3 The SINCO Method

3.1 Relation to Prior Art

Problem (2) is a special case of a semidefinite programming problem (SDP) [20], which can be solved in polynomial time by interior point methods (IPM). However, each iteration requires $O(p^6)$ time and $O(p^4)$ space, which is very costly. Another reason why using IPMs is less desirable for the structure recovery problem is that the sparsity pattern is recovered only in the limit, i.e., the solution does not typically include exact zeros, and thus numerical inaccuracy can potentially interfere with the structure recovery.

As an alternative to IPMs, several more efficient approaches were developed recently for problem (2). Most of those approaches are primarily focused on solving the dual problem in (3). For example, [1] and [7] apply the block-coordinate descent method to the dual formulation, [9] uses a first-order optimal gradient ascent approach, and [5] uses a projected gradient approach.

Herein, we propose a novel algorithm, called SINCO, for Sparse INverse COvariance problem. SINCO solves the primal problem directly and uses coordinate ascent, which naturally preserves the sparsity of the solution. Unlike, for example, COVSEL and *glasso* that optimize one row (and the corresponding symmetric column) of the dual matrix at each step, SINCO only optimizes one diagonal or two (symmetric) off-diagonal entries of the matrix $C$ at each step. The advantage of our approach is that the solution to each subproblem is available in closed form as a root of a quadratic equation. Computation at each step requires a constant number of arithmetic operations, independent of $p$. Hence, in $O(p^2)$ operations a potential step can be computed for all pairs of symmetric elements (i.e., for all pairs $(i,j)$). Then the step which provides the best function value improvement can be chosen, which is the essence of the greedy nature of our approach. Once the step is taken, the update of the gradient information requires $O(p^2)$ operations. Hence, overall, each iteration takes $O(p^2)$ operations. As we will see later, each step is also suitable for massive parallelization.

In comparison, *glasso* and COVSEL require solving a quadratic programming problem when updating a row (column)$^3$, and its theoretical and empirical complexity varies depending on the method used, but always exceeds $O(p^2)$: it is $O(p^4)$ for COVSEL and $O(p^3)$ for *glasso*. Also, the methods of [14] and [5] iterate through the columns and require $O(p^4)$ and $O(p^3)$ time per iteration, respectively (see [5] for detailed discussion). Note, however, that the overall number of iterations can be potentially lower than in the case of SINCO, since the above methods update each row (column) at once. We will mainly focus on comparing our method with *glasso* as a representative state-of-the-art technique, particularly since it is the only other method that maintains the initial sparsity of the solution in a controlled manner. (In some cases, when we only compare the accuracy of the solution (Section 4.4), we perform experiments with COVSEL, a similar but less efficient implementation of the same approach as *glasso*).

$^3$ COVSEL solves the subproblems via an interior point approach (as second order cone quadratic problems (SOCP)), while *glasso* poses the subproblem as a dual of the Lasso problem [17], which is solved by coordinate descent method.
As we will show in our numerical experiments, SINCO, in a serial mode, is comparable to or faster than glasso, which is orders of magnitude faster than COVSEL [7]. Also, SINCO often reaches lower false-positive error than glasso since it introduces nonzero elements greedily. Perhaps the most interesting consequence of SINCO’s greedy nature is that it reproduces the regularization path behavior while using only one value of the regularization parameter $\lambda$ (see Section 4.1). Another important feature of SINCO is the ability to efficiently utilize warm starts in various modes. For instance, it is easy to compute a range of solutions for various values of $\lambda$, which defines matrix $S$.

### 3.2 Algorithm Description

The main idea of the method is the following: at each iteration, the matrix $C'$ or the matrix $C''$ is updated by changing one element on the diagonal or two symmetric off-diagonal elements. This implies that the updated $C$ can be written at $C + \theta(e_i e_j^T + e_j e_i^T)$, where $i$ and $j$ are the indices corresponding to the elements that are being changed. We can therefore rewrite the objective function of the problem (2) as a function of $\theta$ (denoted $f(\theta)$ below). The key observation is that, given the matrix $W = C^{-1}$, the exact line search that optimizes $f(\theta)$ along the direction $e_i e_j^T + e_j e_i^T$ reduces to a solution of a quadratic equation. Hence each such line search takes a constant number of operations. Moreover, given the starting objective value, the new function value on each step can be computed in a constant number of steps. This means that we can perform such line search for all $(i, j)$ pairs in $O(p^2)$ time, which is linear in the number of unknown variables $C_{ij}$. We then can choose the step that gives the best improvement in the value of the objective function. After the step is chosen, the dual matrix $W = C^{-1}$ and, hence, the objective function gradient, are updated in $O(p^2)$ operations$^4$.

We now present the method. First, we can reformulate the problem (2) as:

$$\max_{C', C''} \frac{n}{2} [\ln \det(C' - C'') - \text{tr}(A(C' - C''))] - ||C' - C''||_S,$$

s.t. $C' \geq 0$, $C'' \geq 0$, $C' - C'' > 0$

Note that $||C' - C''||_S = \text{tr}(S(C' + C''))$ if $C'$ and $C''$ have non-overlapping nonzero structure.

For a fixed pair $(i, j)$, we consider the update of $C'$ given by $C'(\theta) = C' + \theta(e_i e_j^T + e_j e_i^T)$, such that $C' \geq 0$. Then we can write the objective as the function of $\theta$:

$$f'(\theta) = \frac{n}{2} [\ln \det(C + \theta e_i e_j^T + \theta e_j e_i^T) - \text{tr}(A(C + \theta e_i e_j^T + \theta e_j e_i^T))] - ||C + \theta e_i e_j^T + \theta e_j e_i^T||_S$$

Similarly, if we consider the update of the form $C''(\theta) = C'' + \theta(e_i e_j^T + e_j e_i^T)$ such that $C'' > 0$, the objective function becomes

$$f''(\theta) = \frac{n}{2} [\ln \det(C - \theta e_i e_j^T - \theta e_j e_i^T) - \text{tr}(A(C - \theta e_i e_j^T - \theta e_j e_i^T))] - ||C - \theta e_i e_j^T - \theta e_j e_i^T||_S$$

$^4$ Note that there is no need to enforce the posdef constraint explicitly, as $\ln \det(C)$ goes to negative infinity when $C$ approaches singularity. At each step, we maximize the objective along the direction of increase until the local maximum is reached. Hence, it is impossible for the method to move past the point where the objective is negative infinity.