



Research



Cite this article: Peixoto TP. 2025 Scalable network reconstruction in subquadratic time. *Proc. R. Soc. A* **481**: 20250345.
<https://doi.org/10.1098/rspa.2025.0345>

Received: 18 April 2025

Accepted: 22 September 2025

Subject Areas:

statistical physics, complexity

Keywords:

network reconstruction, statistical inference, complex networks

Author for correspondence:

Tiago P. Peixoto

e-mail: tiago.peixoto@it-u.at

Scalable network reconstruction in subquadratic time

Tiago P. Peixoto^{1,2}

¹Inverse Complexity Lab, IT:U Interdisciplinary Transformation University, Linz 4040, Austria

²Department of Network and Data Science, Central European University, Vienna 1100, Austria

TPP, 0000-0002-4505-0517

Network reconstruction consists in determining the unobserved pairwise couplings between N nodes given only observational data on the resulting behaviour that is conditioned on those couplings—typically a time-series or independent samples from a graphical model. A major obstacle to the scalability of algorithms proposed for this problem is a seemingly unavoidable quadratic complexity of $\Omega(N^2)$, corresponding to the requirement of each possible pairwise coupling being contemplated at least once, despite the fact that most networks of interest are sparse, with a number of non-zero couplings that are only $O(N)$. Here, we present a general algorithm applicable to a broad range of reconstruction problems that significantly outperforms this quadratic baseline. Our algorithm relies on a stochastic second-neighbour search that produces the best edge candidates with high probability, thus bypassing an exhaustive quadratic search. If we rely on the conjecture that the second-neighbour search finishes in log-linear time, we demonstrate theoretically that our algorithm finishes in subquadratic time, with a data-dependent complexity loosely upper bounded by $O(N^{3/2} \log N)$, but with a more typical log-linear complexity of $O(N \log^2 N)$. In practice, we show that our algorithm achieves a performance that is many orders of magnitude faster than the quadratic baseline—in a manner consistent with our theoretical analysis—allows for easy parallelization, and thus enables the reconstruction of networks with hundreds of thousands and even millions of nodes and edges.

© 2025 The Authors. Published by the Royal Society under the terms of the Creative Commons Attribution License <http://creativecommons.org/licenses/by/4.0/>, which permits unrestricted use, provided the original author and source are credited.

1. Introduction

Networks encode the pairwise interactions that determine the dynamical behaviour of a broad class of interconnected systems. However, in many important cases, the interactions themselves are not directly observed, and instead we have access only to their indirect outcomes, usually as samples from a multivariate distribution modelled as a probabilistic graphical model [1–3], or from time-series data representing some dynamics conditioned on the network structure [4,5]. Instances of this problem include the inference of interactions between microbial species from co-occurrence data [6], financial market couplings from stock prices [7], protein structure from amino-acid contact maps [8], gene regulatory networks from expression data [9], neural connectivity from fMRI and EEG data [10] and epidemic contact tracing [11], among others.

Perhaps, the most well studied formulation of the network reconstruction problem is covariance selection [12], where it is assumed that the data consist of independent samples of a multivariate Gaussian, and the objective is to infer its precision matrix—often assumed to be sparse. The most widely employed algorithm for this purpose is the graphical LASSO (GLASSO) [13] and its many variations [14,15]. More generally, one can consider arbitrary probabilistic graphical models (also known as Markov random fields) [16], where the latent network structure encodes the conditional dependence between variables. Covariance selection is a special case of this family where the variables are conditionally normally distributed, resulting in an analytical likelihood, unlike the general case which involves intractable normalization constants. A prominent non-Gaussian graphical model is the Ising model [17], applicable for binary variables, which has a wide range of applications.

The vast majority of algorithmic approaches so far employed to the network reconstruction problem cannot escape a complexity of at least $O(N^2)$, where N is the number of nodes in the network. The original GLASSO algorithm for covariance selection has a complexity of $O(N^3)$. By exploiting properties that are specific to the covariance selection problem (and hence do not generalize to the broader reconstruction context), the faster QUIC [18] and BIGQUIC [19] approximative methods have $O(N^2)$ and $O(NE)$ complexities, respectively, with E being the number of edges (i.e. non-zero entries in the reconstructed matrix), such that the latter also becomes quadratic in the usual sparse regime with $E = O(N)$. Similarly, for the inverse Ising model [17,20] or graphical models in general [16,21] no known method can improve on a $O(N^2)$ complexity, and the same is true for reconstruction from time series [4,22]. To the best of our knowledge, no general approach exists to the network reconstruction problem with a lower complexity than $O(N^2)$, unless strong assumptions on the true network structure are made. Naively, one could expect this barrier to be a fundamental one, since for the reconstruction task—at least in the general case—we would be required to probe the existence of every possible pairwise coupling at least once.

Instead, in this work, we show that it is in fact possible to implement a general network reconstruction scheme that yields subquadratic complexity, without relying on the specific properties of any particular instance of the problem. Our approach is simple and relies on a stochastic search for the best update candidates (i.e. edges that need to be added, removed or updated) in an iterative manner that starts from a random graph and updates the candidate list by inspecting the second neighbours of this graph—an approach which leads to log-linear performance [23–25]. Furthermore, every step of our algorithm is easily parallelizable, allowing its application for problems of massive size.

This paper is organized as follows. In §2, we introduce the general reconstruction scenario, and the coordinate descent (CD) algorithm, which will function as our baseline with quadratic complexity. In §3, we describe our improvement over the baseline and analyse its algorithmic complexity. In §4, we evaluate the performance of our approach on a variety of synthetic and empirical data and, in §5, we showcase our algorithm with some selected large-scale empirical network reconstruction problems. We finalize in §6 with a conclusion.

2. General reconstruction scenario and the coordinate descent (CD) baseline

We are interested in a general reconstruction setting where we assume some data \mathbf{X} are sampled from a generative model with a likelihood

$$P(\mathbf{X}|\mathbf{W}), \quad (2.1)$$

where \mathbf{W} is a $N \times N$ symmetric matrix corresponding to the weights of an undirected graph of N nodes. In most cases of interest, the matrix \mathbf{W} is sparse, i.e. its number of non-zero entries is $O(N)$, but otherwise we assume no special structure. Typically, the data \mathbf{X} are either a $N \times M$ matrix of M independent samples, with X_{im} being a value associated with node i for sample m , such that

$$P(\mathbf{X}|\mathbf{W}) = \prod_{m=1}^M P(\mathbf{x}_m|\mathbf{W}), \quad (2.2)$$

with \mathbf{x}_m being the m th column of \mathbf{X} , or a Markovian time series with

$$P(\mathbf{X}|\mathbf{W}) = \prod_{m=1}^M P(\mathbf{x}_m|\mathbf{x}_{m-1}, \mathbf{W}), \quad (2.3)$$

given some initial state \mathbf{x}_0 . Our algorithm will not rely strictly on any such particular formulations, only on a generic posterior distribution,

$$\pi(\mathbf{W}) = P(\mathbf{W}|\mathbf{X}) = \frac{P(\mathbf{X}|\mathbf{W})P(\mathbf{W})}{P(\mathbf{X})}, \quad (2.4)$$

that needs to be computable only up to normalization. In appendix B, we list some generative models that we use as examples in this work, but for now it is simpler if we consider the generative model more abstractly. We focus on the maximum *a posteriori* point estimate

$$\hat{\mathbf{W}} = \arg \max_{\mathbf{W}} \pi(\mathbf{W}). \quad (2.5)$$

For many important instances of this problem, such as covariance selection [12,13] and the inverse Ising model [17], the above optimization objective is convex. In this case, one feasible approach is the CD algorithm [26,27], which proceeds by iterating over all variables in sequence, and solving a one-dimensional optimization (which is guaranteed to be convex as well), and stopping when a convergence threshold is reached (see algorithm 1).

Algorithm 1. Coordinate descent (CD).

Input: Objective $\pi(\mathbf{W})$, initial state \mathbf{W}_0 , convergence criterion ϵ

Output: Estimate $\hat{\mathbf{W}} = \arg \max_{\mathbf{W}} \pi(\mathbf{W})$

$\mathbf{W} \leftarrow \mathbf{W}_0$

repeat

$\Delta \leftarrow 0$

for all $i < j$ **do**

$\mathbf{W}'_{ij} \leftarrow \arg \max_{W_{ij}} \pi(\mathbf{W})$

▷ (1)

$\Delta \leftarrow \Delta + |\mathbf{W}'_{ij} - \mathbf{W}_{ij}|$

$\mathbf{W}_{ij} \leftarrow \mathbf{W}'_{ij}$

until $\Delta < \epsilon$

$\hat{\mathbf{W}} \leftarrow \mathbf{W}$

Algorithm 1 has complexity $O(\tau N^2)$; assuming step (1), corresponding to an element-wise optimization, can be done in time $O(1)$ (e.g. using bisection search),¹ where τ is the number of iterations required for convergence—which, in general, will depend on the particulars of the

¹See appendix A for a discussion on the wide class of problems where this holds.

problem and initial state, but typically we have $\tau = O(1)$. We observe that in all our analyses we assume that the initial state W_0 corresponds to an entirely empty network with every entry being equal to zero—corresponding to the worst-case scenario of maximum initial ignorance about the network to be reconstructed.

Note that the CD algorithm does not require a differentiable objective $\pi(W)$, but convergence to the global optimum is only guaranteed if it is convex and sufficiently smooth [28]. In practice, CD is the method of choice for covariance selection and the inverse Ising model, with a speed of convergence that often exceeds gradient descent (which is not even strictly applicable when non-differentiable regularization is used, such as L_1 of GLASSO [13]), since each coordinate can advance further with more independence from the remaining ones, unlike with gradient descent where all coordinates are restricted by the advance of the slowest one.

For a non-convex objective $\pi(W)$, the CD algorithm will, in general, not converge to the global optimum. Nevertheless, it is a fundamental baseline that often gives good results in practice even in non-convex instances and can serve as a starting point for more advanced algorithms. In this work, we are not primarily concerned with issues due to non-convexity, but rather with a general approach that circumvents the need to update all $\binom{N}{2}$ entries of the matrix W .

Our objective is to reduce the $O(N^2)$ complexity of the CD algorithm. But, before continuing, we remark on the feasibility of the reconstruction problem, and the practical obstacle that this quadratic complexity represents. At first, one could hypothesize that the size of the data matrix X would need to be impractically large to allow for the reconstruction of networks with N in the order of hundreds of thousands or millions. In such a case, a quadratic complexity would be the least of our concerns for problem sizes that are realistically feasible. However, for graphical models, it is possible to show that the number of samples required for accurate reconstruction scales only with $M = O(\log N)$ [16,21,29–31], meaning that reconstruction of large networks with relatively little information is possible. In view of this, a quadratic algorithmic complexity on N poses a significant obstacle for practical instances of the problem, which could easily become more limited by the runtime of the algorithm than the available data.

3. Subquadratic network reconstruction

As mentioned in the introduction, our approach consists of implementing a stochastic second-neighbour search to find the entries of the matrix W that yield the largest improvement of the objective function. We proceed in three steps, we define: (i) a greedy extension of the CD algorithm 1 that updates only the best entries per iteration; (ii) a function that finds the best entries to be updated according to an iterated search for the k nearest neighbours (KNNs) of each node; and (iii) a function that solves the KNN problem in subquadratic time using a stochastic second-neighbour search.

In the greedy extension of the CD algorithm 1 (GCD), we select only the κN entries of the matrix W that would individually lead to the steepest increase of the objective function $\pi(W)$, as summarized in algorithm 2.

In this algorithm, the function $\text{FINDBEST}(m, \mathcal{S}, D)$ finds the set of m pairs (i, j) of elements in set \mathcal{S} with the smallest ‘distance’ $D(i, j)$ (which in our case corresponds to $-\max_{W_{ij}} \pi(W)$). Clearly, for $\lfloor \kappa N \rfloor > 0$, if the original CD algorithm converges to the global optimum, so will the GCD algorithm. The function FINDBEST solves what is known as the m closest pairs problem [32]. In that literature, $D(i, j)$ is often assumed to be a metric, typically Euclidean, which allows the problem to be solved in log-linear time, usually by means of spacial sorting. However, this class of solution is not applicable to our case, since we cannot expect that our distance function will, in general, define a metric space. An exhaustive solution of this problem consists in probing all $\binom{|\mathcal{S}|}{2}$ pairs, which would yield no improvement on the quadratic complexity of CD. Instead, we proceed by first solving the m closest pairs problem with an algorithm proposed by [33] that maps it to a recursive KNN problem. The algorithm proceeds by setting $k = \lceil 4m/|\mathcal{S}| \rceil$ and finding for each element i in set \mathcal{S} the KNNs j with smallest $D(i, j)$. From this set of directed pairs, we select the $2m$ best ones to compose the set \mathcal{D}^+ , and construct a set \mathcal{D} with the undirected version of the

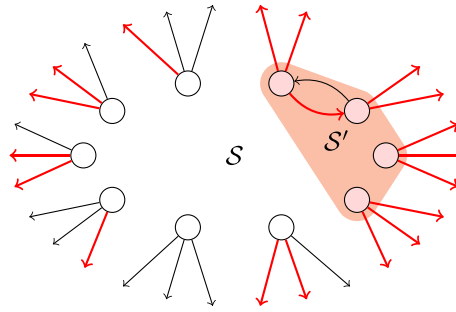


Figure 1. Diagrammatic representation of the sets S and S' in algorithm 3 for $k = 3$. Edges marked in red belong to set \mathcal{D}^+ , i.e. the $2m$ directed pairs (i, j) with smallest $\mathcal{D}(i, j)$. Note that reciprocal edges need not both belong to \mathcal{D}^+ , despite $\mathcal{D}(i, j)$ being symmetric, since ties are broken arbitrarily. The nodes in red have all out-edges (nearest neighbours) in set \mathcal{D} , and hence are assigned to set S' . Since the set of m best pairs could still contain undiscovered pairs of elements in S' , the search needs to continue recursively for members of this set.

Algorithm 2. Greedy coordinate descent (GCD).

Input: Objective $\pi(\mathbf{W})$, greediness factor κ , initial state \mathbf{W}_0 , convergence criterion ϵ

Output: Estimate $\hat{\mathbf{W}} = \arg \max_{\mathbf{W}} \pi(\mathbf{W})$

$\mathbf{W} \leftarrow \mathbf{W}_0$

repeat

$\Delta \leftarrow 0$

$\mathcal{E}_{\text{best}} \leftarrow \text{FINDBEST}(\lfloor \kappa N \rfloor, \{1, \dots, N\}, \mathcal{D})$

$\triangleright |\mathcal{E}_{\text{best}}| = \lfloor \kappa N \rfloor$

for all $(i, j) \in \mathcal{E}_{\text{best}}$ **do**

$W'_{ij} \leftarrow \arg \max_{W_{ij}} \pi(\mathbf{W})$

$\Delta \leftarrow \Delta + |W'_{ij} - W_{ij}|$

$W_{ij} \leftarrow W'_{ij}$

until $\Delta < \epsilon$

$\hat{\mathbf{W}} \leftarrow \mathbf{W}$

function $\mathcal{D}(i, j)$

\triangleright “Distance” function

return $-\max_{W_{ij}} \pi(\mathbf{W})$

pairs in \mathcal{D}^+ , such that $m \leq |\mathcal{D}| \leq 2m$. At this point, we can identify a subset S' of S composed of nodes for which all nearest neighbour edges belong to \mathcal{D} when their direction is discarded, as shown in figure 1. Since these nodes have been saturated, we cannot rule out that the m closest pairs will not contain undiscovered pairs of elements in set S' . Therefore, we proceed recursively for S' , and stop when $|S'|^2 \leq 4m$, in which case the node set has become small enough for an exhaustive search to be performed. This is summarized as algorithm 3, and a proof of correctness is given in [33].

As we shall discuss in a moment, the recursion depth of algorithm 3 is bounded logarithmically on $|S|$, and hence its runtime is dominated by the KNN search. Note that so far we have done nothing substantial to address the overall quadratic performance, since finding the KNNs exhaustively still requires all pairs to be probed. Similarly to the m closest pairs problem, efficient log-linear algorithms exist based on spacial sorting when the distance is Euclidean; however more general approaches do also exist. In particular, the NNDescent algorithm by Dong *et al.* [23] approximately solves the KNN problem in subquadratic time, while not requiring the distance function to be a metric. The algorithm is elegant and requires no special data structure beyond the nearest neighbour graph itself, other than a heap for each node. It works by starting with a random KNN digraph, and successively updating the list of best neighbours by inspecting the

Algorithm 3. Find the m best edge candidates.

function FINDBEST($m, \mathcal{S}, \mathcal{D}$)

```

if  $|\mathcal{S}|^2 \leq 4m$  then
    return  $\{m \text{ pairs } (i, j) \text{ of nodes in set } \mathcal{S} \text{ with smallest } \mathcal{D}(i, j) \text{ found by exhaustive search.}\}$ 
     $\triangleright O(|\mathcal{S}|^2)$ 
 $k \leftarrow \lceil 4m/|\mathcal{S}| \rceil$ 
 $\mathcal{G} \leftarrow \text{FindKNN}(k, \mathcal{S}, \mathcal{D})$ 
 $\triangleright k\text{-nearest neighbour digraph}$ 
 $\mathcal{D}^+ \leftarrow 2m \text{ directed edges } (i, j) \in \mathcal{G} \text{ with smallest } \mathcal{D}(i, j).$ 
 $\mathcal{D} \leftarrow \text{unique undirected pairs } (i, j) \text{ in } \mathcal{D}^+.$ 
 $\triangleright m \leq |\mathcal{D}| \leq 2m$ 
 $\mathcal{S}' \leftarrow \{i \in \mathcal{S} \mid (i, j) \in \mathcal{D}, \forall \text{ out-neighbour } j \text{ of } i \text{ in } \mathcal{G}\}$ 
 $\mathcal{D}' \leftarrow \text{FINDBEST}(m, \mathcal{S}', \mathcal{D})$ 
return  $\{m \text{ pairs } (i, j) \text{ in } \mathcal{D} \cup \mathcal{D}' \text{ with smallest } \mathcal{D}(i, j)\}$ 

```

Algorithm 4. Find KNNs by NNDescent.

Input: Convergence criterion ε
function FINDKNN($k, \mathcal{V}, \mathcal{D}$)

```

 $\mathcal{G} \leftarrow \text{directed graph with node set } \mathcal{V} \text{ and } k \text{ out-neighbours chosen uniformly at random independently}$ 
for all nodes.
repeat
     $\Delta \leftarrow 0$ 
     $\mathcal{G}' \leftarrow \mathcal{G}$ 
     $\mathcal{U} \leftarrow \text{undirected version of } \mathcal{G}$ 
    for all  $i \in \mathcal{V}$  do
        for all  $j$  incident on  $i$  in  $\mathcal{U}$  do
            for all  $v$  incident on  $j$  in  $\mathcal{U}$  do
                if  $v = i$  or  $(i, v) \in \mathcal{G}'$  then
                    continue
                 $\hat{u} \leftarrow \arg \max_u \{\mathcal{D}(i, u) \mid (i, u) \in \mathcal{G}'\}$ 
 $\triangleright (1)$ 
                if  $\mathcal{D}(i, v) < \mathcal{D}(i, \hat{u})$  then
 $\triangleright (2)$ 
                    Replace  $(i, \hat{u})$  with  $(i, v)$  in  $\mathcal{G}'$ 
                     $\Delta \leftarrow \Delta + 1$ 
             $\mathcal{G} \leftarrow \mathcal{G}'$ 
until  $\Delta/(k|\mathcal{V}|) < \varepsilon$ 
return  $\mathcal{G}$ 

```

neighbours of the neighbours in the undirected version of the KNN digraph, as summarized in algorithm 4. The main intuition behind this approach is that if (i, j) and (j, v) are good entries to update, then (i, v) is likely to be a good candidate as well—even if triangle inequality is not actually obeyed.

Steps (1) and (2) in the algorithm can be both performed in time $O(1)$ by keeping a (e.g. Fibonacci) heap containing the nearest k neighbours for each node. Thus, each full iteration of algorithm 4 runs in time $O(k^2N)$, if the degrees of nodes in \mathcal{U} are all $O(k)$, otherwise it would run in time $O(\langle q \rangle N)$ where $\langle q \rangle$ is the average number of second neighbours in \mathcal{U} . In case $\langle q \rangle \gg k^2$, the inner loop is then modified to run over only the first k neighbours for each node, to preserve the $O(k^2N)$ complexity, regardless of any structure that \mathcal{U} may have. Although this algorithm has seen wide deployment, in particular as part of the popular UMAP dimensionality reduction method [34], and has had its performance empirically ‘battle tested’ in a variety of practical workloads, it has so far resisted a formal analysis of its algorithmic complexity. To date, the most careful analyses of this algorithm observe that the number of iterations required for convergence does not exceed $2\lceil \log_{2k} N \rceil$ in empirical settings [25]. The intuitive reasoning is that the initial random graph has as a diameter of approximately $\lceil \log_{2k} N \rceil$, and hence twice this is the number

of steps needed for each node to communicate its neighbourhood to all other nodes, and for the updated information to return.² This conjectured bound on the convergence results in an overall $O(k^2 N \log N)$ complexity. However, this conjecture can only be rigorously proven on a version of the algorithm where the second-neighbour search is replaced by a range query, and the data are generated by a homogeneous Poisson process [24]. These conditions are not applicable to our problem (and we do not assume them); however as we see in our numerical experiments, we find robust evidence that it holds to the case of network reconstruction as well. This typical log-linear complexity of the NNDescent algorithm is what finally enables us to escape the quadratic complexity of the CD algorithm, as we demonstrate shortly.

Importantly, the NNDescent algorithm is approximative, as it does not guarantee that all nearest neighbours are always correctly identified. Although in practice it often yields very good recall rates [23], its inexact nature is not a concern for our purposes, since it does not affect the correctness of our GCD algorithm: if an element of the best set $\mathcal{E}_{\text{best}}$ in algorithm 2 is missed at a given iteration, it will eventually be considered in a future iteration, owing to the random initialization of algorithm 4. Our primary concern is only with the average speed with which it finds the best update candidates. Nevertheless, as we show later, inaccuracies in the solution of the KNN problem often disappear completely by the time algorithm 3 finishes (as the KNN problem considers a much larger set of pairs than what gets eventually selected), such that the recall rates for the m closest pairs problem approach the optimal values as the parameter κ is increased.

In figure 2, we show an example of the intermediate results obtained by our algorithm on simulated data on an empirical network.

In appendix C, we list some low-level optimizations that can improve the overall performance of the algorithm, including parallelization. A C++ implementation of the algorithm (with OpenMP CPU parallelism) is available as part of the `graph-tool` Python library [36].

(a) Algorithmic complexity

We now obtain the overall algorithmic complexity of our GCD algorithm. Since it consists in repeatedly finding the best κN entries in the matrix W to be updated, its overall runtime will depend crucially on the $\text{FINDBEST}(\kappa N, \mathcal{V}, D)$ function defined in algorithm 3.

Lemma 3.1. *Assuming the GCD algorithm 2 converges after τ iterations, with τ independent of N —implying that there are at most $O(N)$ non-zero entries in W —then its overall algorithmic complexity is determined solely by the $\text{FINDBEST}(\kappa N, \mathcal{V}, D)$ function.*

Proof. At each iteration, algorithm 2 spends time $T(N, m)$ on algorithm 3 to find the $m = \kappa N$ best update coordinates and time $O(\kappa N)$ to actually update them. Therefore, the algorithmic complexity per iteration will be given by $T(N, \kappa N)$, since this is bounded from below by κN . If the slowest execution of algorithm 3 takes time $T(N, \kappa N)$, then the overall algorithm has complexity $O(\tau T(N, \kappa N)) = O(T(N, \kappa N))$. ■

Lemma 3.2. *Assuming the function $\text{FINDKNN}(k, \mathcal{V}, D)$ defined in algorithm 4, which implements NNDescent [23], completes in time $O(k^2 N \log N)$ (as conjectured in [24,25]), for $N = |\mathcal{V}|$, then the function $\text{FINDBEST}(\kappa N, \mathcal{V}, D)$ completes in time*

$$O\left[\left(\sum_{t=0}^r \frac{1}{s_t}\right) \kappa^2 N \log N\right], \quad (3.1)$$

with $s_t = N_t/N$, where N_t is the number of nodes being considered at recursion level t and $t = r + 1$ is the first recursion level that results in $N_t \leq 2\sqrt{\kappa N}$.

²We emphasize that this argument relies on the *initial* random KNN graph having a small diameter, *not* the final KNN graph, which can have an arbitrary structure. The initial random graph bounds how much information can be exchanged between the nodes until convergence, whose speed is not affected by the final configuration, and hence we make no assumption on its structure.

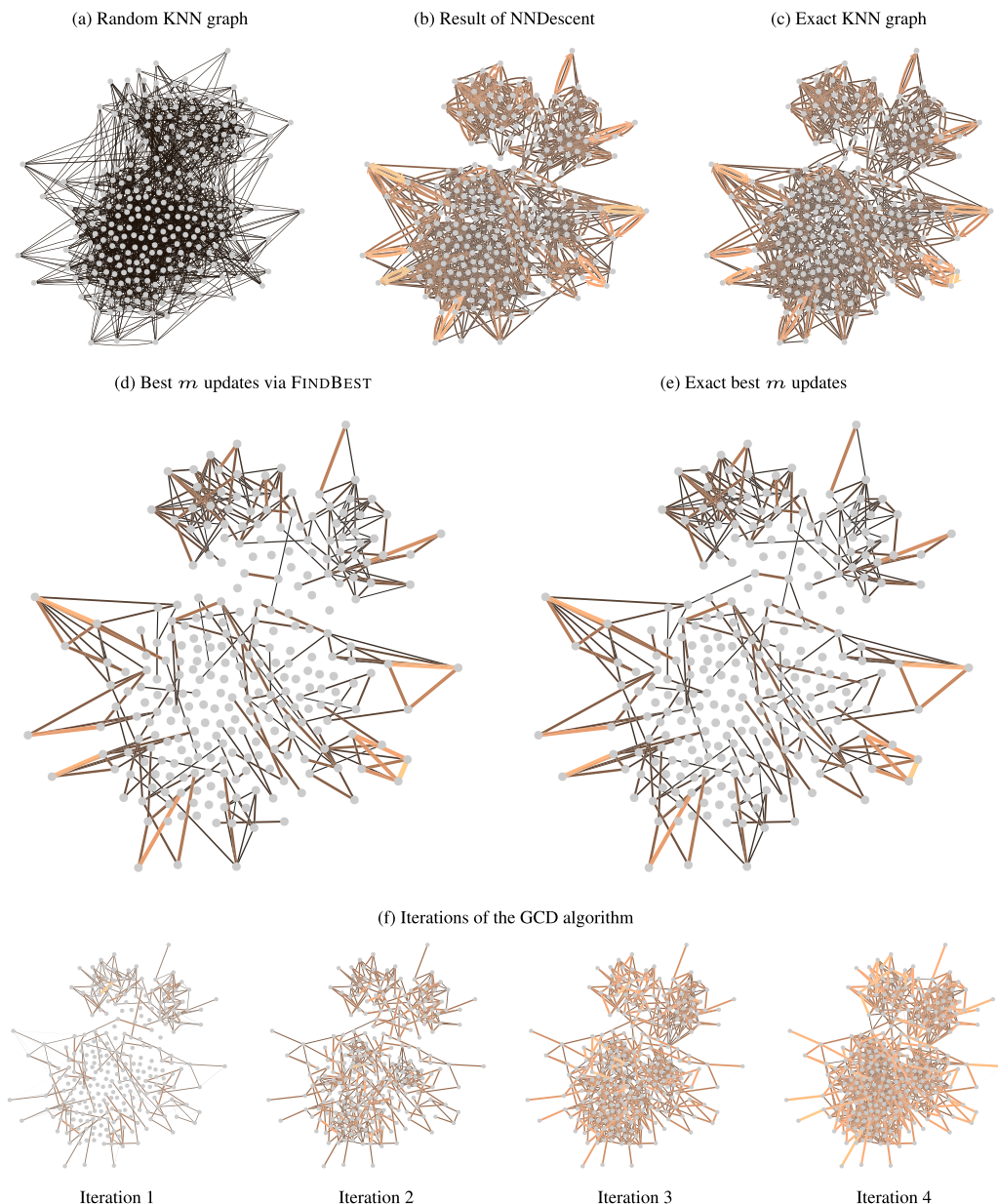


Figure 2. Example of our GCD algorithm for a covariance selection problem on a simulated dataset composed of $M = 500$ samples given a network of friendships among high-school students [35]. The panels show intermediary results of the algorithm, starting from an empty network [i.e. $W_{ij} = 0$ for all (i, j)]. The top row shows (a) the random initialization of NNDescent (algorithm 4) with $k = 4$, (b) its final result, and (c) the exact result found with an exhaustive algorithm. The middle row shows (d) the result of the $m = \kappa N$ best updates using algorithm 3 with $\kappa = 1$ and (e) the exact result according to an exhaustive algorithm. The edge colours indicate the value of $\max_{W_{ij}} \pi(W)$. The bottom row shows the first four iterations of the GCD algorithm.

Proof. Our proof follows closely [33], in which FINDKNN was required to finish in time $O(kN \log N)$, instead of $O(k^2N \log N)$ as is expected for NNDescent.

When using NNDescent, algorithm 3 has a complexity given recursively by

$$T(N, m) = O(k^2N \log N + m \log m) + T(|S'|, m), \quad (3.2)$$

with $k = \lceil 4mN^{-1} \rceil$, and boundary condition $T(N, m) = O(N^2)$ if $N^2 \leq 4m$ (the $m \log m$ term accounts for the sorting of the m pairs just before the function returns).

In general, ignoring an overall multiplicative constant, we can write

$$T(N, m) = \sum_{t=0}^r \frac{m^2}{N_t} \log N_t + (r+1)m \log m + O(m), \quad (3.3)$$

where $N_t = |S'_{t-1}|$ is the number of nodes being considered at recursion t , with S'_t being the set S' at recursion t (assuming $S'_{-1} = \mathcal{V}$) and $t = r+1$ is the first point at which $N_t \leq 2\sqrt{m}$, and hence the final recursion runs in time $O(m)$. Introducing $s_t = N_t/N$ as the fraction of nodes at recursion t , we can write

$$T(N, \kappa N) = \kappa^2 N \left[(\log N) \sum_{t=0}^r \frac{1}{s_t} + \sum_{t=0}^r \frac{\log s_t}{s_t} \right] + (r+1)\kappa N \log \kappa N + O(\kappa N). \quad (3.4)$$

Since $\log s_t \leq 0$ and $1/s_t \geq 1$, this will lead to an overall complexity of

$$T(N, \kappa N) = O \left[\left(\sum_{t=0}^r \frac{1}{s_t} \right) \kappa^2 N \log N \right]. \quad (3.5)$$

■

The prefactor in the above expression will, in general, depend on the data and the stage of the algorithm, as we shall see.

We can now obtain a loose upper bound to the running time of `FINDBEST` by assuming the worst case where the progression of the algorithm is (in principle) the slowest.

Theorem 3.3 (Upper bound on `FINDBEST`). *Under the same conditions as in lemma 3.2, the function `FINDBEST`($\kappa N, \mathcal{V}, D$) completes in time upper bounded by*

$$O(\kappa^{3/2} N^{3/2} \log N). \quad (3.6)$$

Proof. We first observe that we must always have $s_t \leq 1/2^t$, i.e. the number of nodes being considered must decay at least exponentially fast with each recursion. This is because in algorithm 3 we have $|D_t^+| \geq k|S'_t|$ and $|D_t^+| = 2m$, and thus $N_{t+1} = |S'_{t+1}| \leq 2m/k = 2m/\lceil 4m/N_t \rceil \leq N_t/2$, and hence $s_{t+1} \leq s_t/2$, which leads to $s_t \leq 1/2^t$ since $s_0 = 1$. Therefore, the worst case is $s_t = 1/2^t$, giving us $\sum_{t=0}^r 1/s_t = \sum_{t=0}^r 2^t = 2^{r+1} - 1$, and $2^{r+1} = \sqrt{N/4\kappa}$, and hence a complexity of

$$T(N, \kappa N) = O(\kappa^{3/2} N^{3/2} \log N). \quad (3.7)$$

■

However, although already subquadratic, this upper bound is not tight. This is because it is not in fact possible for the worst case $s_t = 1/2^t$ to be realized at every recursion, and the number of nodes being considered will generically decrease faster than exponentially. We notice this by performing a more detailed analysis of the runtime, as follows.

Lemma 3.4. *The worst case $s_t = 1/2^t$ considered in the proof of theorem 3.3 is not realizable.*

Proof. Let A be the graph consisting of N nodes and the m closest pairs we want to find as edges (i.e. the output of the `FINDBEST` function). Further, let $P(d)$ be the degree distribution of A (i.e. the fraction of nodes with degree d), and $F(d) = \sum_{d'=d}^{\infty} P(d')$ the tail cumulative distribution function of $P(d)$. At recursion t , we discover at most $k_t = \lceil 4m/N_t \rceil = \lceil 4\kappa/s_t \rceil$ neighbours of each node in A . Therefore, every node in A with degree $d \geq k_t$ will belong to set S'_t and be considered for the next

recursion $t + 1$. This means that we can write

$$N_{t+1} = NF(k_t), \quad (3.8)$$

and dividing by N on both sides leads to

$$s_{t+1} = F\left(\left\lceil \frac{4\kappa}{s_t} \right\rceil\right), \quad (3.9)$$

which no longer depends on N .

At this point, we can see why the worst case $s_t = 1/2^t$ considered for the upper bound in theorem 3.3 is strictly impossible: it would correspond to $F(d) = 2\kappa/d$ for $d \geq 4\kappa$ (this can be verified by inserting $F(d)$ in equation (3.9) and iterating) which is incompatible with the mean of $P(d)$ being finite and equal to 2κ —a requirement of the graph A having N nodes and $m = \kappa N$ edges. We notice this by writing the mean as $2\kappa = \sum_d dP(d) = \sum_d d[F(d) - F(d+1)] = \sum_d F(d) - 1$, and the fact that the sum $\sum_{d=4\kappa}^{\infty} 1/d$ diverges. ■

Given some valid $F(d)$, we can obtain the prefactor in equation (3.5) using the fact that the algorithm terminates when $s_{r+1} \leq \sqrt{4\kappa/N}$, and therefore we can recursively invert equation (3.9) as

$$\frac{1}{s_r} = \frac{F^{-1}(\sqrt{4\kappa/N})}{4\kappa} \quad (3.10)$$

and

$$\frac{1}{s_t} = \frac{F^{-1}(s_{t+1})}{4\kappa} \quad (\text{for } 0 < t < r). \quad (3.11)$$

The recursion depth and fraction of nodes considered will depend on the degree distribution of A via $F^{-1}(z)$ and, therefore, it will vary for different instances of the problem.

We now consider a few representative cases for how the output of FINDBEST may be, and consider their respective running times.

Theorem 3.5. *If the output of FINDBEST($\kappa N, \mathcal{V}, D$) consists of a d -regular graph, then its running time is $O(\kappa^2 N \log^2 N)$.*

Proof. If the graph A is d -regular, i.e. every node has degree of exactly 2κ (assuming this is an integer), corresponding to the extreme case of maximum homogeneity, this gives us $F(d) = \{1 \text{ if } d \leq 2\kappa, 0 \text{ otherwise}\}$, and, therefore, it follows directly from equation (3.9) that $s_t = 0$ for $t > 0$, and hence the overall complexity becomes simply

$$T(N, \kappa N) = O(\kappa^2 N \log N), \quad (3.12)$$

corresponding to a single call of the KNN algorithm. ■

We move now to a case with intermediary heterogeneity, with the output following a geometric degree distribution.

Theorem 3.6. *If the output of FINDBEST($\kappa N, \mathcal{V}, D$) consists of a graph with a geometric degree distribution $P(d) = (1-p)^d p$, with $(1-p)/p = 2\kappa$, then its running time becomes*

$$O(\kappa^2 N \log^2 N). \quad (3.13)$$

Proof. A geometric distribution gives us $F(d) = [2\kappa/(2\kappa+1)]^d$, and $F^{-1}(z) = \log z / \log[2\kappa/(2\kappa+1)]$. Inserting this in equation (3.11) yields

$$\frac{1}{s_r} = \frac{\log \sqrt{N/(4\kappa)}}{4\kappa \log[(2\kappa+1)/(2\kappa)]} = O(\log N), \quad (3.14)$$

$$\frac{1}{s_{r-1}} = \frac{\log 1/s_r}{4\kappa \log[(2\kappa+1)/(2\kappa)]} = O(\log \log N) \quad (3.15)$$

$$\text{and} \quad \frac{1}{s_{r-2}} = \frac{\log 1/s_{r-1}}{4\kappa \log[(2\kappa+1)/(2\kappa)]} = O(\log \log \log N), \quad (3.16)$$

and so on, such that the term $1/s_r$ dominates, giving us an overall log-linear complexity,

$$T(N, \kappa N) = O(\kappa^2 N \log^2 N). \quad (3.17)$$

This result means that for relatively more heterogeneous degree distributions we accrue only an additional $\log N$ factor in comparison to the d -regular case, and remain fairly below the upper bound found previously.

Based on the above, we can expect broader degree distributions in A to cause longer runtimes. A more extreme case is given by the Zipf distribution, which we now consider.

Theorem 3.7. *If the output of $\text{FINDBEST}(\kappa N, \mathcal{V}, \mathcal{D})$ consists of a graph with Zipf degree distribution $P(d) = d^{-\alpha}/\zeta(\alpha)$, where $\zeta(\alpha)$ is the Riemann zeta function, with α chosen so that the mean is 2κ , then its runtime becomes*

$$O(\kappa^{1-(1/2(\alpha-1))} N^{1+(1/2(\alpha-1))} \log N). \quad (3.18)$$

Proof. In this case, we can approximate $F(d) = \sum_{d'=d}^{\infty} P(d') \approx \zeta(\alpha)^{-1} \int_d^{\infty} x^{-\alpha} dx \propto d^{1-\alpha}$, and $F^{-1}(z) \propto z^{1/(1-\alpha)}$. Substituting the above into equation (3.11), we get

$$\frac{1}{s_r} = \frac{(\sqrt{4\kappa/N})^{1/1-\alpha}}{4\kappa} = O(\kappa^{\frac{1}{2(1-\alpha)}-1} N^{1/2(\alpha-1)}), \quad (3.19)$$

$$\frac{1}{s_{r-1}} = \frac{(1/s_r)^{1/\alpha-1}}{4\kappa} = O(\kappa^{1/2(1-\alpha)-2} N^{1/2(\alpha-1)^2}) \quad (3.20)$$

$$\text{and} \quad \frac{1}{s_{r-\ell}} = \frac{(1/s_{r-\ell+1})^{1/\alpha-1}}{4\kappa} = O(\kappa^{1/2(1-\alpha)-\ell-1} N^{1/2(\alpha-1)^{\ell+1}}), \quad (3.21)$$

which is again dominated by $1/s_r$, and hence gives us

$$T(N, \kappa N) = O(\kappa^{1-(1/2(\alpha-1))} N^{1+(1/2(\alpha-1))} \log N). \quad (3.22)$$

The value of α is not a free parameter, since it needs to be compatible with the mean degree 2κ . For very large $2\kappa \gg 1$, we have $\alpha \rightarrow 2$, and hence the complexity will be asymptotically similar to the upper bound we found previously, i.e.

$$T(N, \kappa N) = O(\kappa^{1/2} N^{3/2} \log N), \quad (3.23)$$

although this is not how the algorithm is realistically evoked, as we need $\kappa = O(1)$ for a reasonable performance. For example, if $\alpha = 5/2$ we have $2\kappa \approx 1.947$, and hence a complexity of $O(\kappa^{2/3} N^{4/3} \log N)$, and for low $2\kappa \rightarrow 1$ we have $\alpha \rightarrow \infty$, yielding the lower limit

$$T\left(N, \frac{N}{2}\right) = O(N \log N), \quad (3.24)$$

compatible with the d -regular case for $d = 1$, as expected. Therefore, even in such an extremely heterogeneous case, the complexity remains close to log-linear for reasonably small values of κ .

We emphasize that the graph A considered for the complexity analysis above is distinct from the final network W we want to reconstruct. The latter might have an arbitrary structure, but the graph A represents only the updates that need to be performed, and it has a density which is controlled by the parameter κ of our algorithm. Thus, even if W has a very broad degree distribution, the one we see in A will be further limited by the parameter κ and for which updates are needed by the GCD algorithm, and in general W and A will be very different from each other (for an example, compare panels (d) and (f) in figure 2).

4. Performance assessment

We now conduct an analysis of the performance of our algorithm in a variety of artificial and empirical settings. We begin with an analysis of the runtime of the FINDBEST function

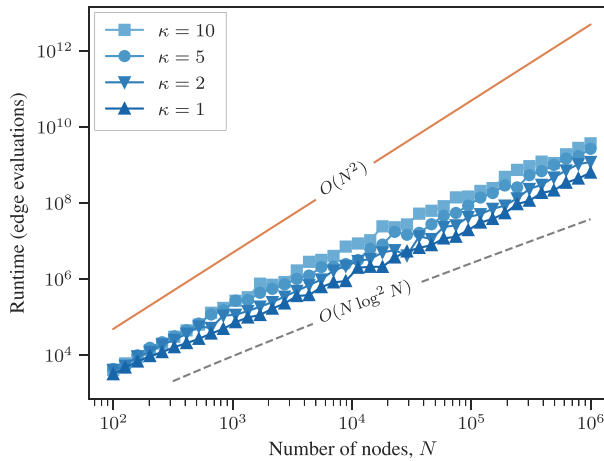


Figure 3. Runtime of the `FindBest` function (algorithm 3), for different values of κ , on $M = 10$ samples of a multivariate Gaussian (see appendix B) on N nodes and non-zero entries of \mathbf{W} sampled as an Erdős–Rényi graph with mean degree 5 and non-zero weights independently normally sampled with mean -10^3 and standard deviation 10, and diagonal entries $W_{ii} = \sum_{j \neq i} |W_{ij}| / (1 - \epsilon)^2$ with $\epsilon = 10^{-3}$. The results show averages over 10 independent problem instances.

(algorithm 3) on $M = 10$ samples of a multivariate Gaussian (see appendix B) on N nodes and non-zero entries of \mathbf{W} sampled as an Erdős–Rényi graph with mean degree 5. As we see in figure 3, the runtime of the algorithm is consistent with a $O(N \log^2 N)$ scaling as obtained theoretically in the previous section for a homogeneous update graph.

This kind of log-linear performance is promising for the scalability of the overall algorithm, but it stills needs to be determined if the results of `FINDBEST` are sufficiently accurate for a speedy progression of the GCD algorithm. In figure 4, we show the cumulative recall rates of the exact best pairs obtained with an exhaustive algorithm, defined as the fraction of best pairs correctly identified up to a given rank position (with the best pair having rank 1), for a variety of empirical data. We observe in general very good recall rates, compatible with what is obtained with the `NNDescent` algorithm [23]. Importantly, in every case we considered, we observed that the cumulative recall values start at 1, meaning that the first best edges are always correctly identified. For some data, the algorithm may falter slightly for intermediary pairs and a low κ , but increasing κ has the systematic effect of substantially improving the recall rates (at the expense of longer runtimes). We emphasize again that it is not strictly necessary for the `FINDBEST` function to return exact results, since it will be called multiple times during the GCD loop, and its random initialization guarantees that every pair will eventually be considered—it needs only to be able to locate the best edge candidates with high probability. Combined with the previous result of figure 3, this indicates that the fast runtime of the `FINDBEST` function succeeds in providing the GCD algorithm with the entries of the \mathbf{W} matrix that are most likely to improve the objective function, while avoiding an exhaustive $O(N^2)$ search.

We can finally evaluate the final performance of the GCD algorithm by its convergence speed, as shown in figure 5 for artificial and empirical data. For small data with $N = 100$, we observe only a modest improvement over the CD baseline, but this quickly improves for larger N : for $N = 1000$ we already observe a $100\times$ runtime improvement, which increases to $1000\times$ for $N = 10000$. Interestingly, we observe that the $\kappa = 1$ results show the fastest convergence, indicating that the decreased accuracy of the `FINDBEST` function—resulting in it needing to be called more often—is compensated by its improved runtime. For $\kappa = 10$ we see that the speed of convergence per iteration is virtually the same as the CD baseline, but it significantly outperforms it in real time. This seems to demonstrate that, as expected for the reconstruction of a sparse network, most $O(N^2)$ updates performed by the CD baseline algorithm are wasteful, and only a $O(N)$ subset is actually needed at each stage for the actual progression of the algorithm—which the `FINDBEST`

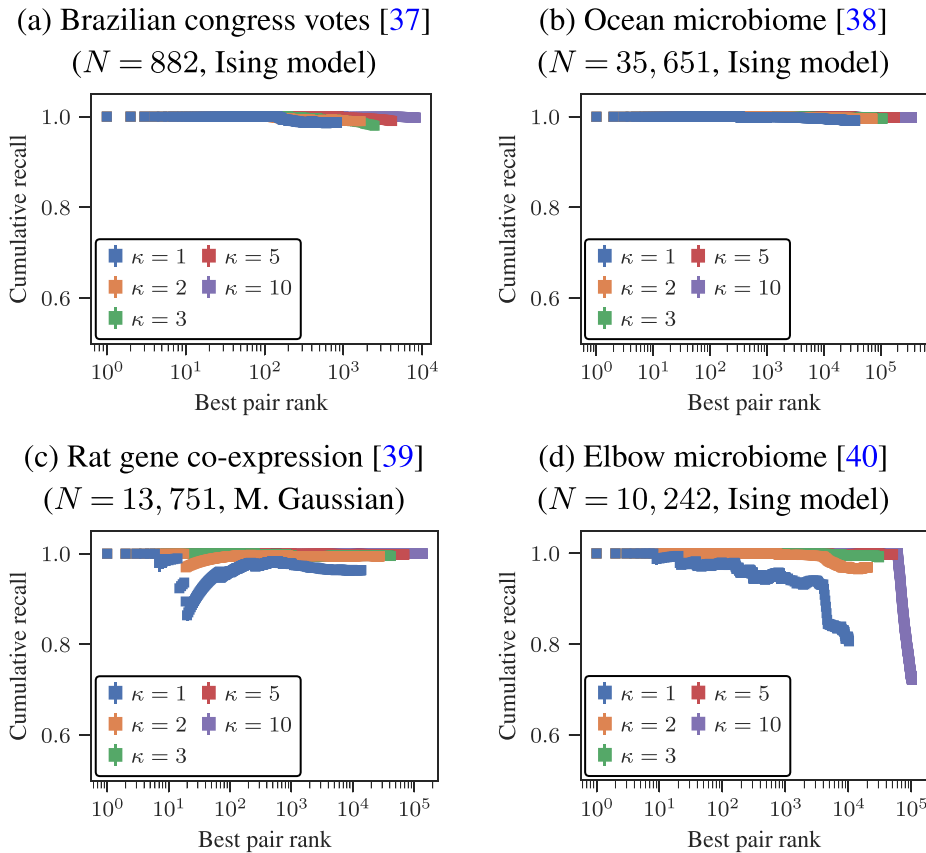


Figure 4. Cumulative recall rates for the *FindBest* function for different values of κ on a variety of empirical data and reconstruction objectives, as shown in the legend (see appendix B). The results show averages over 10 runs of the algorithm.

function is capable of identifying in subquadratic time. We explore this in more detail in figure 6, where we see that for $\kappa > 2$ the performance per iteration is virtually indistinguishable from the CD baseline, but the runtime increases monotonically with $\kappa > 1$ and decreases with $\kappa < 1$, giving us an optimum at $\kappa = 1$, which we therefore judge to be the best initial value to be used in most applications.

The above results are representative of what we have observed on a larger set of empirical data (not shown). We were not able to find a single instance of an empirical or artificial scenario that contradicts the log-linear runtime of the *FINDBEST* function, or where our algorithm does not provide a significant improvement over the $O(N^2)$ CD baseline (except for very small N).

5. Empirical examples

We demonstrate the use of our algorithm with a few large-scale datasets, which would be costly to analyse with a quadratic reconstruction algorithm. We showcase on the following:

(a) Earth microbiome project [41]

A collection of crowd-sourced microbial samples from various biomes and habitats across the globe, containing abundances of operational taxonomic units (OTUs), obtained via DNA sequencing and mass spectrometry. An OTU is a proxy for a microbial species, and a single sample consists of the abundances of each OTU measured at the same time. This dataset consists of $M = 23\,828$ samples involving $N = 317\,314$ OTUs.

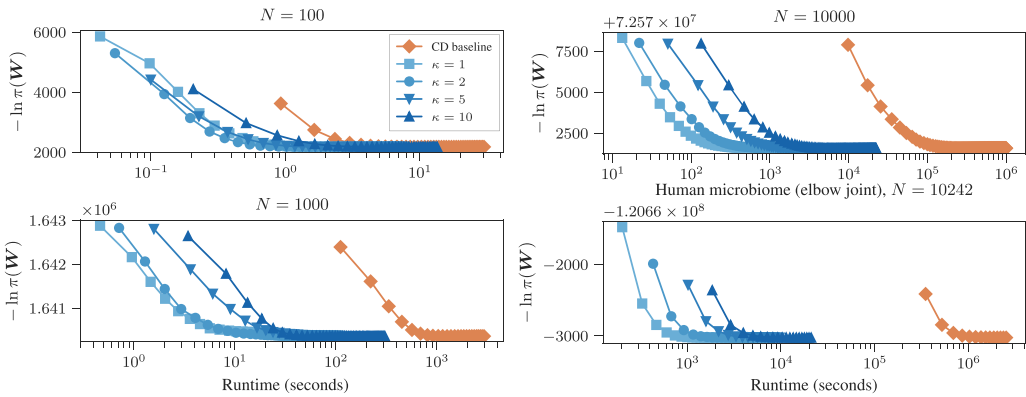


Figure 5. Convergence of the CD and GCD algorithms for artificial data sampled from a multivariate Gaussian (same parameterization as figure 3 but with $M = 100$ samples) for three different values of the number of nodes N and values of κ , together with the CD baseline. The bottom panel shows the results obtained for empirical data for the human microbiome samples of the elbow joint, using the Ising model.

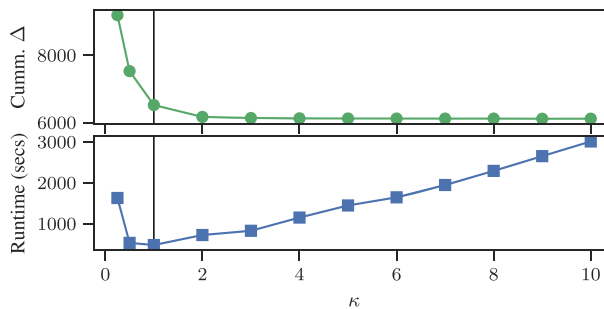


Figure 6. Speed of convergence of the GCD algorithm as a function of κ for the same artificial data as in figure 5, for $N = 10^4$. The top panel shows the cumulative sum of the absolute values of Δ in algorithm 2, and the bottom panel the total runtime in seconds until convergence. The solid lines correspond to $\kappa = 1$.

(b) Human microbiome project [40]

A collection of microbial samples from 300 healthy adults between the ages of 18 and 40, collected at five major body sites (oral cavity, nasal cavity, skin, gastrointestinal tract and urogenital tract) with a total of 15 or 18 specific body sites. The abundances of OTUs were obtained using 16S rRNA and whole metagenome shotgun (mWGS) sequencing. This dataset consists of $M = 4788$ samples involving $N = 45\,383$ OTUs.

For both co-occurrence datasets above, we binarized each sample as $x_i \in \{-1, 1\}$, for absence and presence, respectively, and used the Ising model for the network reconstruction (see appendix B). In this case, the matrix \mathbf{W} yields the strength of the coupling between two OTUs, and a value $W_{ij} = 0$ means that OTUs i and j are conditionally independent from one another.

(c) Animal gene expression database COXPRESdb [39]

This database consists of genome-wide gene expression reads for 10 animal species as well as budding and fission yeast. Each gene expression sample was obtained using RNAseq, with read counts converted to base-2 logarithms after adding a pseudo-count of 0.125, and batch corrected using Combat [42]. We used the nematode dataset, consisting of $M = 1357$ samples of $N = 17\,256$

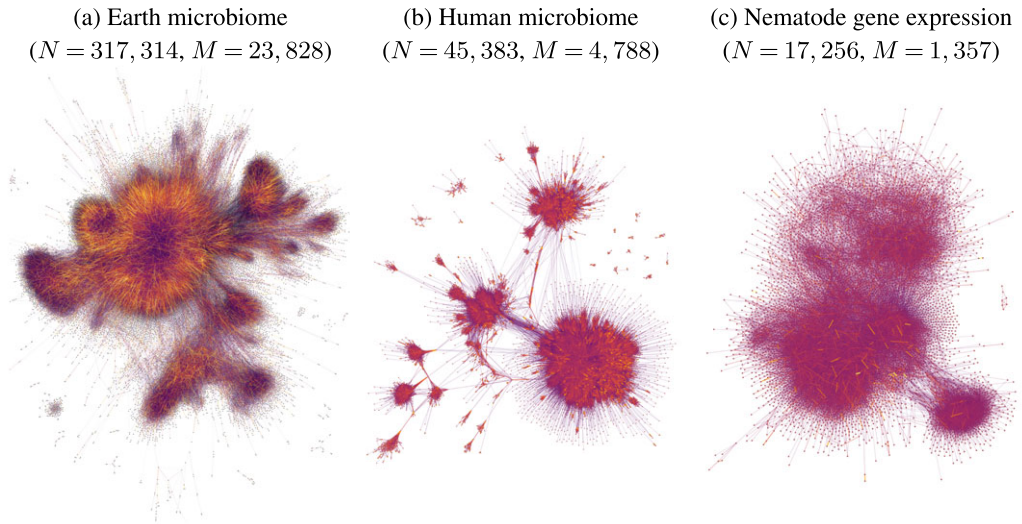


Figure 7. Reconstructed networks for the empirical datasets described in the text, as shown in the legend, using the Ising model for (a) and (b), and multivariate Gaussian for (c). The edge colours indicate the magnitude of the entries of the W matrix. Nodes with degree zero were excluded from the visualization.

genes. For this dataset, we used the multivariate Gaussian model (see appendix. B), where the matrix W corresponds to the inverse covariance between gene expression levels (also known as precision matrix). In this case, the partial correlation between two genes, i.e. their degree of association controlling for all other genes, is given by $-W_{ij}/\sqrt{W_{ii}W_{jj}}$, so that gene pairs with $W_{ij} = 0$ are conditionally independent.

The reconstructed networks for all three datasets are shown in figure 7. They all seem to display prominent modular structure. In the case of microbial co-occurrence datasets the clusters correspond mostly to different habitats and geographical regions for the earth microbiome, and to different body sites for the human microbiome.

6. Conclusion

We have described a method to reconstruct interaction networks from observational data that avoids a seemingly inherent quadratic complexity in the number of nodes, in favour of a data-dependent runtime that is typically log-linear. Our algorithm does not rely on particular formulations of the reconstruction problem, other than the updates on the edge weights being done in constant time with respect to the total number of nodes. Together with its straightforward parallelizability, our proposed method removes a central barrier to the reconstruction of large-scale networks and can be applied to problems with a number of nodes and edges on the order of hundreds of thousands, millions or potentially even more depending on available computing resources.

Our algorithm relies on the NNDescent [23] approach for approximate KNN search. Despite the robust empirical evidence for its performance, a detailed theoretical analysis of this algorithm is still lacking, in particular of its potential modes of failures. We expect further progress in this direction to elucidate potential limitations and improvements to our overall approach.

In this work, we focused on convex reconstruction objectives, such as the inverse Ising model and multivariate Gaussian with L_1 regularization. More robust regularization schemes or different models may no longer be convex, in which case CD will fail, in general, at converging to the global optimum. However, it is clear that our strategy of finding the best edge candidates in

subquadratic time is also applicable for algorithms that can be used with non-convex objectives, such as stochastic gradient descent or simulated annealing. We leave such extensions for future work.

Data accessibility. All amplicon sequence data and metadata have been made public through the EMP data portal (<http://qiita.microbio.me/emp>).

Declaration of AI use. I have not used AI-assisted technologies in creating this article.

Authors' contributions. T.P.: conceptualization, data curation, formal analysis, funding acquisition, investigation, methodology, project administration, resources, software, supervision, validation, visualization, writing—original draft, writing—review and editing

Conflict of interest declaration. I declare I have no competing interests.

Funding. This work has been funded by the Vienna Science and Technology Fund (WWTF) and by the State of Lower Austria [Grant ID: 10.47379/ESS22032].

Acknowledgements. Sample processing, sequencing, and core amplicon data analysis were performed by the Earth Microbiome Project (www.earthmicrobiome.org), and all amplicon sequence data and metadata have been made public through the EMP data portal (<http://qiita.microbio.me/emp>).

Appendix A. Single entry updates

In the main text, it is assumed that a single coordinate update of the matrix \mathbf{W} , i.e. $W_{ij} \rightarrow W'_{ij}$, can be performed in constant time $O(1)$ with respect to the number of nodes N . This is feasible, for example, when the likelihood of equation (2.1) can be written as a function of the weighted sum of the states of the neighbours, $u_i = \sum_j W_{ji}x_j$. For example, for a Markovian dynamical model this means that the probability of transition from state x_m to x_{m+1} can be written in the general form

$$P(x_{m+1}|x_m, \mathbf{W}, \boldsymbol{\theta}) = \prod_{i=1}^N P\left(x_{i,m+1} \left| \sum_j W_{ji}x_{j,m}, \boldsymbol{\theta} \right.\right), \quad (\text{A } 1)$$

where $\boldsymbol{\theta}$ is an arbitrary set of additional parameters. In this case, a constant time update $W_{ij} \rightarrow W'_{ij}$ can be computed simply by keeping track of the quantities $u_{im} = \sum_j W_{ji}x_{j,m}$ at all times, since the change of a single entry W_{ij} will only affect the values u_{im} and u_{jm} (assuming \mathbf{W} is symmetric), for every m . The form of equation (A 1) encompasses a wide array of standard dynamical network models, including binary state models such as epidemic models [37,43], the kinetic Ising model (Glauber dynamics), the voter model, boolean threshold models and several others [44], as well as a very large class of differential equation models (including, e.g. generalized Lotka–Volterra systems [45] and coupled Kuramoto oscillators [46]). The form of equation (A 1) also describes the pseudolikelihood approximation [47] for pairwise graphical models. Because of this, we expect that the scenario considered in the main text to be widely representative of network reconstruction problems.

Nevertheless, for problems that lie outside of the aforementioned class, if a single coordinate update has an arbitrary algorithmic complexity given by $\Delta(N)$, then the CD baseline would have an overall complexity of $O(\Delta(N)N^2)$, while our improved algorithm would have a typical complexity of $O(\Delta(N)N \log^2 N)$, thereby providing precisely the same relative speedup.

Appendix B. Generative models

In our examples, we use two graphical models: the Ising model [17] and a multivariate Gaussian distribution [12]. The Ising model is a distribution on N binary variables $x \in \{-1, 1\}^N$ given by

$$P(x|\mathbf{W}, \boldsymbol{\theta}) = \frac{e^{\sum_{i<j} W_{ij}x_ix_j + \sum_i \theta_i x_i}}{Z(\mathbf{W}, \boldsymbol{\theta})}, \quad (\text{B } 1)$$

with θ_i being a local field on node i and $Z(\mathbf{W}, \boldsymbol{\theta}) = \sum_x e^{\sum_{i<j} W_{ij}x_ix_j + \sum_i \theta_i x_i}$ a normalization constant. Since this normalization cannot be computed in closed form, we make use of the pseudolikelihood

approximation [47],

$$P(x|\mathbf{W}, \boldsymbol{\theta}) = \prod_i P(x_i | x \setminus x_i, \mathbf{W}, \boldsymbol{\theta}) \quad (\text{B2})$$

$$= \prod_i \frac{e^{x_i(\sum_j W_{ij}x_j + \theta_i)}}{2 \cosh(\sum_j W_{ij}x_j + \theta_i)}, \quad (\text{B3})$$

as it gives asymptotically correct results and has excellent performance in practice [17]. Similarly, the (zero-mean) multivariate Gaussian is a distribution on $x \in \mathbb{R}^N$ given by

$$P(x|\mathbf{W}) = \frac{e^{-\frac{1}{2}x^\top \mathbf{W}x}}{\sqrt{(2\pi)^N |\mathbf{W}^{-1}|}}, \quad (\text{B4})$$

where \mathbf{W} is the precision (or inverse covariance) matrix. Unlike the Ising model, this likelihood is analytical—nevertheless, the evaluation of the determinant is computationally expensive, and therefore we make use of the same pseudolikelihood approximation [48],

$$P(x|\mathbf{W}, \boldsymbol{\theta}) = \prod_i \frac{e^{-(x_i + \theta_i^2 \sum_{j \neq i} W_{ij}x_j)^2 / 2\theta_i^2}}{\sqrt{(2\pi)\theta_i}}, \quad (\text{B5})$$

where we parameterize the diagonal entries as $\theta_i = 1/\sqrt{W_{ii}}$.

In both cases, we have an additional set of N parameters $\boldsymbol{\theta}$ which we update alongside the matrix \mathbf{W} in our algorithms. Updates on an individual entry W_{ij} of \mathbf{W} can be computed in time $O(1)$ (independently of the degrees of i and j in a sparse representation of \mathbf{W}) by keeping track of the weighted sum of the neighbours $m_i = \sum_{j \neq i} W_{ij}x_j$ for every node and updating it as appropriate.

For both models, we use a Laplace prior,

$$P(\mathbf{W}|\lambda) = \frac{\prod_{i < j} \lambda e^{-\lambda |W_{ij}|}}{2}, \quad (\text{B6})$$

which provides a convex L_1 regularization with a penalty given by λ , chosen to achieve a desired level of sparsity via k -fold cross-validation (in our experiments we chose $k = 5$). This prior is not ideal, as it tends to cause overfitting. In [49], we provide a more robust alternative that can also be used with our proposed algorithm.

Appendix C. Low-level optimizations

Below, we describe a few low-level optimizations that we found to give good improvements to the runtime of the algorithm we propose in the main text.

(a) Caching

The typical case for objective functions $\pi(\mathbf{W})$ is that the computation of $\max_{W_{ij}} \pi(\mathbf{W})$ will require $O(M)$ operations, where M is the number of data samples available. Since this computation is done in the innermost loops of algorithm 4, it will amount to an overall multiplicative factor of $O(M)$ in its runtime. However, because the distance function $D(i, j)$ will be called multiple times for the same pair (i, j) , a good optimization strategy is to cache its values, for example in a hash table, such that repeated calls will take time $O(1)$ rather than $O(M)$. We find that this optimization can reduce the total runtime of algorithm 4 by at least one order of magnitude in typical cases.

(b) Gradient as distance

The definition of $D(i, j) = -\max_{W_{ij}} \pi(\mathbf{W})$ is sufficient to guarantee the correctness of algorithm 3, but in situations where it cannot be computed in closed form, requiring for example a bisection

search, a faster approach is to use instead the absolute gradient $D(i, j) = -|\partial/\partial W_{ij} \log \pi(\mathbf{W})|$, which often can be analytically computed or well approximated with finite difference. In general, this requires substantially fewer likelihood evaluations than bisection search. This approach is strictly applicable only with differentiable objectives, although we observed correct behaviour for L_1 -regularized likelihoods when approximating the gradient using central finite difference. We observed that this optimization improves the runtime by a factor of approximately six in typical scenarios.

(c) Parallelization

The workhorse of the algorithm is the NNDescent search (algorithm 4), which is easily parallelizable in a shared memory environment, since the neighbourhood of each node can be inspected, and its list of nearest neighbours can be updated, in a manner that is completely independent of the other nodes, and hence requires no synchronization. Thus, the parallel execution of algorithm 4 is straightforward.

The actual updates of the matrix \mathbf{W} in the GCD algorithm 2 can also be done in parallel, but that requires some synchronization. For many objectives $\pi(\mathbf{W})$, we can only consider the change of one value W_{ij} at a time for each node i and j , since the likelihood will involve sums of the type $m_i = \sum_j W_{ij} x_j$ for a node i , where x_j are data values. Therefore, only the subset of the edges $\mathcal{E}_{\text{best}}$ in algorithm 2 that are incident to an *independent vertex set* in the graph will be able to be updated in parallel. This can be implemented with mutexes on each node, which are simultaneously locked by each thread (without blocking) for each pair (i, j) before W_{ij} is updated, which otherwise proceeds to the next pair if the lock cannot be acquired. Empirically, we found that is enough to keep up to 256 threads busy with little contention for $N > 10^4$ with our OpenMP implementation [36].

References

1. Lauritzen SL. 1996 *Graphical models*, vol. 17. Oxford: Clarendon Press.
2. Jordan MI. 2004 Graphical models. *Stat. Sci.* **19**, 140–155. (doi:10.1214/088342304000000026)
3. Drton M, Maathuis MH. 2017 Structure learning in graphical modeling. *Annu. Rev. Stat. Appl.* **4**, 365–393. (doi:10.1146/annurev-statistics-060116-053803)
4. Timme M, Casadiego J. 2014 Revealing networks from dynamics: an introduction. *J. Phys. A: Math. Theor.* **47**, 343001. (doi:10.1088/1751-8113/47/34/343001)
5. Hallac D, Park Y, Boyd S, Leskovec J. 2017 Network inference via the time-varying graphical lasso. (<http://arxiv.org/abs/10.48550/arXiv.1703.01958>)
6. Guseva K, Darcy S, Simon E, Alteio LV, Montesinos-Navarro A, Kaiser C. 2022 From diversity to complexity: microbial networks in soils. *Soil Biol. Biochem.* **169**, 108604. (doi:10.1016/j.soilbio.2022.108604)
7. Bury T. 2013 A statistical physics perspective on criticality in financial markets. *J. Stat. Mech: Theory Exp.* **2013**, P11004. (doi:10.1088/1742-5468/2013/11/P11004)
8. Weigt M, White RA, Szurmant H, Hoch JA, Hwa T. 2009 Identification of direct residue contacts in protein–protein interaction by message passing. *Proc. Natl Acad. Sci. USA* **106**, 67–72. (doi:10.1073/pnas.0805923106)
9. D'haeseleer P, Liang S, Somogyi R. 2000 Genetic network inference: from co-expression clustering to reverse engineering. *Bioinformatics* **16**, 707–726. (doi:10.1093/bioinformatics/16.8.707)
10. Manning JR, Ranganath R, Norman KA, Blei DM. 2014 Topographic factor analysis: a Bayesian model for inferring brain networks from neural data. *PLoS ONE* **9**, e94914. (doi:10.1371/journal.pone.0094914)
11. Braunstein A, Ingrosso A, Muntoni AP. 2019 Network reconstruction from infection cascades. *J. R. Soc. Interface* **16**, 20180844. (doi:10.1098/rsif.2018.0844)
12. Dempster AP. 1972 Covariance selection. *Biometrics* **28**, 157–175. (doi:10.2307/2528966)
13. Friedman J, Hastie T, Tibshirani R. 2008 Sparse inverse covariance estimation with the graphical lasso. *Biostatistics* **9**, 432–441. (doi:10.1093/biostatistics/kxm045)

14. Mazumder R, Hastie T. 2012 The graphical lasso: new insights and alternatives. *Electron. J. Stat.* **6**, 2125–2149. (doi:10.1214/12-EJS740)
15. Hastie T, Tibshirani R, Wainwright M. 2015 *Statistical learning with sparsity: the lasso and generalizations*. Boca Raton, FL: CRC Press.
16. Bresler G, Mossel E, Sly A. 2008 Reconstruction of markov random fields from samples: some observations and algorithms. In *Approximation, Randomization and Combinatorial Optimization. Algorithms and Techniques*, Lecture Notes in Computer Science, pp. 343–356. Berlin, Heidelberg: Springer.
17. Nguyen HC, Zecchina R, Berg J. 2017 Inverse statistical problems: from the inverse Ising problem to data science. *Adv. Phys.* **66**, 197–261. (doi:10.1080/00018732.2017.1341604)
18. Hsieh CJ, Sustik MA, Dhillon IS, Ravikumar P. 2014 QUIC: quadratic approximation for sparse inverse covariance estimation. *J. Mach. Learn. Res.* **15**, 2911–2947.
19. Hsieh CJ, Sustik MA, Dhillon IS, Ravikumar PK, Poldrack R. 2013 BIG & QUIC: sparse inverse covariance estimation for a million variables. In *Advances in Neural Information Processing Systems*, vol. 26. Curran Associates, Inc.
20. Bresler G. 2015 Efficiently learning Ising models on arbitrary graphs. In *Proc. of the Forty-seventh Annual ACM Symposium on Theory of Computing STOC '15*, pp. 771–782. New York, NY: ACM.
21. Bresler G, Mossel E, Sly A. 2010 Reconstruction of markov random fields from samples: some easy observations and algorithms. (<http://arxiv.org/abs/10.48550/arXiv.0712.1402>)
22. Bresler G, Gamarnik D, Shah D. 2018 Learning graphical models from the glauher dynamics. *IEEE Trans. Inf. Theory* **64**, 4072–4080. (doi:10.1109/TIT.2017.2713828)
23. Dong W, Moses C, Li K. 2011 Efficient k-nearest neighbor graph construction for generic similarity measures. In *Proc. of the 20th Int. Conf. on World Wide Web WWW '11*, pp. 577–586. New York, NY: Association for Computing Machinery.
24. Baron JD, Darling RWR. 2020 K-nearest neighbor approximation via the friend-of-a-friend principle. (<http://arxiv.org/abs/10.48550/arXiv.1908.07645>)
25. Baron JD, Darling RWR. 2022 Empirical complexity of comparator-based nearest neighbor descent. (<http://arxiv.org/abs/10.48550/arXiv.2202.00517>)
26. Tseng P. 2001 Convergence of a Block Coordinate Descent Method for Nondifferentiable Minimization. *J. Optim. Theory Appl.* **109**, 475–494. (doi:10.1023/A:1017501703105)
27. Wright SJ. 2015 Coordinate descent algorithms. *Math. Program.* **151**, 3–34. (doi:10.1007/s10107-015-0892-3)
28. Spall JC. 2012 Cyclic Seesaw process for optimization and identification. *J. Optim. Theory Appl.* **154**, 187–208. (doi:10.1007/s10957-012-0001-1)
29. Abbeel P, Koller D, Ng AY. 2006 Learning factor graphs in polynomial time and sample complexity. *J. Mach. Learn. Res.* **7**, 1743–1788.
30. Wainwright MJ, Lafferty J, Ravikumar P. 2006 High-dimensional graphical model selection using ℓ_1 -regularized logistic regression. In *Advances in Neural Information Processing Systems*, vol. 19. Cambridge, MA: MIT Press.
31. Santhanam N, Wainwright MJ. 2009 Information-theoretic limits of selecting binary graphical models in high dimensions. (<http://arxiv.org/abs/10.48550/arXiv.0905.2639>)
32. Smid M. 2000 Closest-point problems in computational geometry. In *Handbook of Computational Geometry* (eds JR Sack, J Urrutia), pp. 877–935. Amsterdam, The Netherlands: North-Holland.
33. Lenhof HP, Smid M. 1992 The k closest pairs problem. Unpublished manuscript.
34. McInnes L, Healy J, Melville J. 2018 UMAP: uniform manifold approximation and projection for dimension reduction. (<http://arxiv.org/abs/1802.03426>)
35. Moody J. 2001 Race, school integration, and friendship segregation in america. *AJS* **107**, 679–716. (doi:10.1086/338954)
36. Peixoto TP. 2014 The graph-tool python library. Figshare. Available at <https://graph-tool.skewed.de>. (doi:10.6084/m9.figshare.1164194)
37. Peixoto TP. 2019 Network reconstruction and community detection from dynamics. *Phys. Rev. Lett.* **123**, 128301. (doi:10.1103/PhysRevLett.123.128301)
38. Sunagawa S *et al.* 2015 Structure and function of the global ocean microbiome. *Science* **348**, 1261359. (doi:10.1126/science.1261359)
39. Obayashi T, Kodate S, Hibara H, Kagaya Y, Kinoshita K. 2023 COXPRESdb v8: an animal gene coexpression database navigating from a global view to detailed investigations. *Nucleic Acids Res.* **51**, D80–D87. (doi:10.1093/nar/gkac983)

40. Huttenhower C *et al.* The Human Microbiome Project Consortium 2012 Structure, function and diversity of the healthy human microbiome. *Nature* **486**, 207–214. (doi:10.1038/nature11234)
41. Thompson LR *et al.* 2017 A communal catalogue reveals earth's multiscale microbial diversity. *Nature* **551**, 457–463. (doi: 10.1038/nature24621)
42. Johnson WE, Li C, Rabinovic A. 2007 Adjusting batch effects in microarray expression data using empirical Bayes methods. *Biostatistics* **8**, 118–127. (doi:10.1093/biostatistics/kxj037)
43. Pastor-Satorras R, Castellano C, Van Mieghem P, Vespignani A. 2015 Epidemic processes in complex networks. *Rev. Mod. Phys.* **87**, 925–979. (doi:10.1103/RevModPhys.87.925)
44. Gleeson JP. 2013 Binary-state dynamics on complex networks: pair approximation and beyond. *Phys. Rev. X* **3**, 021004. (doi:10.1103/PhysRevX.3.021004)
45. Bunin G. 2017 Ecological communities with Lotka-Volterra dynamics. *Phys. Rev. E* **95**, 042414. (doi:10.1103/PhysRevE.95.042414)
46. Arenas A, Díaz-Guilera A, Kurths J, Moreno Y, Zhou C. 2008 Synchronization in complex networks. *Phys. Rep.* **469**, 93–153. (doi:10.1016/j.physrep.2008.09.002)
47. Besag J. 1974 Spatial interaction and the statistical analysis of lattice systems. *J. R. Stat. Soc.: Series B (Methodological)* **36**, 192–225. (doi:10.1111/j.2517-6161.1974.tb00999.x)
48. Khare K, Oh SY, Rajaratnam B. 2015 A convex pseudolikelihood framework for high dimensional partial correlation estimation with convergence guarantees. *J. R. Stat. Soc. Ser. B: Stat. Methodol.* **77**, 803–825. (doi:10.1111/rssb.12088)
49. Peixoto TP. 2025 Network reconstruction via the minimum description length principle. *Phys. Rev. X* **15**, 011065. (doi:10.1103/PhysRevX.15.011065)