cluster with 1,024 nodes, where it is able to fit ≈819 billion parameters in ≈17 minutes.

Here is an outline for the rest of the paper. In the next section, we give background on the CONCORD and PseudoNet estimators, as well as communication-avoiding algorithms, required to understand our method; in Section 3, we describe our method, HP-CONCORD. In Section 4, we evaluate HP-CONCORD on high-dimensional synthetic data, comparing it to BigQUIC. In Section 5, we present an in-depth empirical study, where we use HP-CONCORD to estimate the underlying partial correlation structure of the brain from high-dimensional fMRI data, requiring fitting ≈4 billion parameters. We wrap-up in Sec. 6.

2 BACKGROUND

CONCORD. Recent work [23] proposed the CONCORD estimator; CONCORD is a *pseudolikelihood*-based [10] estimator of the inverse covariance matrix, meaning (roughly) that it obtains an estimate by solving a sequence of lasso-like problems, rather than explicitly minimizing an ℓ_1-penalized Gaussian likelihood, making CONCORD suitable for situations where the underlying distribution is suspected to be non-Gaussian. Along these lines, CONCORD outperformed a number of strong competitors, including the graphical lasso [19], on several real-world data sets [23, 34, 3]. CONCORD also enjoys favorable estimation error and support recovery guarantees [23, 3].

PseudoNet. Follow-up work [3] proposed the PseudoNet estimator, which generalizes CONCORD, and attains much better statistical and empirical performance. The PseudoNet estimate is defined as the solution to the convex optimization problem,

$$\underset{\Omega \in \mathbf{R}^{p\times p}}{\text{minimize}} \ -\log\det(\Omega_D^2) + \mathbf{tr}(\Omega S \Omega) + \lambda_1 \|\Omega_X\|_1 + \frac{\lambda_2}{2}\|\Omega\|_F^2, \tag{1}$$

where $\Omega_D, \Omega_X \in \mathbf{R}^{p\times p}$ denote the matrices containing just the diagonal and off-diagonal entries of Ω, respectively; $S = \frac{1}{n}X^T X \in \mathbf{S}_+^p$ is the sample covariance matrix; $X \in \mathbf{R}^{n\times p}$ is the observation matrix; $\lambda_1, \lambda_2 \geq 0$ are tuning parameters; and $\|\cdot\|_1, \|\cdot\|_F$ denote the elementwise ℓ_1- and Frobenius norms, respectively. As far as the criterions are concerned, the only difference between PseudoNet and CONCORD is the presence of the squared Frobenius norm penalty, *i.e.*, setting $\lambda_2 = 0$ recovers the CONCORD criterion, analogous to the relationship between the elastic net [49] and the lasso. Thus, to keep things simple, we use the names CONCORD and PseudoNet interchangeably.

As the criterion (1) is the sum of smooth and nonsmooth convex functions, optimizing (1) with a proximal gradient method [35] is natural. Applying a proximal gradient method to (1), as in [3], yields Algorithm 1. A comment on notation: we use $\mathcal{S}_\alpha(Z)$ to denote elementwise soft-thresholding operator (*i.e.*, the proximal operator of the ℓ_1-norm) at $Z \in \mathbf{R}^{p\times p}$ with $\alpha > 0$,

$$[\mathcal{S}_\alpha(Z)]_{ij} = \begin{cases} Z_{ij} - \alpha, & Z_{ij} > \alpha \\ Z_{ij} + \alpha, & Z_{ij} < -\alpha \\ 0, & \text{otherwise} \end{cases}, \ i,j = 1,\ldots,p. \tag{2}$$

Key bottlenecks. Algorithm 1 has some key computational bottlenecks that are especially problematic in a high-dimensional setting. First and foremost, assuming dense matrices, computing the matrix product $\Omega^{(k)}S$, on each proximal gradient and line search iteration, costs $O(p^3)$. Computing the covariance matrix S has the one-time upfront cost of $O(p^2 n)$. Finally, transposing $\Omega^{(k)}S$ swaps $O(p^2)$ entries each iteration.

Thus, despite CONCORD's many favorable statistical properties, these bottlenecks make scaling CONCORD to massive data sets challenging. For example, Algorithm 1 can reconstruct the underlying gene-gene associations in a breast cancer data set, where $p \approx 4$k, in just ≈10 minutes, but it quickly becomes extremely slow or even intractable when analyzing the complete

Algorithm 1 Proximal gradient method for computing the CONCORD/PseudoNet estimate [3].

Input: data matrix $X \in \mathbf{R}^{n \times p}$; tuning parameters $\lambda_1, \lambda_2 \geq 0$; optimization tolerance $\epsilon > 0$
Output: estimate $\hat{\Omega} \in \mathbf{R}^{p \times p}$ of the underlying inverse covariance matrix Ω^0

1: Set the initial point to the identity matrix: $\Omega^{(0)} \leftarrow I$
2: Compute $S \leftarrow \frac{1}{n} X^T X$ ▷ Compute the sample covariance matrix (once)
3: **for** $k = 0, 1, 2, \ldots$ ▷ k denotes the iteration counter
4: $\quad G^{(k)} \leftarrow -(\Omega_D^{(k)})^{-1} + \frac{1}{2}(S\Omega^{(k)} + \Omega^{(k)}S) + \lambda_2 \Omega^{(k)}$ ▷ Compute the gradient, $G^{(k)}$, of the smooth part of (1)
5: $\quad g(\Omega^{(k)}) \leftarrow -\log\det((\Omega_D^{(k)})^2) + \mathbf{tr}(\Omega^{(k)} S \Omega^{(k)}) + \frac{\lambda_2}{2}\|\Omega^{(k)}\|_F^2$ ▷ Evaluate the smooth part of (1), used below
6: $\quad$ **for** $\tau = 1, \frac{1}{2}, \frac{1}{4}, \ldots$ ▷ Choose the step size τ via backtracking line search
7: $\quad\quad \Omega^{(k+1)} \leftarrow \mathcal{S}_{\tau\lambda_1}(\Omega^{(k)} - \tau G^{(k)})$ ▷ Apply the elementwise soft-thresholding operator $\mathcal{S}_{\tau\lambda_1}$; see (2)
8: $\quad\quad g(\Omega^{(k+1)}) \leftarrow -\log\det((\Omega_D^{(k+1)})^2) + \mathbf{tr}(\Omega^{(k+1)} S \Omega^{(k+1)}) + \frac{\lambda_2}{2}\|\Omega^{(k+1)}\|_F^2$ ▷ Evaluate the smooth part of (1)
9: $\quad$ **until** $g(\Omega^{(k+1)}) \leq g(\Omega^{(k)}) - \mathbf{tr}((\Omega^{(k)} - \Omega^{(k+1)})^T G^{(k)}) + \frac{1}{2\tau}\|\Omega^{(k)} - \Omega^{(k+1)}\|_F^2$
10: **until** a stopping criterion is satisfied, using ϵ
11: **return** the estimate $\hat{\Omega} \leftarrow \Omega^{(k)}$

gene expression data set, where $p \approx 30$k [34]. Furthermore, the running time required to compute the CONCORD estimates across a grid of tuning parameters, as in resampling methods such as cross-validation, the bootstrap, and stability selection [33, 29], would be prohibitive. To address these computational deficiencies, we build on the recent work coming out of the literature on communication-avoiding algorithms.

Communication-avoiding algorithms. Communication often dominates the overall cost of a distributed algorithm [5, 16] and therefore should be minimized. In scientific computing, communication-avoiding algorithms (see, *e.g.*, [8, 39, 17]) significantly reduce communication by (i) choosing a suitable *data layout*, *i.e.*, sensibly distributing data across nodes; (ii) *replicating*, *i.e.*, duplicating data across nodes to cut down on communication at the expense of space; and (iii) computing a quantity of interest, given the layout and replication, in a communication-efficient way. Speedups ranging from 2-100× have been reported in the literature [13, 24, 25, 38, 44, 6]; in fact, communication-avoiding algorithms decrease communication asymptotically in the amount of replication. However, communication-avoiding algorithms have gone relatively unnoticed in statistics, until very recently [47, 15, 40]. Our approach is motivated by the recent "2.5D" [39] and "1.5D" [24] communication-avoiding matrix multiplication algorithms.

Related work. Wang et al. [43] present a distributed memory approach to sparse inverse covariance estimation, based on the alternating direction method of multipliers [12], without optimizing the amount of communication; it would be interesting to combine their method with a communication-avoiding algorithm. GINCO [22] is a distributed greedy algorithm, which is not guaranteed to converge to a globally optimal point. We compare to BigQUIC [21], a well-known Gaussian likelihood method.

3 HP-CONCORD

HP-CONCORD resolves the key computational bottlenecks in Algorithm 1, by distributing the data (*e.g.*, X), any intermediate variables (*e.g.*, the iterates $\Omega^{(k)}$), and any computations across a network of nodes/processors. Some of the computations may then be done in an embarrassingly parallel way; for the others, we turn to a communication-avoiding approach.

We begin by observing that $\Omega^{(k)}S$, critical to most of the calculations in Algorithm 1, can be computed in two ways, with different computation and communication costs. The first approach, which we call "Cov", explicitly computes $S = \frac{1}{n}X^T X$, using S to then compute $\Omega^{(k)}S$. The second approach, which we call "Obs", never explicitly computes S, opting instead to compute $Y^{(k)} = \frac{1}{n}\Omega^{(k)}X^T$ and $Y^{(k)}X = \Omega^{(k)}S$. (The fact that Cov computes the covariance matrix S, while Obs never does, now explains their names.) Below, we broadly describe how Cov and Obs parallelize CONCORD's bottlenecks; then we discuss the details.

Parallelizing CONCORD's key bottlenecks. *The Cov variant.* As mentioned, Cov proceeds by computing $S = \frac{1}{n}X^T X$ (line 2 in Algorithm 1), *i.e.*, the product of two dense matrices, upfront. Cov then uses S to compute $W^{(k)} = \Omega^{(k)}S$, a sparse-dense product, on every line search iteration. The transpose $(W^{(k)})^T$ is formed on every proximal gradient iteration. All the matrix products can be computed using the communication-avoiding algorithm for dense-dense and sparse-dense matrix multiplication that we present below, while the transpose can be computed via partial all-to-all communication. Using $W^{(k)}$, Cov then computes the rest of the gradient $G^{(k)}$ (line 4) in an embarrassingly parallel way, as inverting a diagonal matrix (equivalent to inverting the entries on the diagonal) and applying the soft-thresholding operator $\mathcal{S}_{\tau\lambda_1}$ are just simple elementwise operations. As for the line

Algorithm 2 The Cov variant of HP-CONCORD, for computing a sparse estimate of the inverse covariance matrix.

Input: data matrix $X \in \mathbf{R}^{n \times p}$; tuning parameters $\lambda_1, \lambda_2 \geq 0$; optimization tolerance $\epsilon > 0$
Output: estimate $\hat{\Omega} \in \mathbf{R}^{p \times p}$ of the underlying inverse covariance matrix Ω^0

1: $\Omega^{(0)} \leftarrow I$
2: Compute $S \leftarrow \frac{1}{n} X^T X$ ▷ Compute (once) via a distributed dense-dense matrix multiplication
3: Compute $W^{(0)} \leftarrow \Omega^{(0)} S$ ▷ Compute via a distributed sparse-dense matrix multiplication
4: **for** $k = 0, 1, 2, \ldots$
5: Form $(W^{(k)})^T$ ▷ Form via a distributed matrix transpose
6: $G^{(k)} \leftarrow -(\Omega_D^{(k)})^{-1} + \frac{1}{2}((W^{(k)})^T + W^{(k)}) + \lambda_2 \Omega^{(k)}$ ▷ Use $W^{(k)}, (W^{(k)})^T$
7: $g(\Omega^{(k)}) \leftarrow -2 \sum_i \log(\Omega_{ii}^{(k)}) + \mathbf{tr}(W^{(k)} \Omega^{(k)}) + \frac{\lambda_2}{2} \|\Omega^{(k)}\|_F^2$ ▷ Use $(W^{(k)})^T$; see text for details
8: **for** $\tau = 1, \frac{1}{2}, \frac{1}{4}, \ldots$
9: $\Omega^{(k+1)} \leftarrow \mathcal{S}_{\tau\lambda_1}(\Omega^{(k)} - \tau G^{(k)})$ ▷ Apply the soft-thresholding operator, $\mathcal{S}_{\tau\lambda_1}$, in a distributed manner
10: Compute $W^{(k+1)} \leftarrow \Omega^{(k+1)} S$ ▷ Compute via a distributed sparse-dense matrix multiplication
11: $g(\Omega^{(k+1)}) \leftarrow -2 \sum_i \log(\Omega_{ii}^{(k+1)}) + \mathbf{tr}(W^{(k+1)} \Omega^{(k+1)}) + \frac{\lambda_2}{2} \|\Omega^{(k+1)}\|_F^2$ ▷ See text for details
12: **until** $g(\Omega^{(k+1)}) \leq g(\Omega^{(k)}) - \mathbf{tr}((\Omega^{(k)} - \Omega^{(k+1)})^T G^{(k)}) + \frac{1}{2\tau} \|\Omega^{(k)} - \Omega^{(k+1)}\|_F^2$ ▷ See text for details
13: **until** a stopping criterion is satisfied, using ϵ
14: **return** the estimate $\hat{\Omega} \leftarrow \Omega^{(k)}$

search (lines 6–9): since (i) $\log\det(A) = \sum_i \log(A_{ii})$, (ii) $\mathbf{tr}(BC) = \sum_{i,j} B_{ij} C_{ij}$ and (iii) $\|B\|_F^2 = \mathbf{tr}(B^2)$, for a diagonal matrix A and symmetric matrices B, C, these may be done in an embarrassingly parallel way.

The Obs variant. Obs never explicitly computes S; rather, it proceeds by computing $Y^{(k)} = \frac{1}{n} \Omega^{(k)} X^T$ on every proximal gradient and line search iteration via a communication-avoiding algorithm, as with Cov. The rest of the gradient $G^{(k)}$ is then computed by forming $Z^{(k)} = Y^{(k)} X$ as well as $(Z^{(k)})^T$, and using the same embarrassingly parallel elementwise operations as before. Noticing $\mathbf{tr}(\Omega^{(k)} S \Omega^{(k)}) = \frac{1}{n} \mathbf{tr}(\Omega^{(k)} X^T X \Omega^{(k)}) = \frac{1}{n} \|\Omega^{(k)} X^T\|_F^2 = \frac{1}{n} \|Y^{(k)}\|_F^2$, the line search (except for forming $Y^{(k)}$) also consists of elementwise operations.

We present a high-level description of the Cov variant of HP-CONCORD in Algorithm 2, highlighting the main differences vs. Algorithm 1 in blue. The pseudocode for Obs, shown in the supplement, is similar.

Matrix layouts and multiplication algorithms. Parallel matrix multiplication is a well-studied problem, but most existing work considers dense matrices, and there is room for improvement in special cases. The most popular algorithm uses a 2D layout [2, 41], treating the processors as a square grid and making each processor responsible for all computations associated with one submatrix of the output matrix. 3D [1] and 2.5D [39] algorithms are provably communication-optimal in certain cases, and instead divide up the 3D iteration space, by essentially making c copies of the output matrix, and having a square group of processors responsible for a subset of updates to that copy; the copies are eventually summed to produce the final answer. As we discuss next, these are not always the fastest methods in our setting, due to the matrix sizes and sparsity levels that arise with Cov and Obs.

Cov. For the sparse-dense product $W^{(k)} = \Omega^{(k)} S$, shifting around only $\Omega^{(k)}$ can use much less bandwidth, and could outperform the classic 2D/2.5D/3D algorithms by up to two orders of magnitude [24]. Therefore, we put $\Omega^{(k)}$ in 1D block row and S in 1D block column layout. As for $S = \frac{1}{n} X^T X$: when $p > n$, X^T is tall and X is wide, so partitioning X in a 2D layout, as 2D/2.5D/3D algorithms do, would result in tall and short local matrices, which perform poorly on local memory hierarchies. Instead, we group processors into teams of c members each, and arrange the teams as a 1D array, distributing the rows of X^T (a 1D block row layout) and the cols. of X (a 1D block col. layout).

Obs. The dense-dense product $Z^{(k)} = Y^{(k)} X$ is similar to $X^T X$. For the sparse-dense product $Y^{(k)} = \Omega^{(k)} X^T$, we proceed just as for $W^{(k)} = \Omega^{(k)} S$ with Cov, putting $Y^{(k)}, \Omega, X^T$ all in a 1D block row layout.

Figure 1 illustrates all the distributed operations.

Replication. Here, we extend the 1.5D algorithm [24] to support different replication factors for each matrix operand. We use our algorithm, detailed in Algorithm 3, to compute all the matrix products arising with Cov and Obs. To compute the product $C = AB$, our algorithm rotates either A or B around (call this matrix R), fixing the other (call this F), and leaving C stationary. Let P be the number of homogeneous processors, let c_R and c_F be the replication factors of R and F, and let G_R and G_F be the logical grids of processors of sizes $P/c_R \times c_R$ and $P/c_F \times c_F$, respectively. We partition R equally, in 1D, into P/c_R parts, and let processors $G_R(i, :)$ have the ith part. Similarly, we partition F and C equally, in 1D, into P/c_F parts, letting processors $G_F(j, :)$ have the jth part. The processors $G_F(j, :)$ now work together as a team to compute the jth part of C, with each member

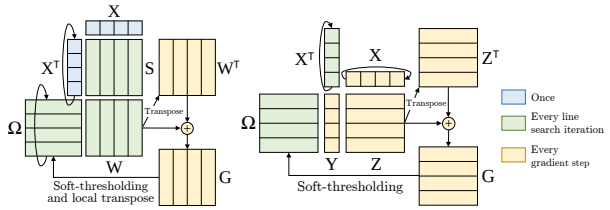

Figure 1: Left: Cov first computes $S = \frac{1}{n}X^TX$ by shifting X^T; then, on every iteration k, Cov computes $W^{(k)} = \Omega^{(k)}S$ by shifting $\Omega^{(k)}$, globally transposes $W^{(k)}$, computes $G^{(k)}$ from $W^{(k)}$ and $(W^{(k)})^T$, soft-thresholds $G^{(k)}$ to get $\Omega^{(k)}$, and converts $\Omega^{(k)}$ back to 1D block row layout by doing a local matrix transpose. Right: Obs computes $Y^{(k)} = \frac{1}{n}\Omega^{(k)}X^T$ by shifting X^T, computes $Z^{(k)} = Y^{(k)}X$ by shifting X, globally transposes $Z^{(k)}$ to get $(Z^{(k)})^T$ in the same layout, computes $G^{(k)}$ from $Z^{(k)}$ and $(Z^{(k)})^T$, and then soft-thresholds $G^{(k)}$ to get the new $\Omega^{(k)}$.

multiplying the jth part of F with different parts of R. Further details are contained in the supplement.

Algorithm 3 Our 1.5D matrix multiplication algorithm.

1: **for each** $G_R(i, \ell_R) = G_F(j, \ell_F)$, in parallel, **do**
2: $\quad\delta \leftarrow \min(\ell_F, \ell_R) \cdot \max(1, c_F/c_R)$
3: $\quad$Shift R by δ
4: $\quad$**for** $\frac{p}{c_F c_R}$ rounds **do**
5: $\quad\quad$Calculate local $C = AB$
6: $\quad\quad$Shift R by c_F
7: $\quad$**end for**
8: $\quad$SumReduce/Allgather C between $G_F(j, :)$
9: **end for**

Final computation and communication costs. We model the total running time of a distributed algorithm as $T = F\gamma + L\alpha + W\beta$, where F is the total number of flops across all processors; L is the *latency cost*, *i.e.*, the total number of messages sent by all processors; and W is the *bandwidth cost*, *i.e.*, the size of all the messages sent. We define γ, α, β as machine-dependent quantities, measuring the time per flop, time to initiate a message, and time to send a *word* (*i.e.*, an atomic unit of communication), respectively.

We also make the following definitions. Let s be the total number of proximal gradient iterations for HP-CONCORD, t be the average number of line search iterations on each proximal gradient iteration, and d be the average number of nonzero entries in a row of $\Omega^{(k)}$ (averaged over all rows and all st line search iterations). Also, let P be the number of processors and $\mathbf{nnz}(C)$ denote the number of nonzeros in C.

The following lemma, proven by working through the details presented above for computing $W^{(k)} = \Omega^{(k)}S$, $S = \frac{1}{n}X^TX$, $Y^{(k)} = \Omega^{(k)}X^T$, and $Z^{(k)} = Y^{(k)}X$, presents the total dominant flop counts for Cov and Obs. Using the flop counts, the lemma also tells us when Cov is cheaper than Obs (when Cov is "worth it"). All our proofs are contained in the supplement.

Lemma 3.1. *Cov costs* $F_{Cov} = 2np^2 + 2dp^2(st + 1)$ *flops, while Obs costs* $F_{Obs} = 2np^2s + 2dnp(st + 1)$ *flops. Therefore, Cov is cheaper than Obs as long as:* $d/p < (n/(p - n)) \cdot (1/t)$.

Turning now to the communication costs, our next lemma establishes that Algorithm 3 can reduce the communication involved in transposing a matrix, helpful as transposing can be an expensive operation since it involves all-to-all communication for a 1D layout.

Lemma 3.2. *In Algorithm 3, transposing the matrix* C *requires:* $\log_2(Q)$ *messages and* $(\mathbf{nnz}(C) \cdot c_R c_F \cdot Q \log_2(Q))/P$ *words, where* $Q = \max(P/c_R^2, P/c_F^2)$.

The following lemma shows we can save a factor of $c_R c_F$ in latency and c_R in bandwidth, by using Alg. 3.

Lemma 3.3. *Algorithm 3 sends* $\frac{P}{c_R c_F}$ *messages and* $\frac{\mathbf{nnz}(R)}{c_F}$ *words, where* R *is the matrix that is rotated.*

Our final lemma presents the total communication costs for Cov and Obs, by using the preceding results and working through the communication costs of computing the various matrices.

Lemma 3.4. *Let* $Q = \max(P/c_X^2, P/c_\Omega^2)$. *Then Cov's communication costs are:*

$$L_{Cov} = \frac{P}{c_X^2} + st\frac{P}{c_X c_\Omega} + \log_2(Q),$$
$$W_{Cov} = \frac{np}{c_X} + st\frac{dp}{c_X} + p^2\frac{c_X c_\Omega}{P}Q\log_2(Q),$$

while Obs' communication costs are:

$$L_{Obs} = s(t+1)\frac{P}{c_\Omega c_X} + \log_2(Q),$$
$$W_{Obs} = s(t+1)\frac{np}{c_\Omega} + p^2\frac{c_X c_\Omega}{P}Q\log_2(Q).$$

4 NUMERICAL EXAMPLES

In this section, we evaluate the computational performance of HP-CONCORD on several synthetic data sets. We start by empirically verifying Lemma 3.1, running experiments with various values of n and checking when Cov becomes faster than Obs. Afterwards, we investigate the benefits of replication, as discussed in Section 3, by varying the tuning parameters c_X, c_Ω, controlling the amount of replication for the matrices X, Ω, respectively. Finally, we run several head-to-head timing comparisons with BigQUIC.

All our experiments in this section were run with double precision on "Edison", a 5,586-node supercomputer at the National Energy Research Scientific Computing Center, where each node consists of two 12-core

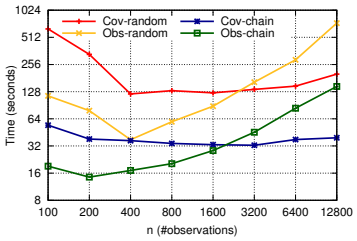

Figure 2: Runtimes in seconds for Cov (blue/red) vs. Obs (green/yellow), on synthetic data coming from chain and random graphs, with 16 nodes, $p = 40$k, and various n.

Figure 3: Runtimes in seconds for Obs, at various replication factors, on a chain graph with 256 nodes, $p = 40$k, and $n = 100$. Colder/warmer cells mean better/worse runtimes. The worst runtime is due to the non-communication-avoiding configuration, circled in blue; the best is circled in red, with a 5× speedup due to replication.

Method		$p = 10$k	$p = 20$k	$p = 40$k	$p = 80$k
HP-CONCORD	PPV	**99.75**	**99.92**	**99.94**	**99.94**
	FDR	**0.25**	**0.08**	**0.06**	**0.06**
BigQUIC	PPV	99.48	99.71	99.78	99.81
	FDR	0.52	0.29	0.22	0.19

Table 1: The positive predictive values (PPVs) and false discovery rates (FDRs) for HP-CONCORD vs. BigQUIC, relative to the sparsity pattern of Ω^0, for various p. Best (*i.e.*, highest PPVs and lowest FDRs) in bold.

2.4 GHz Intel Xeon E5-2695 processors with 64 GB of DDR3 RAM. Our code was written in C++ using MPI and OpenMP. We used 2 MPI processes per node. For the results in Section 5, we used the "Eos" supercomputer at Oak Ridge Leadership Computing Facility.

When does Cov become worth it? We consider both banded and random strictly diagonally dominant Ω^0's, corresponding to chain and random graphs, respectively, and sample Gaussian data. For each setup, we fix $p = 40{,}000$, vary $n \in \{100, 200, \ldots, 12{,}800\}$, and plot in Figure 2 the runtimes (on 16 nodes) required for Cov and Obs to generate estimates attaining the same average degree as the underlying graph (2 for the chain graph, 60 for the random graph). It is clear from the figure that Obs' runtime grows linearly with n, while Cov's does not, consistent with Lemma 3.1, as the dominant term in Obs' cost depends linearly on n and Cov's does not. The trend reverses when we fix n and increase p, *i.e.*, Obs becomes asymptotically more appealing than Cov. Interestingly, the crossover point where Cov becomes faster than Obs happens later than Lemma 3.1 predicts, since most of Cov's cost comes from sparse-dense matrix multiplications, which have higher time per flop than the dense-dense matrix multiplications that dominate Obs' overall cost (*i.e.*, $\gamma_{\text{sparse-dense}} \gg \gamma_{\text{dense-dense}}$).

Benefits of replication. To illustrate the benefits of replication, we run Obs on all possible replication configurations (*i.e.*, all c_X, c_Ω combinations) with 256 nodes on a chain graph, where $p = 40$k, $n = 100$, plotting all the runtimes in seconds in Figure 3. The figure shows both extremes: (a) when $c_X = c_\Omega = 1$, Obs is in a purely non-communication-avoiding mode, partitioning all the matrices into P equal blocks, and takes the longest to run; and (b) when $c_X = 512$ and $c_\Omega = 1$, every processor maintains the entire X, does all matrix multiplications locally, and only communicates when replicating X and during the transpose. The best result comes when $c_X = 8$ and $c_\Omega = 16$, a 5× speedup over the non-communication-avoiding configuration.

Comparison with BigQUIC. Finally, we compare HP-CONCORD with BigQUIC, on chain and random graphs. To put the two algorithms on an equal footing, we choose the tuning parameters so that the estimates are equally sparse. For the chain graphs, we fix $n = 100$ and vary p from 10,000 to 1.28 million. As d is not much smaller than n, we use the Obs variant of HP-CONCORD and vary the number of nodes, reporting the best runtime across a range of replication levels. We present the results in the left panel of Figure 4, and interpolate BigQUIC's result for $p = 1{,}280{,}000$ as it took longer than four days to converge. From the figure, we see HP-CONCORD matching BigQUIC in a shared memory setup (*i.e.*, one node), and demonstrating good scalability as more nodes are added, which the user may customize to fit their needs; *e.g.*, when $p = 80{,}000$, an estimate may be computed in less than four seconds with 1,024 nodes.

For the random graphs, we vary p from 10,000 to 320,000. In the middle panel of Figure 4, we fix $n = 100$, and again use Obs, as d here is even larger than it was for the chain graphs. In the right panel, we set $n = p/4$, and use Cov because n is large. In both figures, we see Obs outperforming BigQUIC by about an order of magnitude, and exhibiting even better scalability as more nodes are added than with the chain graphs, since Ω^0 is comparatively denser here. Lastly, in Table 1, we present the best positive predictive values and false discovery rates for both methods, computed by looking at the differences between the estimated and true sparsity patterns, across a range of tuning parameters. At all problem sizes, our method demonstrates better graph recovery than BigQUIC.

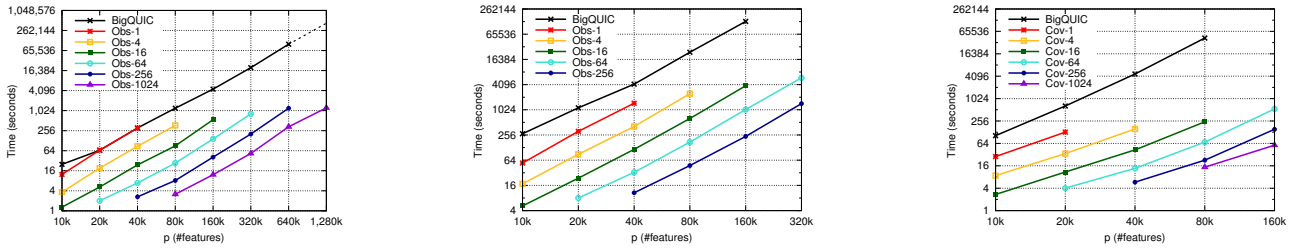

Figure 4: Runtimes in seconds for HP-CONCORD vs. BigQUIC. Left: chain graph with $n = 100$. Middle: random graph with $n = 100$. Right: random graph with $n = p/4$. "Obs-x" or "Cov-x" denotes Obs or Cov run with x nodes.

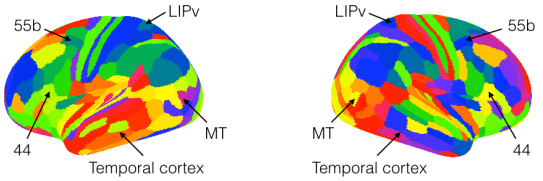

Figure 5: The clusterings from Glasser et al., for the left and right hemispheres of the brain; the colors only serve to differentiate the different clusters and carry no additional meaning. The clusterings rely on multiple exogenous data sources and a significant amount of domain knowledge.

5 CASE STUDY: GRAPH ESTIMATION FROM fMRI DATA

We now present a case study, where we use HP-CONCORD to make progress on a challenging and important problem in neuroscience: obtaining a biologically meaningful clustering of the brain (more specifically, points on the cerebral cortex), from high-dimensional fMRI data. The data we use, from the Human Connectome Project [37], is a $(91,282 \times 91,282)$-dimensional sample covariance matrix, roughly 60 GB in size, and requires fitting ≈4 billion parameters. The large number of dimensions rules out most methods for sparse inverse covariance estimation, but makes HP-CONCORD a natural choice. Below, we qualitatively and quantitatively compare the clusterings generated by HP-CONCORD to those from Glasser et al. [20], a state-of-the-art clustering from the neuroscience literature, presented in Figure 5. Previewing our findings, we see that the entirely data-driven clusterings generated by HP-CONCORD are able to capture many of the important features also present in Glasser et al.

Approach. We pursue a two-step approach, where for step (i) we generate a partial correlation graph using HP-CONCORD, and for step (ii) we apply a graph-based clustering algorithm to the partial correlation graph arising from the sparsity pattern of the HP-CONCORD estimate. For step (i), we consider all combinations of the tuning parameters $\lambda_1 \in \{0.48, 0.5, 0.52, 0.54, 0.57, 0.59, 0.61, 0.64, 0.67, 0.69, 0.72\} \times \lambda_2 \in \{0.10, 0.13, 0.16, 0.2, 0.25, 0.31, 0.39, 0.49\}$; tuning parameters outside these ranges yielded either trivially sparse or dense estimates. Running HP-CONCORD on a single (λ_1, λ_2) pair took ≈37 minutes. For step (ii), the clustering algorithms we consider are: the well-known Louvain method [11], as well as a relatively new clustering method from the persistent homology literature [18] that leverages the degree matrix associated with the partial correlation graph (details in the supplement). Additionally, because the clusterings from Glasser et al. treat the left and right hemispheres of the brain separately, we run and evaluate our clustering algorithms on the subgraphs for the left and right hemispheres separately.

Evaluation. As mentioned, our main points of comparison are the recent clusterings, for the left and right hemispheres, from Glasser et al., presented in Figure 5. However, we also consider a simple baseline, given by discarding {99, 99.1, ..., 99.8, 99.9, 99.91, ..., 99.98, 99.99}% of the sample covariance matrix entries: *i.e.*, keep entries with the largest magnitudes (c.f. [31]) in order to generate marginal correlation graphs. This baseline lets us probe the comparative advantage of using marginal vs. partial correlations. To quantitatively compare clusterings, we consider a variant of the Jaccard score; details are in the supplement.

Results. The top and middle rows of Table 2 present the best clusterings generated by the Cov variant of HP-CONCORD, followed by the persistent homology and Louvain methods, respectively, when compared to those of Glasser et al. presented in Figure 5, according to the (modified) Jaccard score. The bottom row presents the best clusterings generated by thresholding the sample covariance matrix at various levels. The left and middle columns present the results for the left and right hemispheres, respectively. We see that the persistent homology clusterings perform the best, in terms of Jaccard score, across both hemispheres.

Qualitatively, the persistent homology clusterings are able to identify several clusters of interest to the neuroscience community (c.f. Figure 3 in [20]); this is certainly encouraging, since we do not expect perfect re-

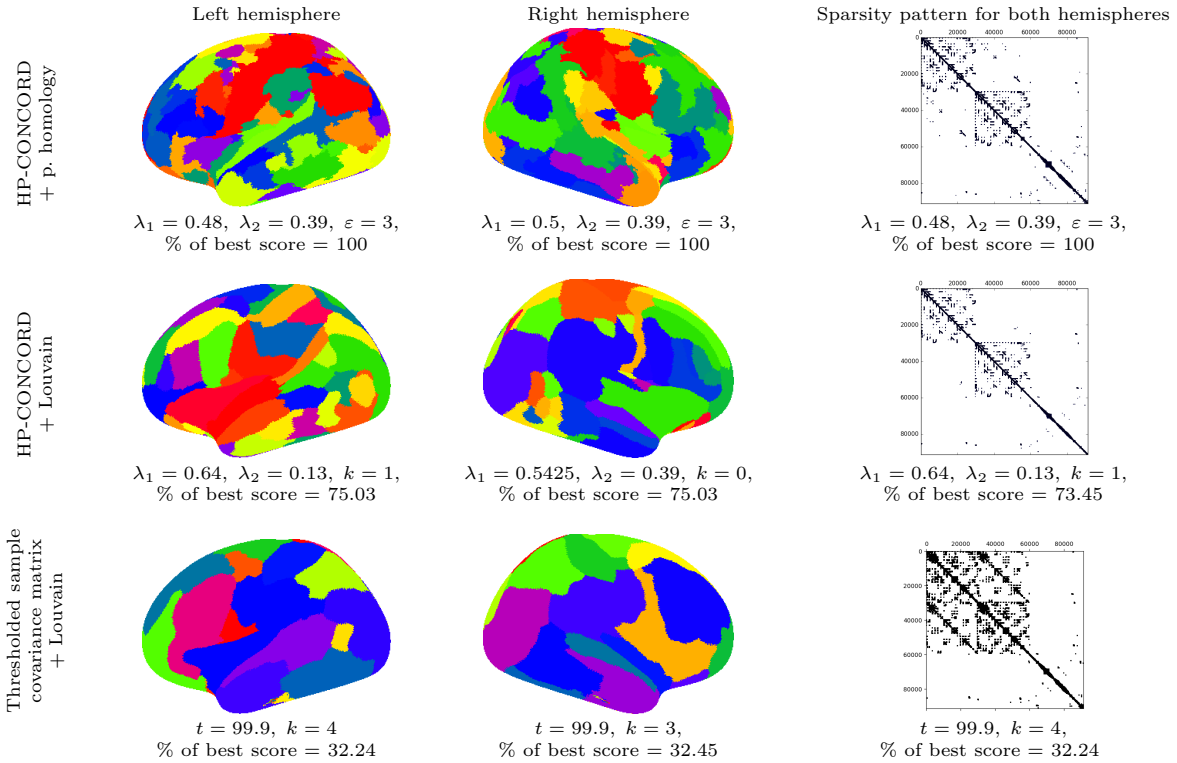

Table 2: The left and middle columns present the best clusterings for the left and right hemispheres, respectively, according to the (modified) Jaccard score, relative to those of Glasser et al.; the colors only serve to differentiate the different clusters. The right-most column presents the sparsity patterns for both hemispheres yielding the best clustering for the left hemisphere; black indicates a nonzero entry. The rows correspond to the various methods. The scores under the figures are the percentages of the best Jaccard score attained; higher is better. Since the persistent homology clusterings perform the best, these percentages are just 100. Also indicated are the tuning parameter values $(\lambda_1, \lambda_2, \varepsilon, k, t)$ yielding the best clusterings. The Jaccard scores, tuning parameter details, and an expanded set of results are in the supplement.

covery of all the clusters in Figure 5, as the latter clusters rely on multiple exogenous data sources and a significant amount of domain knowledge. Some examples: the persistent homology clusterings seem to pick out area 55b, involved in hearing; the lateral intraparietal cortex (LIPv), involved in eye movement; and much of the variation in the temporal cortex, involved in processing information from the senses. On the other hand, the Louvain method and the clusterings generated by the sample covariance matrix seem to miss these clusters, as they appear overly smooth. All the methods seem to miss Brodmann's area 44, involved in hearing and speaking, and the middle temporal visual area (MT), involved in seeing moving objects.

Lastly, it is also interesting to analyze the sparsity patterns of the HP-CONCORD estimates; details are contained in the supplement, due to space constraints.

6 CONCLUSION

We presented HP-CONCORD, a communication-avoiding distributed proximal gradient method for estimating a sparse inverse covariance matrix from "massive-scale" data. HP-CONCORD can fit ≈819 billion parameters ($p = 1.28$ million) in ≈17 minutes in a distributed memory setting, and is an order of magnitude faster than BigQUIC when fitting ≈800 million parameters ($p = 40$k) in a shared memory setting. For future work, it may be interesting to apply a divide-and-conquer strategy based on a block structure assumption, to further optimize the computation. We also used HP-CONCORD to capture the functional connectivity structure of the cerebral cortex as a partial correlation graph, which, in turn, was clustered into distinct regions; some regions showed good agreement with a result from the neuroscience literature.

Acknowledgements. This work was supported in part by: the Department of Energy's Office of Science Advanced Scientific Computing Research (DOE/SC/ASCR) X-Stack and Applied Mathematics programs at LBNL (DEAC02-05CH11231) and at UC Berkeley (UCB) (DE-SC0008700); the Department of Defense; the Gordon and Betty Moore Foundation at UCB (GBMF3834); the Alfred P. Sloan Foundation at UCB (2013-10-27); a DOE Computational Science Graduate Fellowship (DE-FG02-97ER25308); and by UCSB, including a Faculty Research Grant. It used resources supported by DOE/SC/ASCR at the National Energy Research Scientific Computing Center (DE-AC02-05CH11231) and the Oak Ridge Leadership Computing Facility (DE-AC05-00OR22725). The U.S. Government (USG) retains, and the publisher, by accepting the article for publication, acknowledges, that the USG retains a non-exclusive, paid-up, irrevocable, world-wide license to publish or reproduce the published form of this manuscript, or allow others to do so, for USG purposes. We thank Ryan J. Tibshirani and J. Zico Kolter for the helpful feedback.

References

[1] R. C. Agarwal, S. M. Balle, F. G. Gustavson, M. Joshi, and P. Palkar. A three-dimensional approach to parallel matrix multiplication. *IBM Journal of Research and Development*, 39(5):575–582, 1995.

[2] A. Aggarwal, A. K. Chandra, and M. Snir. Communication complexity of PRAMs. *Theoretical Computer Science*, 71(1):3–28, 1990.

[3] A. Ali, K. Khare, S.-Y. Oh, and B. Rajaratnam. Generalized pseudolikelihood methods for inverse covariance estimation. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, 2017.

[4] A. Ali, J. Z. Kolter, and R. J. Tibshirani. The multiple quantile graphical model. In *Advances in Neural Information Processing Systems 29*, 2016.

[5] S. Ashby, P. Beckman, J. Chen, P. Colella, B. Collins, D. Crawford, J. Dongarra, D. Kothe, R. Lusk, P. Messina, et al. The opportunities and challenges of exascale computing. *Summary Report of the Advanced Scientific Computing Advisory Committee Subcommittee*, pages 1–77, 2010.

[6] A. Azad, G. Ballard, A. Buluç, J. Demmel, L. Grigori, O. Schwartz, S. Toledo, and S. Williams. Exploiting multiple levels of parallelism in sparse matrix-matrix multiplication. *SIAM Journal on Scientific Computing*, 38(6):C624–C651, 2016.

[7] K. Baba, R. Shibata, and M. Sibuya. Partial correlation and conditional correlation as measures of conditional independence. *Australian and New Zealand Journal of Statistics*, 46(4):657–664, 2004.

[8] G. Ballard, E. Carson, J. Demmel, M. Hoemmen, N. Knight, and O. Schwartz. Communication lower bounds and optimal algorithms for numerical linear algebra. *Acta Numerica*, 23:1155, 2014.

[9] O. Banerjee, L. El Ghaoui, and A. d'Aspremont. Model selection through sparse maximum likelihood estimation for multivariate Gaussian or binary data. *Journal of Machine Learning Research*, 9:485–516, 2008.

[10] J. Besag. Spatial interaction and the statistical analysis of lattice systems. *Journal of the Royal Statistical Society: Series B*, 36(2):192–236, 1974.

[11] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(10):P10008, 2008.

[12] S. Boyd, N. Parikh, E. Chu, B. Peleato, J. Eckstein, et al. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning*, 3(1):1–122, 2011.

[13] J. Demmel, D. Eliahu, A. Fox, S. Kamil, B. Lipshitz, O. Schwartz, and O. Spillinger. Communication-optimal parallel recursive rectangular matrix multiplication. In *Parallel & Distributed Processing (IPDPS), 2013 IEEE 27th International Symposium on*, pages 261–272. IEEE, 2013.

[14] J. Demmel, M. Hoemmen, M. Mohiyuddin, and K. Yelick. Avoiding communication in sparse matrix computations. In *Proceedings of the 22nd IEEE International Parallel and Distributed Processing Symposium*, 2008.

[15] A. Devarakonda, K. Fountoulakis, J. Demmel, and M. W. Mahoney. Avoiding communication in primal and dual block coordinate descent methods. *arXiv:1612.04003*, 2016.

[16] J. Dongarra, J. Hittinger, J. Bell, L. Chacon, R. Falgout, M. Heroux, P. Hovland, E. Ng, C. Webster, and S. Wild. Applied mathematics research for exascale computing. Technical report, Lawrence Livermore National Laboratory, 2014.

[17] M. Driscoll, E. Georganas, P. Koanantakool, E. Solomonik, and K. Yelick. A communication-optimal N-body algorithm for direct interactions. In *Proceedings of the 27th IEEE International Parallel and Distributed Processing Symposium*, 2013.

[18] H. Edelsbrunner and D. Morozov. Persistent homology: theory and applications. In *Proceedings of the 6th European Congress of Mathematics*, 2012.

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

[20] M. F. Glasser, T. S. Coalson, E. C. Robinson, C. D. Hacker, J. Harwell, E. Yacoub, K. Ugurbil, J. Andersson, C. F. Beckmann, M. Jenkinson, S. M. Smith, and D. C. Van Essen. A multimodal parcellation of human cerebral cortex. *Nature*, 536(7615):171–178, 2016.

[21] C.-J. Hsieh, M. A. Sustik, I. S. Dhillon, P. K. Ravikumar, and R. Poldrack. Big & QUIC: sparse

inverse covariance estimation for a million variables. In *Advances in Neural Information Processing Systems 26*, pages 3165–3173, 2013.

[22] P. Kambadur and A. Lozano. A parallel, block greedy method for sparse inverse covariance estimation for ultra-high dimensions. In *Proceedings of the 16th International Conference on Artificial Intelligence and Statistics*, 2013.

[23] K. Khare, S.-Y. Oh, and B. Rajaratnam. A convex pseudolikelihood framework for high-dimensional partial correlation estimation with convergence guarantees. *Journal of the Royal Statistical Society: Series B*, 77(4):803–825, 2015.

[24] P. Koanantakool, A. Azad, A. Buluç, D. Morozov, S. Y. Oh, L. Oliker, and K. Yelick. Communication-avoiding parallel sparse-dense matrix-matrix multiplication. In *Proceedings of the 30th IEEE International Parallel and Distributed Processing Symposium*, 2016.

[25] P. Koanantakool and K. Yelick. A computation- and communication-optimal parallel direct 3-body algorithm. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage, and Analysis*, 2014.

[26] S. Lauritzen. *Graphical Models*. Oxford University Press, 1996.

[27] A. J. Lawrance. On conditional and partial correlation. *The American Statistician*, 30(3):146, 1976.

[28] O. Ledoit and M. Wolf. Honey, I shrunk the sample covariance matrix. *UPF Economics and Business Working Paper*, (691), 2003.

[29] C. Lim and B. Yu. Estimation stability with cross-validation (ESCV). *Journal of Computational and Graphical Statistics*, 25(2):464–492, 2016.

[30] H. Markowitz. Portfolio selection. *Journal of Finance*, 7(1):77–91, 1952.

[31] R. Mazumder and T. Hastie. Exact covariance thresholding into connected components for large-scale graphical lasso. *Journal of Machine Learning Research*, 13:781–794, 2012.

[32] N. Meinshausen and P. Bühlmann. High-dimensional graphs and variable selection with the lasso. *The Annals of Statistics*, 34(3):1436–1462, 2006.

[33] N. Meinshausen and P. Bühlmann. Stability selection. *Journal of the Royal Statistical Society: Series B*, 72(4):417–473, 2010.

[34] S.-Y. Oh, O. Dalal, K. Khare, and B. Rajaratnam. Optimization methods for sparse pseudolikelihood graphical model selection. In *Advances in Neural Information Processing Systems 27*. 2014.

[35] N. Parikh and S. Boyd. Proximal algorithms. *Foundations and Trends in Optimization*, 1(3):123–231, 2013.

[36] A. Rothman, P. Bickel, E. Levina, and J. Zhu. Sparse permutation invariant covariance estimation. *Electronic Journal of Statistics*, 2:494–515, 2008.

[37] S. M. Smith, C. F. Beckmann, J. Andersson, E. J. Auerbach, J. Bijsterbosch, G. Douaud, E. Duff, D. A. Feinberg, L. Griffanti, M. P. Harms, et al. Resting-state fMRI in the human connectome project. *NeuroImage*, 80:144–168, 2013.

[38] E. Solomonik, A. Buluç, and J. Demmel. Minimizing communication in all-pairs shortest paths. In *Proceedings of the 27th IEEE International Parallel and Distributed Processing Symposium*, 2013.

[39] E. Solomonik and J. Demmel. Communication-optimal parallel 2.5d matrix multiplication and LU factorization algorithms. In *Proceedings of the 17th International Conference on Parallel Processing*. 2011.

[40] S. Soori, A. Devarakonda, J. Demmel, M. Gurbuzbalaban, and M. M. Dehnavi. Avoiding communication in proximal methods for convex optimization problems. *arXiv:1710.08883*, 2017.

[41] R. van de Geijn and J. Watts. SUMMA: scalable universal matrix multiplication algorithm. *Concurrency and Computation: Practice and Experience*, 9(4):255–274, 1997.

[42] H. Wang, A. Banerjee, C.-J. Hsieh, P. K. Ravikumar, and I. S. Dhillon. Large-scale distributed sparse precision estimation. In *Advances in Neural Information Processing Systems 26*, pages 584–592, 2013.

[43] H. Wang, A. Banerjee, C.-J. Hsieh, P. K. Ravikumar, and I. S. Dhillon. Large-scale distributed sparse precision estimation. In *Advances in Neural Information Processing Systems 26*, 2013.

[44] S. Williams, M. Lijewski, A. Almgren, B. V. Straalen, E. Carson, N. Knight, and J. Demmel. *s*-step Krylov subspace methods as bottom solvers for geometric multigrid. In *Proceedings of the 28th IEEE International Parallel and Distributed Processing Symposium*, pages 1149–1158, May 2014.

[45] J. Won, J. Lim, S. Kim, and B. Rajaratnam. Condition number-regularized covariance estimation. *Journal of the Royal Statistical Society: Series B*, 75(3):427–450, 2013.

[46] M. Wytock and J. Z. Kolter. Sparse Gaussian conditional random fields: algorithms, theory, and application to energy forecasting. In *Proceedings of the 30th International Conference on Machine Learning*, 2013.

[47] Y. You, J. Demmel, K. Czechowski, L. Song, and R. Vuduc. CA-SVM: communication-avoiding support vector machines on distributed systems. In *Proceedings of the 29th IEEE International Parallel and Distributed Processing Symposium*, 2015.

[48] M. Yuan and Y. Lin. Model selection and estimation in the Gaussian graphical model. *Biometrika*, 94(1):19–35, 2007.

[49] H. Zou and T. Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B*, 67:301–320, 2005.