Inferring large graphs using l1-penalized likelihood

Magali Champion, Victor Picheny, Matthieu Vignes

Inferring large graphs using ℓ_1-penalized likelihood

Magali Champion
Laboratoire MAP5, Université Paris Descartes,
Sorbonne Paris Cité, France
Victor Picheny
INRA, UR875 Applied Mathematics and Computer Science Unit,
Castanet-Tolosan, France
and
Matthieu Vignes
Institute of Fundamental Sciences, Massey University,
Palmerston North, New Zealand.

Abstract

We address the issue of recovering the structure of large sparse directed acyclic graphs from noisy observations of the system. We propose a novel procedure based on a specific formulation of the ℓ_1-norm regularized maximum likelihood, which decomposes the graph estimation into two optimization sub-problems: topological structure and node order learning. We provide convergence inequalities for the graph estimator, as well as an algorithm to solve the induced optimization problem, in the form of a convex program embedded in a genetic algorithm. We apply our method to various data sets (including data from the DREAM4 challenge) and show that it compares favorably to state-of-the-art methods. This algorithm is available on CRAN as the R package `GADAG`.

Keywords: Directed Acyclic Graphs Lasso Convex program Optimization

1 Introduction

Revealing the true structure of a complex system is paramount in many fields to identify system regulators, predict its behavior or decide where interventions are needed to disentangle direct relationships (Newman, 2003; Souma et al., 2006; Verma et al., 2014). This problem can often be seen as a graph inference problem. Given observational data, we aim at predicting the presence (or absence) of edges between elements of the system, which form the vertices of a graph. Edges specify the relational structure of the system depicted as a graph or network. As a motivating problem, the reconstruction of Gene Regulatory Networks (GRN), which model activation and inhibition relationships between genes, is one of the main challenges in modern computational biology (Barabási & Oltvai, 2004).

A popular approach consists in assuming that the data are generated by a Directed Acyclic Graph (DAG) (Pearl, 2009). DAGs are made of a collection of vertices, which stand for variables, and directed edges to model the dependency structure among the variables, avoiding self-loops and cycles. However, inferring a DAG is a rather challenging

problem. Firstly, the number of nodes p of the graph may be so large that exploring relevant DAG topologies is simply infeasible, since the number of possible DAG structures is super-exponential in p (Robinson, 1973; Koivisto & Sood, 2004; Tsamardinos et al., 2006; Grzegorczyk & Husmeier, 2008). Another dimension flaw occurs when p, even being reasonable, is larger than the number of observations, and parameter estimation is jeopardized. High-dimensional statistical techniques are then needed to overcome this issue (Bühlmann & van de Geer, 2011; Giraud, 2015). Secondly, even if the ratio between p and the sample size n is not impeding model estimation, the nature of the data can be an additional obstacle (Ellis & Wong, 2008; Guyon et al., 2010; Fu & Zhou, 2013). The available observational data are in general not sufficient to identify the true underlying DAG, and can only determine an equivalence class of DAGs (Verma & Pearl, 1991). This approach relies on the assumption that the joint distribution is Markov and faithful with respect to the true graph (Spirtes et al., 2000).

A large number of methods have been proposed for estimating DAGs, including for instance score-based methods (Bayesian score, Friedman & Koller 2003 or Bayesian Information Criterion, Schwarz 1978), complex space sampling (Zhou, 2011) or the PC algorithm (Spirtes et al., 2000). The latter has been proved to be uniformly consistant in the high-dimensional case, but requires a test of conditional independences that quickly becomes computationally intractable (Kalisch & Bühlmann, 2007). Recent works stress the limitations of the absence of cycles in DAGs in the study of complex systems (De Smet & Marchal, 2010). Wright (1921) already described more general directed dependencies between variables when introducing genetic path analysis, but the data were then very limited. Structural Equation Modelling later introduced the notion of noise measurement (Hoyle, 1995) and Pearl (2009) extended them beyond linearity. Moreover, the directed cyclic graph (Spirtes, 1995) framework received little attention as compared to its acyclic counterpart. Finally, the actual discovery of causal cycles requires temporal data, *e.g.* in the context of dynamic Bayesian networks (Perrin et al., 2003; Dondelinger et al., 2013), data which are difficult and very expensive to collect despite efforts in this direction (Sachs et al., 2009).

In this work, we focus on Gaussian structural equation models associated with maximum likelihood estimators (MLE). In the last years, the ℓ_0-regularization of the MLE drew the attention of a large number of works since it leads to infer sparse graphs. In DAGs, a not necessarily topological ordering of the nodes can always be defined according to edge distribution (Kahn, 1962). Identifying this ordering is known to be a challenging problem (Cook, 1985). Additional data, like gene knock-out data or more general perturbations data (Maathuis et al., 2010; Shojaie et al., 2014) can give information in that way. More generally, biological prior knowledge, retrieved from specific data bases, can be used to assist the network reconstruction algorithm (Husmeier & Werhli, 2007), or a partial knowledge of the network can inform the inference process efficiently, *e.g.* in the semi-supervised framework of Mordelet & Vert (2008). For a known order among the variables in the graph, Shojaie & Michailidis (2010) present results for the estimation of high-dimensional graphs based on independent linear regressions using an adaptive Lasso scheme. When the order of the variables is unknown, van de Geer & Bühlmann (2013) studied the convergence of the ℓ_0-penalized likelihood. However, the ℓ_0-regularized approaches (Silander & Myllymäki, 2006; Hauser & Bühlmann, 2012) remain impractical for estimating graphs with more than 20 vertices, either due to an exhaustive exploration of the set of DAGs or overfitting (Chen & Chen, 2008). Quite recently, Aragam et al. (2015) explored a penalized least-squares estimator for a variety of concave penalization terms. They obtain theoretical guarantees of sparsity bounds and consistency results in model selection. Their theoretical results would greatly benefit an implementation of the methods and an empirical study to demonstrate the effectiveness of the approach. The uni-

fying framework for pseudolikelihood-based graphical modelling of Kahre et al. (2015) extends classical regularization methods. The authors obtain theoretical convergence results, and offer an algorithm with an associated implementation. Their simulation results are quite promising, in particular in terms of computational time.

Our objective consists in overcoming this drastic dimensional limitation, and find inference strategies for graphs with up to several hundred nodes. Such strategies must ensure a high level of sparsity, be supported by computationally affordable algorithms, while preserving sound theoretical bases. Here, we propose to use the ℓ_1-regularization, similarly to Fu & Zhou (2013) and Shojaie & Michailidis (2010), to penalize the MLE. From a computational point of view, this regularization makes the criterion to maximize partially convex while ensuring sparse estimates. Our contribution is two-fold: firstly, we provide convergence inequalities that guarantee good theoretical performances of our proposed estimator in the sparse high-dimensional setting. Secondly, we provide an efficient algorithm to infer the true unknown DAG, in the form of a convex program embedded in a genetic algorithm.

The next section covers the model definition and the associated penalized MLE problem. Section 3 details the convergence inequalities, and Section 4 our inference algorithm. Section 5 reports numerical experiments both on toy problems and realistic data sets.

2 The ℓ_1-penalized likelihood for estimating DAGs

2.1 DAG's modelling and estimation

This work considers the framework of an unknown DAG $\mathcal{G}_0 = (V, E)$, consisting of vertices $V = \{1, ..., p\}$ and a set of edges $E \subseteq V \times V$. The p nodes are associated to random variables $X^1, ..., X^p$. A natural approach, developped by Meinshausen & Bühlmann (2006) to solve the network inference problem is to consider that each variable X^i $(1 \leq i \leq p)$ of the DAG can be represented as a linear function of all other variables X^j $(j \neq i)$ through the Gaussian Structural Equation Model:

$$\forall j \in [\![1, p]\!], \quad X^j = \sum_{i=1}^{p} (G_0)_i^j X^i + \varepsilon^j, \tag{1}$$

with $\varepsilon^j \sim \mathcal{N}(0, \sigma_j^2)$ (σ_j^2 known) a Gaussian residual error term. The set of edges E, which is assumed to be of size s ($s \leq p(p-1)/2$), corresponds to the non-zero coefficients of G_0, *i.e.* $(G_0)_i^j$ encodes the relationship from variable X^i to variable X^j.

Assume that we observe an n-sample consisting of n i.i.d. realizations $(X^1, ..., X^p)$ of Equation (1), distributed according to a $\mathcal{N}(0, \Sigma)$ law where Σ is non-singular. We denote by $X := (X^1, ..., X^p)$ the $n \times p$ data matrix. The relations between the variables can then be represented in its matrix form:

$$X = XG_0 + \varepsilon, \tag{2}$$

where $G_0 = ((G_0)_i^j)_{1 \leq i,j \leq p}$ is the $p \times p$ matrix compatible with the graph $\mathcal{G}_0$ and $\varepsilon := (\varepsilon^1, ..., \varepsilon^p)$ is the $n \times p$ matrix of noise vectors.

The negative log-likelihood of the model is then (Rau et al., 2013):

$$\ell(G) = \frac{np}{2} \log(2\pi) + n \sum_{j=1}^{p} \log \sigma_j + \sum_{k=1}^{n} \sum_{j=1}^{p} \frac{1}{\sigma_j^2} \left(X_k (I - G)^j \right)^2, \tag{3}$$

where I denotes the $p \times p$ identity matrix and X_k the vector of length p corresponding to the k-th observation of $X^1, ..., X^p$.

To recover the structure of the DAG $\mathcal{G}_0$ and make the estimated graph sparse enough, we focus on a penalized maximum likelihood procedure (Bickel & Li, 2006):

$$\hat{G} = \underset{G \in \mathcal{G}_{\text{DAG}}}{\operatorname{argmin}} \ \{\ell(G) + \lambda \operatorname{pen}(G)\}, \tag{4}$$

where $\ell(.)$ is the negative log-likelihood of Equation (3), pen(.) is a penalization function, λ is a trade-off

parameter between penalization and fit to the data, and $\mathcal{G}_{\text{DAG}}$ is the set of $p \times p$ matrices compatible with a DAG over p nodes.

In the setting of Gaussian structural equation models with equal noise variance, Peters et al. (2011, 2014) showed that the true DAG was identifiable for respectively discrete and continuous data. In a nutshell, it implies that the true DAG could be inferred, not just the Markov equivalence class of the underlying DAG - a partially directed graph exactly encoding the conditional dependency structure. Using an ℓ_0-norm regularization in Equation (4) is an attractive option to infer sparse graphs. From a computational point of view, the main difficulty when solving the optimization problem in Equation (4) lies in exploring the set of DAGs $\mathcal{G}_{\text{DAG}}$. (Chickering, 1996) showed it to be an NP-hard problem: an ℓ_0-regularization does not set a favorable framework for this task. To avoid the whole exploration of $\mathcal{G}_{\text{DAG}}$, a dynamic programming method has been proposed in Silander & Myllymäki (2006), using a particular decomposition of the ℓ_0-penalized maximum likelihood. The greedy equivalent search algorithm of Chickering (2002) is a hill climbing alternative method. (Hauser & Bühlmann, 2012) rather restricted the search space to the smaller space of equivalence classes, and they provide an efficient algorithm without enumerating all the equivalent DAGs. They showed that they are asymptotically optimal under a faithfulness assumption (*i.e.* independences in the distribution are those read from $\mathcal{G}_0$). However, none of the approaches above can be used on high-dimensional data to estimate graphs with a large number of nodes. In this context, we focus on the ℓ_1-norm convex regularization instead of ℓ_0 for its sparse, high-dimensional and computational properties.

The ℓ_1-regularization clearly improves the computation of (4). It allows us to write a convex formulation of the problem (see Section 2.2). Given Equation (3) and omitting constant terms, the ℓ_1-penalized likelihood estimator we consider is:

$$\hat{G} = \underset{G \in \mathcal{G}_{DAG}}{\operatorname{argmin}} \left\{ \frac{1}{n} \left\| X(I - G) \right\|_F^2 + \lambda \left\| G \right\|_1 \right\}, \quad (5)$$

where, for any matrix $M := (M_i^j)_{1 \leq i,j \leq p}$, we denote by $\|M\|_F = \sum_{i,j} (M_i^j)^2$ the Frobenius norm and by $\|M\|_1 = \sum_{i,j} |M_i^j|$ the ℓ_1-norm.

2.2 A new formulation for the estimator

We propose here a new formulation of the minimization problem of Equation (5). It naturally decouples the initial problem into two steps of the minimisation procedure: node ordering and graph topology search. A key property is that any DAG leads to a topological ordering of its vertices, denoted $\leq$, where a directed path from node X^i to node X^j is equivalent to $X^j \leq X^i$ (Kahn, 1962; Cormen et al., 2001) (see Example 1 below for more explanations). This ordering is not unique in general. Proposition 2.1 from Bühlmann (2013) then gives an equivalent condition for a matrix to be compatible with a DAG.

Proposition 2.1 (Bühlmann, 2013) *A matrix G is compatible with a DAG $\mathcal{G}$ if and only if there exists a permutation matrix P and a strictly lower triangular matrix T such that:*

$$G = PTP^T.$$

Graphically, the permutation matrix sets an ordering of the nodes of the graph and is associated to a complete graph. The strictly lower triangular matrix T sets the graph structure, *i.e.* the non-zero entries of G, as illustrated in Example 1.

Example 1 *Consider the DAG $\mathcal{G}$ given in Figure 1 (left). The corresponding matrix G can then be written as the strictly lower-triangular matrix T by permutation of its rows and columns using P:*

$$G = \begin{pmatrix} 0 & 0 & 0 & 7 & 5 \\ 2 & 0 & 1 & 6 & 4 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} = PTP^T,$$

$$T = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 3 & 0 & 0 & 0 & 0 \\ 5 & 0 & 7 & 0 & 0 \\ 4 & 1 & 6 & 2 & 0 \end{pmatrix} \quad \textit{and} \quad P = \begin{pmatrix} 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

Looking at the non-zero values of P column by column, P defines a node hierarchy $X^5 \leq X^3 \leq X^4 \leq X^1 \leq X^2$ compatible with the topological orderings of $\mathcal{G}$. Graphically, P is associated to the complete graph represented in Figure 1 (bottom). The dashed edges then correspond to the lower zero entries of T. Note that since X^3 is not connected with X^1, X^4 and X^5, four topological ordering are possible ($X^5 \leq X^4 \leq X^1 \leq X^3$, $X^5 \leq X^4 \leq X^3 \leq X^1$, $X^5 \leq X^3 \leq X^4 \leq X^1$ and $X^3 \leq X^5 \leq X^4 \leq X^1$).

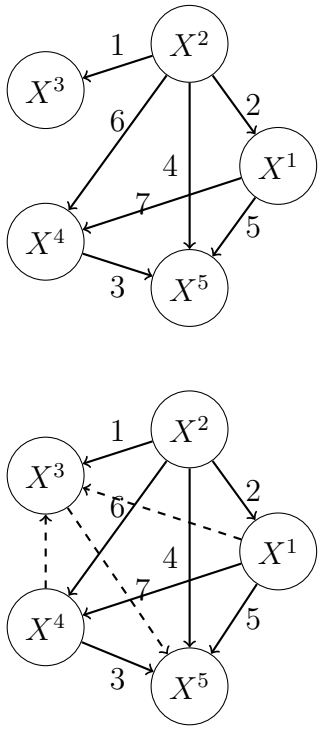


Figure 1: An example of DAG $\mathcal{G}$ (top) and the action of P and T on $\mathcal{G}$: P is associated to a complete graph that orders the nodes of the graph (bottom) and T sets the weights on the edges. The dashed edges correspond to null weight edges (a zero entry in T).

Using Proposition 2.1, the estimator in (5) leads to the following equivalent optimization problem:

$$(\hat{P}, \hat{T}) = \underset{(P,T)\in\mathcal{C}}{\operatorname{argmin}} \left\{ \frac{1}{n} \left\| X(I - PTP^T) \right\|_F^2 + \lambda \left\| T \right\|_1 \right\}, \quad (6)$$

where the optimization space of constraints $\mathcal{C}$ is defined as $\mathcal{C} = \mathbb{P}_p(\mathbb{R}) \times \mathbb{T}_p(\mathbb{R})$, with $\mathbb{P}_p(\mathbb{R})$ the set of permutation matrices and $\mathbb{T}_p(\mathbb{R})$ the set of strictly lower-triangular matrices. Note that a similar formulation has already been proposed by van de Geer & Bühlmann (2013) to ensure good theoretical properties for the ℓ_0-penalized log-likelihood estimation. However, it has never been exploited from a computational point of view to recover the graph structure optimizing problem (5). In the following two sections, we propose a theoretical analysis of the proposed estimator (Section 3) and a computationally effficient algorithm to solve Problem (6) (Section 4).

3 Convergence inequalities for the DAG estimation

The main result of this section deals with convergence rates: in Theorem 1, we provide upper bound for error associated with the ℓ_1-penalized maximum likelihood estimator considered in Equation (6), both in prediction (Equation 7) and estimation (Equation 8). Following the works of van de Geer & Bühlmann (2013) on the ℓ_0-penalized maximum likelihood estimator and of Bickel et al. (2009) on the Lasso and the Dantzig Selector, we obtain two convergence results under some mild sparsity assumptions, when the number of variables is large but upper bounded by a function $\varphi(n)$ of the sample size n.

3.1 Estimating the true order of variables

For a known ordering among the variables of the graph (Shojaie & Michailidis, 2010), an unrealistic

assumption in many applications, the DAG inference problem can be cast in a convex optimization problem. To provide convergence inequalities of the proposed estimator in the most general case of an unknown order we consider here, we first focus on the problem of estimating the true variable order. Let us denote by Π_0 the set of permutation matrices compatible with the true DAG $\mathcal{G}_0$:

$$\Pi_0 = \left\{ P \in \mathbb{P}_p(\mathbb{R}), P^T G_0 P \in \mathbb{T}_p(\mathbb{R}) \right\}.$$

Π_0 contains one or more permutation matrice(s) (see Example 1). We will have to make a decision as to whether the estimated order of variables $\hat{P}$ given by Equation (6) is in Π_0 or not.

To answer this question, we investigate the effect of learning an erroneous order of variables $P \notin \Pi_0$. We introduce the following additional notations: for any permutation matrix $P \in \mathbb{P}_p(\mathbb{R})$, we denote by $G_0(P)$ the matrix defined as:

$$G_0(P) = P T_0 P^T,$$

with $T_0 = P_0^T G_0 P_0$ a lower triangular decomposition of G_0. From a graphical point of view, while $P \notin \Pi_0$, the graph $\mathcal{G}_0(P)$ associated to $G_0(P)$ is obtained from $\mathcal{G}_0$ by permuting some of its nodes (see Example 2), otherwise, if $P \in \Pi_0$, $\mathcal{G}_0(P) = \mathcal{G}_0$. We also denote by $\varepsilon(P) := X - XG_0(P)$ the associated residual term. We denote by $\Omega(P)$ the covariance matrix of $\varepsilon(P)$ and $\omega_j(P) := \mathrm{Var}(\varepsilon^j(P))$ the associated noise variances of each node.

With these notations and checking that the assumptions presented in Section 3.2 hold, we ensure that, with large probability, we choose a right order of variables and the estimated graph converges to the true graph when n and p grow to infinity (see Section 3.3).

Example 2 *Let*

$$P = \begin{pmatrix} 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{pmatrix} \notin \Pi_0$$

a wrong permutation.

In Figure 2, we represent the permuted graph $\mathcal{G}_0(P)$ (bottom) associated to the graph $\mathcal{G}_0$ (top). The latter is obtained from $\mathcal{G}_0$ after permutation of its nodes using PP_0^T, where P_0 (corresponding to the matrix P in Example 1) defines a right order of variables.

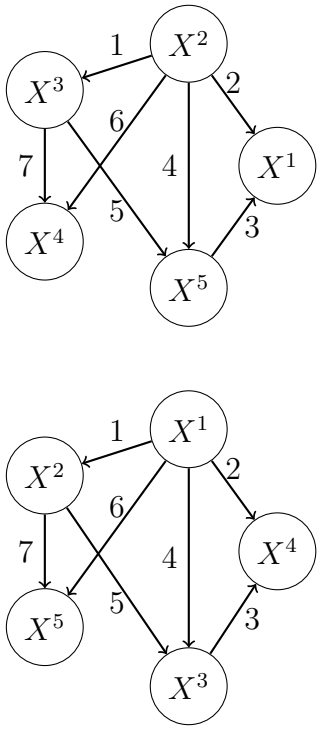


Figure 2: The graph $\mathcal{G}_0$ (top) and the permuted graph $\mathcal{G}_0(P)$ (bottom) associated to the permutation P.

3.2 Assumptions on the model

For a square matrix $M \in \mathcal{M}_{p\times p}(\mathbb{R})$ and a subset $\mathcal{S}$ of $[\![1,p]\!]^2$, we denote by $M_{\mathcal{S}} \in \mathcal{M}_{p\times p}(\mathbb{R})$ the matrix that has the same elements as M on $\mathcal{S}$ and zero on the complementary set $\mathcal{S}^C$ of $\mathcal{S}$. We now introduce the assumptions we used to obtain statistical properties of our estimator.

Hypotheses

$\mathbf{H_1}$ There exists $\sigma^2 > 0$ such that

$$\forall j \in [\![1, p]\!], \mathrm{Var}(\varepsilon^j) = \sigma^2.$$

$\mathbf{H_2}$ There exists σ_0^2, independent of p and n, such that

$$\max_{1 \leq j \leq p} \mathrm{Var}(X^j) \leq \sigma_0^2.$$

$\mathbf{H_3}$ There exists $\lambda^* > 0$ such that the minimal eigenvalue of the covariance matrix Σ of X satisfies

$$\lambda_{min} \geq \lambda^* > 0.$$

$\mathbf{H_4}$ There exists $g_{max} < \infty$ such that the maximal weight of the DAG $\mathcal{G}_0$ is bounded

$$\max_{1 \leq i,j \leq p} \left|(G_0)_i^j\right| \leq g_{max}.$$

$\mathbf{H_5}$ The number of nodes p satisfies

$$p \log p = \mathcal{O}(n).$$

$\mathbf{H_6}$ There exists $\kappa(t) > 0$ with $1 \leq t \leq p^2$ such that:

$$\min \left\{ \frac{\|XM\|_F}{\sqrt{n}\,\|M_{\mathcal{S}}\|_F} \right\} \geq \kappa(t),$$

where the minimum is taken over the set of $p \times p$ matrices satisfying $\|M_{\mathcal{S}^C}\|_1 \leq 3\,\|M_{\mathcal{S}}\|_1$, with $\mathcal{S} \subset [\![1, p]\!]^2$ and $|\mathcal{S}| \leq t$.

$\mathbf{H_7}$ There exists $0 < \eta \leq C \frac{n}{p \log p} \times \frac{1}{\sqrt{s}}$ such that, for all permutations $P \notin \Pi_0$,

$$\frac{1}{p} \sum_{j=1}^{p} \left(|\omega_j(P)|^2 - 1 \right)^2 > \frac{1}{\eta}.$$

Assumption **$\mathbf{H_1}$** states that the noise variances are the same among all variables. This assumption is clearly hard to test in practice but makes the problem identifiable and ensures that we can recover the true DAG. Otherwise, minimizing (5) only leads to the identification of one element of the Markov equivalence class of the true DAG (partially directed graph). To simplify the theoretical results and proofs, until the end of this work, we assume that the noise variances σ^2 are equal to one. Our results are still valid even if $\sigma^2 \neq 1$, by small modifications in the constant terms as long as they are all equal.

Assumption **$\mathbf{H_5}$** deserves a special attention since it bounds the high dimensional setting. The considered problem is obviously non-trivial and requires a sufficient amount of information. A more detailled discussion about assumptions **$\mathbf{H_3}$** and **$\mathbf{H_5}$** is proposed in Section 3.4.

Assumption **$\mathbf{H_6}$** is a natural extension of the Restricted Eigenvalue condition of Bickel et al. (2009) to our multi-task setting. More precisely, denoting

$$\tilde{X} = \underbrace{\begin{pmatrix} X & & 0 \\ & \ddots & \\ 0 & & X \end{pmatrix}}_{p^2} \Bigg\updownarrow n \times p,$$

$\mathbf{H_6}$ is equivalent to assuming that the Gram matrix $\frac{\tilde{X}\tilde{X}^T}{n}$ is non-degenerate on a restricted cone (Lounici et al., 2009; Bühlmann & van de Geer, 2011). Notice that this condition is very classical in the literature. It yields good practical performance even for small sample sizes, and some recent works discuss an accurate population eigenvalue estimation even in a large dimension setting (Mestre, 2008; El Karoui, 2008; Liu et al., 2014; Ledoit & Wolf, 2015).

The last assumption **$\mathbf{H_7}$** is an identifiability condition needed to ensure that the estimated permutation $\hat{P}$ is in Π_0. This assumption was introduced by van de Geer & Bühlmann (2013) as the "omega-min" condition. In a sense, it separates the set of compatible permutations from its complement in a finite sample scenario.

3.3 Main result

The result we establish in this section is double-edged: (a) with large probability, we ensure that the estimated $\hat{P}$ belongs to Π_0, and (b) we provide convergence inequalities both in prediction and estimation for the graph estimated from the minimisation problem (6). This result clearly states the desirable

theoretical properties of the derived estimator, assuming reasonable conditions on the complex system embedding the data.

Theorem 1 *Assume that* $\mathbf{H_{1-7}}$ *are satisfied, with* $s \subset [\![1, p^2]\!]$ *in* $\mathbf{H_6}$ *such that* $\sum_{i,j} \mathbb{1}_{(G_0)_i^j \neq 0} \leq s$ *(G_0 is s-sparse). Let* $\lambda = 2C\sqrt{s^{1/2}\frac{\log p}{n}}$. *Then, with probability greater than* $1 - 5/p$, *any solution* $\hat{G} = \hat{P}\hat{T}\hat{P}^T$ *of the minimization problem (6) satisfies that* $\hat{P} \in \Pi_0$. *Moreover, with at least the same probability, the following inequalities hold:*

$$\frac{1}{n}\left\|X\hat{G} - XG_0\right\|_F^2 \leq \frac{16C^2}{\kappa^2(s)} s^{3/2}\frac{\log p}{n}. \tag{7}$$

$$\left\|\hat{G} - G_0\right\|_1 \leq \frac{16C}{\kappa^2(s)}\sqrt{s^{5/2}\frac{\log p}{n}}. \tag{8}$$

The proof of this result is deferred in Section C of the Supplementary Materials.

Theorem 1 states that with probability at least $1 - 5/p$, we choose a compatible order of variables over the set of permutations. Inequalities (7) and (8) give non-asymptotic upper bounds on the loss under conditions depending on s, p and n (see Section 3.4). They also ensure that the estimated $\hat{T}$ is close to the true T_0 with large probability.

3.4 Discussion on the high-dimensional scenario

Sparsity of the graph Assumption $\mathbf{H_7}$ and Theorem 1 naturally require a trade-off between signal sparsity, dimensionality and sample size. In the ultra sparse regime (where the sparsity s of the true graph is bounded by $s^* > 0$), Theorem 1 provides convergence inequalities for $\hat{G}$ choosing $\eta \leq \frac{\alpha}{\sqrt{s^*}}$ with $p\log(p) = \alpha n$ in Assumption $\mathbf{H}_7$.

In the standard sparsity scenario, if s is at least of the order of p, then η should be of the order of $\alpha/\sqrt{p}$, which is unrealistic as $p \to +\infty$. This case thus requires a stronger dimensional assumption $\mathbf{H}_5$. Taking at least $p^2\log(p) = \mathcal{O}(n)$ ensures a good estimation of the graph.

Note however that universal conditions cannot be overcome and the ultra-high dimension settings (*e.g.* Wainwright (2009); Verzelen (2012)) is an insurmountable limit.

Minimal eigenvalue condition Assumption $\mathbf{H_3}$ ensures that the minimal eigenvalue of the covariance matrix Σ of X is not too small. In the high-dimensional scenario, this could be hard to verify, λ_{min} decreasing while n, p growing to infinity (Hogben, 2007). A natural bound for λ_{min} is:

$$\lambda_{min} \geq \frac{1}{p\max\left(1, g_{max}^2\right)\left(1 + \sqrt{s}\right)}, \tag{9}$$

with g_{max} and s as in $\mathbf{H_4}$ and Theorem 1.

Assumption $\mathbf{H_3}$ can thus be relaxed by allowing λ_{min} to decrease with $1/p\sqrt{s}$. The price to pay for this relaxation is a data dimensionality reduction $p^3\log(p) = \mathcal{O}(n)$, which automatically implies:

$$\frac{3\lambda_{min}}{4} - 2\sqrt{\frac{\log(p)}{n}} - 3\sigma_0\sqrt{\frac{2p\log(p)}{n}} > 0,$$

with Equation (9) (for more details, see Section A of the Supplementary Material *Proof details*).

4 Inference algorithm

4.1 Global algorithm overview

In this section, we present GADAG (*Genetic Algorithm for learning Directed Acyclic Graphs*), a computational procedure devoted to solve Equation (6) and available as a `R` package on CRAN at `https://cran.r-project.org/package=GADAG`. Although decomposing the original problem made it more natural to handle, this problem is still a very challenging task from an optimization point of view, due to the different nature of the variables P and T, the non-convexity of the cost function and the high dimension of the search space.

An intuitive approach consists in using an alternating scheme: one of the variables P or T is fixed and the other one is sought so as to optimize the

score function, then the roles of P and T are reversed and the procedure is repeated iteratively until convergence for some criterion (Csiszár & Tusnády, 1984). However, the structure of our problem does not allow us to use such a scheme: looking for an optimal T given a fixed P makes sense, but changing P for a fixed T does not.

In our inference algorithm GADAG, an outer loop is used to perform the global search among the DAGs space, which is driven by the choice of P, while a nested loop is used to find an optimal T for each given fixed P (see Figure 3). As we show in the following, population-based metaheuristics algorithms are a natural and efficient choice for exploring the space of permutation matrices (Section 4.3). The nested optimization problem can be resolved using a steepest descent approach (Section 4.2).

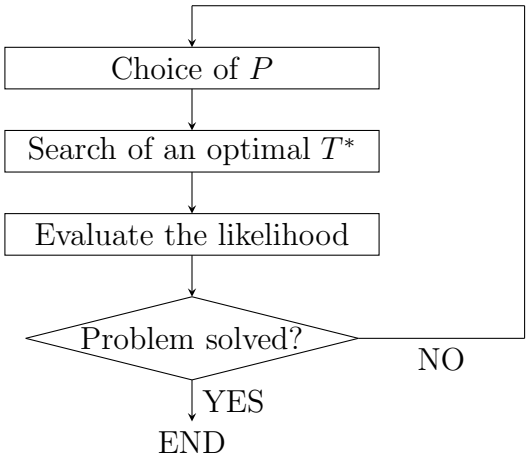


Figure 3: Overview of our hybrid algorithm GADAG.

4.2 Graph structure learning when the variable order is fixed

Assume first that the variable ordering $P \in \mathbb{P}_p(\mathbb{R})$ is fixed. The problem of inferring a graph is then reduced to estimating the graph structure, which can be solved by finding a solution of:

$$\min_{T \in \mathbb{T}_p(\mathbb{R})} \left\{ \frac{1}{n} \left\| X(I - PTP^T) \right\|_F^2 + \lambda \left\| T \right\|_1 \right\}. \quad (10)$$

The minimization problem given by Equation (10) looks like a well-studied problem in machine learning, as it is closely related to the ℓ_1-constrained quadratic program, known as the Lasso in the statistics literature (Tibshirani, 1996). Indeed, the ℓ_1-regularization leads to variable selection and convex constraints that make the optimization problem solvable. We note here that this allows us to always provide a locally optimal solution, *i.e.* optimal weight estimates given a hierarchy between the nodes.

A large number of efficient algorithms are available for computing the entire path of solutions as λ is varied, *e.g.* the LARS algorithm of Efron et al. (2004) and its alternatives. For example, in the context of the estimation of sparse undirected graphical models, Meinshausen & Bühlmann (2006) fit a Lasso model to each variable, using the others as predictors, and define some rules for model symmetrization as they do not work on DAGs. The graphical Lasso (or glasso, Friedman et al. 2007) algorithm directly relies on the estimation of the inverse of a structure covariance matrix assumed to be sparse. Improvements were proposed for example by Duchi et al. (2008) (improved stopping criterion) and Witten et al. (2011) (estimation of a block-diagonal matrix). Other authors propose to solve the optimization problem using an adaptation of classical optimization methods, such as interior point (Yuan & Lin, 2007) or block coordinate descent methods (Banerjee et al., 2008; Friedman et al., 2007).

We propose here an original convex optimization algorithm to find the solution in Equation (10) in a form similar to a steepest descent algorithm. Our proposed algorithm is much quicker than a glasso approach, a desirable feature as it will run at each iteration of the global algorithm (see the "Search of an optimal T^*" box in Figure 3 and the "Evaluate the new individuals" item in Algorithm 2). Moreover, its mechanistic components (see Section B of the Supplementary Material *Proof details*) allowed us to derive the theoretical results of Theorem 1. The proposed scheme can be seen as an adaptation

of the LARS algorithm with matrix arguments.

Let $(T_k)_{k\geq 0}$ the sequence of matrices defined for all $i,j \in [\![1,p]\!]^2$ as:

$$(T_{k+1})_i^j = \text{sign}\left((U_k)_i^j\right) \max\left(0, \left|(U_k)_i^j\right| - \frac{\lambda}{L}\right), \quad (11)$$

where for all $k \geq 0, U_k = T_k - \frac{\nabla\left(\frac{1}{n}\left\|X(I-PT_kP^T)\right\|_F^2\right)}{L}$, L is the Lipschitz constant of the gradient function $\nabla\left(\frac{1}{n}\left\|X(I-PT_kP^T)\right\|_F^2\right)$ and sign() is the sign of any element. Then, a solution of (10) is given by performing Algorithm 1, where:

- the projection $\text{Proj}_{\mathbb{T}_p(\mathbb{R})}(T)$ of any $p \times p$ real-valued matrix $T = ((T_k)_i^j)_{i,j}$ on the set $\mathbb{T}_p(\mathbb{R})$ is given by

$$\left(\text{Proj}_{\mathbb{T}_p(\mathbb{R})}(T_k)\right)_i^j = \begin{cases} 0 & \text{if } i < j, \\ (T_k)_i^j & \text{otherwise.} \end{cases} \quad (12)$$

- the gradient of $\frac{1}{n}\left\|X(I-PT_kP^T)\right\|_F^2$ is

$$\nabla\left(\frac{1}{n}\left\|X(I-PT_kP^T)\right\|_F^2\right) = -\frac{2}{n}(XP)^T(X-XPT_kP^T)P. \quad (13)$$

The detailed calculations are deferred to Section B of the Supplementary Material *Proof details*.

Algorithm 1: Graph structure learning - minimization of Equation (10)

Input: $\lambda, L, \epsilon > 0$.
Initialization: T_0 the null squared $p \times p$ matrix, $k = 0$ and $e = +\infty$.
while $e > \epsilon$ **do**
  Compute $U_k = T_k - \frac{\nabla\left(\frac{1}{n}\left\|X(I-PT_kP^T)\right\|_F^2\right)}{L}$ with Equation (13);
  Using Equation (11), compute the current matrix $T_{k+1} = \left((T_{k+1})_i^j\right)_{i,j}$;
  Project T_{k+1} on $\mathbb{T}_p(\mathbb{R})$ with Equation (12): $T_{k+1} \leftarrow \text{Proj}_{\mathbb{T}_p(\mathbb{R})}(T_{k+1})$;
  Compute $e = \|T_{k+1} - T_k\|_F$;
  Increase k: $k \leftarrow k+1$;
end
Output: $T_k \in \mathbb{T}_p(\mathbb{R})$ the unique solution of (10).

4.3 A Genetic Algorithm for a global exploration of the permutation matrices space incorporating network topologies

As the optimal T can be calculated for any P using Algorithm 1 and with a very good approximation accuracy according to Theorem 1, the optimization task (6) comes down to exploring the $\mathbb{P}_p(\mathbb{R})$ space of permutation matrices in dimension p and to evaluating the quality of permutation candidates $P \in \mathbb{P}_p(\mathbb{R})$. We first note that the number of permutation matrices is $p!$, which rules out any exact enumeration method, even for relatively small p. We propose instead to use a meta-heuristic approach, which has proven to be successful for many discrete optimization problems like wire-routing, transportation problems or traveling salesman problem (Michalewicz, 1994; Dréo et al., 2006).

Among the different meta-heuristics (Simulated annealing, Tabu search, Ant Colony,...) we focused on Genetic Algorithms (GA) because, despite limited convergence results (Cerf, 1998; Michalewicz, 1994), they were found much more efficient in problems related to ours than alternatives with more established convergence proofs (*e.g.* Granville et al. (1994) for simulated annealing), while allowing the use of parallel computation.

GAs mimic the process of natural evolution, and use a vocabulary derived from natural genetics: populations (a set of potential solutions of the optimization problem), individuals (a particular solution) and genes (the components of a potential solution). Each generation/iteration of the algorithm will improve the constituting elements of the population. In short, a population made of N potential solutions of the optimization problem samples the

search space. This population is sequentially modified, with the aim of achieving a balance between exploiting the best solutions and exploring the search space, until some termination condition is met.

We use here a classical Genetic Algorithm, as described in Michalewicz (1994) for instance, which is based on three main operators at each iteration: selection, crossover and mutation. The population is reduced by selection; selection shrinks the population diversity based on the individual fitness values. The crossover allows the mixing of good properties of the population to create new composite individuals. Mutations change one (or a few in more general GAs) components of the individuals to allow random space exploration. The complete sketch of algorithm GADAG is given in Algorithm 2. A discussion on parameters to set in Algorithm 2 is found in Section 5.1. The details of the different operators are given in the following.

As we show in Example 3, any $P \in \mathbb{P}_p(\mathbb{R})$ is uniquely defined by a permutation vector of $[\![1, p]\!]$. Hence, we use as a the search space $\mathfrak{S}_p$ the set of permutations of $[\![1, p]\!]$, which is a well-suited formulation for GAs.

Example 3 *Consider the permutation matrix ($p = 5$):*

$$P = \begin{pmatrix} 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

Then, P is represented by the $\boxed{5}\boxed{3}\boxed{4}\boxed{1}\boxed{2}$ *vector, looking at the ranks of non-null values of P column by column. The nodes are ranked according to their topological ordering.*

Note that our problem closely resembles the classical Traveling Salesman Problem (TSP), which has been succesfully addressed by means of genetic algorithms (Grefenstette et al., 1985; Davis, 1991). Identically to the TSP, we optimize over the space of permutations, which induces specific constraints for defining the crossover and mutation operators. However, unlike the TSP, the problem is not circular (in the TSP, the last city is connected to the first one), and the permutation here defines a hierarchy between nodes rather than a path, which makes the use of TSP-designed operators a potentially poor solution. As we show in the following, we carefully chose these operators in order to respect the nature of the problem at hand. In particular, we emphasize two of their desirable features: their efficiency in exploring the search space and the interpretable aspect they offer in terms of modifications on a given network or the blend of two different networks (crossover).

Fitness function Given a potential solution $p_i \in \mathfrak{S}_p$, the fitness function is defined as:

$$J_i = J(p_i) = \frac{1}{n} \left\| X(I - P_i T_i^* P_i^T) \right\|_F^2 + \lambda \left\| T_i^* \right\|_1, \tag{14}$$

with P_i constructed from p_i as in Example 3 and T_i^* the solution of Equation (10) with $P = P_i$. As mentioned earlier, at each step of the proposed GA, the evaluation of the fitness function thus requires running the nested loop of our global algorithm GADAG.

Selection operator The selection operator (or survival step) consists in generating a population of N individuals from the N existing individuals by random sampling (with replacement, hence some individuals are duplicated and others are deleted). It aims at improving the average quality of the population by giving to the best potential solutions a higher probability to be copied in the intermediate population. We have chosen to use the classical proportional selection of Holland (1975): each individuals is selected with a probability inversely proportional to its fitness value of Equation (14).

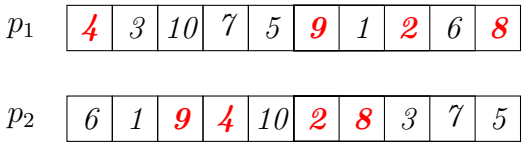
Crossover operator A crossover operator generates a new set of potential solutions (children) from existing solutions (parents). Crossover aims at achieving at the same time (i) a good exploration of

the search space by mixing the characteristics of the parents to create potentially new ones while (ii) preserving some of the desirable characteristics of the parents. By desirable features, we mean features of the network which lead to good fitness values, and which in turn are favored by selection over the generations. The crossover population (set of parents) is obtained by selecting each individual of the population with a probability p_{xo}; the parents are then paired randomly.

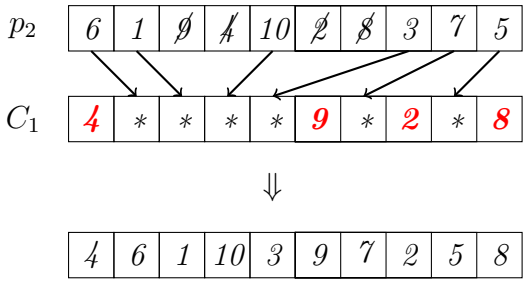
We have chosen the *order-based* crossover, originally proposed for the TSP (Michalewicz, 1994, Chapter 10), which is defined as follows. Given two parents p_1 and p_2, a random set of crossover points are selected, which we denote Ω. It consists in a permutation of k elements taken from $[\![1, p]\!]$, with k uniformly drawn between 0 and p. A first child C_1 between p_1 and p_2 is then generated by:

1. fixing the crossover points of p_1,
2. completing C_1 with the missing numbers in the order they appear in p_2.

Example 4 *Consider the two following parents:*



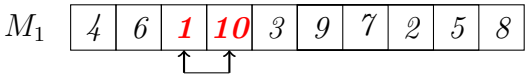
Assume that the crossover points randomly chosen are 4, 9, 2 *and* 8 *(in bold red above). Then, the child* C_1 *is defined by inheriting those points from* p_1 *and filling the other points in the order they appear in* p_2*:*



From a graphical point of view, a crossover between p_1 and p_2, which encode two complete graphs $\mathcal{G}_{P_1}$ and $\mathcal{G}_{P_2}$, constructs two new graphs. One of them, $\mathcal{G}_{C_1}$ is composed of the sub-graph of $\mathcal{G}_{P_1}$ induced by the set of crossover points Ω and the sub-graph of $\mathcal{G}_{P_1}$ induced by the complementary set Ω^C of Ω in $[\![1, p]\!]$ (see Figure 4). The second child graph $\mathcal{G}_{C_2}$ is obtained in an identical manner by reversing the roles played by the two parental graphs.

Mutation operator Mutation operators usually correspond to the smallest possible change in an individual (unary operator). We thus define it as an alteration of two neighbouring genes (see Example 5). Graphically, a mutation consists in switching the arrowhead of an edge between two nodes. Mutation is applied to each child with probability p_m.

Example 5 *A possible mutation for the first child of Example 4 is to swap the genes "1" and "10" (in bold red below):*



Stopping criterion Two quantities are monitored along the iterations: the heterogeneity of the population and the value of the objective function.

For the first indicator, we use the Shannon entropy, defined for each rank position $j \in [\![1, p]\!]$ as:

$$H_j = -\sum_{i=1}^{p} \frac{N_{i,j}}{N} \log\left(\frac{N_{i,j}}{N}\right),$$

where $N_{i,j}$ is the number of times when i appears in position j. $H_j = 0$ if all the individuals "agree" on the position of a node and the population is perfectly homogeneous at this node. On the contrary, it is maximum when we observe a uniform distribution of the different nodes at a given position and the population is in this case at a maximum of heterogeneity or disorder for this position. The algorithm stops if the population entropy value $H = \sum_{j=1}^{N} H_j$ drops below a threshold since $H = 0$ if all the individuals are identical. A second criterion can terminate GADAG if difference in the average fitness

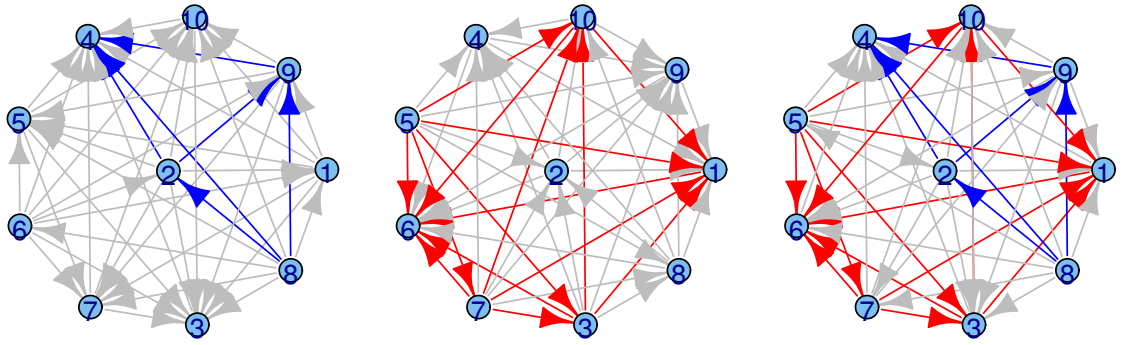


Figure 4: Graphical representation of crossover between two 10-node graphs. The two parental graphs are represented on the left. The third graph, on the right, is obtained by combining the blue and red part of its parents using the crossover operator.

(denoted $\bar{J}$ thereafter) of the population between a given number of consecutive iterations, does not change by more than a predefined threshold.

Algorithm 2: GADAG overview

Input: $p_{xo}, p_m, \epsilon_H > 0, \epsilon_J > 0$, $k_{max} > 0$, $i_{max} > 0$, λ, L.

Initialization: Generate the initial population $\mathcal{P}_0$ with N permutations of $[\![1, p]\!]$, $k = 0$ and $e_J = +\infty$.

while $H > \epsilon_H \quad \& \quad e_J > \epsilon_J \& \quad k \leq k_{max}$ **do**

- Generate $\mathcal{P}_{k+1}$ as a random **selection** of N individuals from $\mathcal{P}_k$;
- Pick an even subset $\mathcal{P}_{xo}$ of $\mathcal{P}_{k+1}$ (each individual of $\mathcal{P}_{k+1}$ selected with probability p_{xo});
- Perform **crossover** on $\mathcal{P}_{xo}$ by randomly pairing the individuals;
- **Mutate** each obtained individual with probability p_m ;
- **Evaluate** the new individuals $\mathcal{P}_m$ by running Algorithm 1;
- Replace $\mathcal{P}_{xo}$ by $\mathcal{P}_m$ in $\mathcal{P}_{k+1}$;
- Compute the Shannon entropy H and the difference in the average fitness $e_J = \max_{0 \leq i \leq i_{max}} \left(\bar{J}(\mathcal{P}_{k+1}) - \bar{J}(\mathcal{P}_{k-i}) \right)$;
- Increase k: $k \leftarrow k + 1$;

end

5 Numerical experiments

This section is dedicated to experimental studies to assess practical performances of our method through two kinds of datasets. In a first phase, the aim of these applications is to show that GADAG has a sound behavior on simulated toy data with a variety of different settings. In a second phase, we demonstrate the ability of our algorithm to analyse datasets that mimic the activity of a complex biological system, and we compare it to other state-of-the art methods. The competing methods are presented in Section 5.4.1. In Section 5.1, we present the calibration of the Genetic Algorithm parameters. Section 5.2 introduces the measures we used to assess the merits of the methods. Experimental results are then detailed in Section 5.3 for the simulated high-dimensional toy datasets and in Section 5.4.2 for the dataset with features encountered in real situations.

All experiments have been performed on `R` (R Core Team, 2017) using the package `GADAG` (Champion et al., 2017). The computational times reported in Section 5.3 correspond to a Windows 7 laptop computer with 8 threads on a 4-core hyper-threaded 2.50GHz processor, with 4GB of RAM.

5.1 Algorithm parameters

Running the procedure of Algorithm 2 requires to define parameters of the outer loop, which generates our population of P's, and of the nested loop to find the optimal T^*. The evaluation of the Lipschitz gradient constant L, used to find the optimal graph structure T^*, is known as a hard established problem in optimization. Some authors propose to choose an estimate of L from a set of possible values (Jones et al., 1993; Sergeyev & Kvasov, 2006), to estimate local Lipschitz constants (Sergeyev, 1995), or to set it a priori to a fixed value (Evtushenko et al., 2009; Horst & Pardalos, 1995). Here, observing Equation (13), a major bound for L is given by:

$$L \leq \frac{2}{n} \left\| X^T X \right\|_F .$$

We found that setting L to this bound worked well in practice in all our scenarios.

Five parameters need to be tuned to run the Genetic Algorithm: the crossover rate p_{xo}, the mutation rate p_m, the constant of the stopping criteria ϵ_H and ϵ_J and the size of the population N. For the first four parameters, we observed that their value had a limited effect on the efficiency, hence we chose commonly used values in the literature (see Table 1). The size of the population has a more complex effect and has been investigated in several prospective papers (*e.g.* Schaffer et al. 1989; Alander 1992; Piszcz & Soule 2006; Ridge 2007) but without providing a definitive answer to the problem. In our simulation study, we chose as a rule-of thumb $N = 5p$, which was found as a good compromise between computational cost and space exploration on several experiments.

The complete parameter settings used in our experiments are reported in Table 1.

Table 1: Algorithm parameter settings

Parameter	Value
p_{xo}	0.25
p_m	0.5
N	$5 \times p$
L	$\frac{2}{n} \left\| X^T X \right\|_F$
max. nb. of eval.	10^4
ϵ_H	10^{-6}
ϵ_J	10^{-4}

5.2 Performance metrics

A classical performance measure for graph inference methods consists in comparing predicted interactions with the known edges in the true graph $\mathcal{G}_0$ using precision versus recall (P/R) curves. We denote by TP, FP, FN and TN, the true positive (correctly predicted) edges, the false positive (inferred by mistake) edges, the false negative (missed) edges, and the true negative (correctly non-predicted) edges. The recall, defined as $\frac{TP}{TP+FN}$, measures the power (or sensitivity) of reconstruction of non-zero elements of the true matrix G (or equivalently of the true network) for one method, whereas the precision, equal to $\frac{TP}{TP+FP}$, measures the accuracy of the reconstruction. The closer to one the precision and the recall the better.

P/R curves represent the evolution of those quantities when varying the sparsity of the methods. GADAG is based on penalized optimization: it seeks linear dependencies between the variables with a controlled level of parsimony (λ in Equation (5)). For λ varying from 0 (complete graph) to $+\infty$ (empty graph), it thus produces a list of edges successively introduced in the model. This list of edges defines the precision versus recall curve. As a summary performance measurement, we also computed the classical area under the P/R curve (AUPR).

5.3 Exploratory analysis on toy datasets

We first considered simulated data from randomly generated networks with different characteristics in order to assess the capabilities and limits of our algorithm. Given a number of nodes p, a random set of s edges were generated, and the non-zero param-

eters of the matrix G_0 associated to the corresponding DAG were uniformly sampled between 0 and 1. Using this graph, we generated N observations following the hypotheses of Gaussian, homoscedastic and centred error. We then ran GADAG on this dataset to recover the graph. Note that other assumptions presented in Section 3.2 may not be fulfilled here, but we aimed at evaluating the robustness of GADAG for recovering DAGs in such scenario.

In our experiments, we varied the number of nodes p, of edges s and of available observations n. We chose four different settings $p = 50, 100, 500$ and $1,000$ with n/p varying from 100% to 10% and s/p from 100% to 400%. Unless otherwise stated, all experiments were replicated 50 times each and results were averaged over these replicates. Averaged computational times correspond to one iteration of GADAG, for a fixed parameter of penalization λ.

Results, in terms of area under the P/R curves and computational time are sumarized in Table 2. We can first remark a crude decrease of performance results when the number of samples is very small ($p = 50$ and 100, $n/p = 10\%$, so respectively 5 and 10 samples). In that case, GADAG is incapable of recovering any signal (AUPR $< 10\%$). When the sample size is of the order of p ($n/p = 100\%$, first row of Table 2 a), GADAG works well, although it is clearly a favorable case, far from the high-dimensional one. With half of the samples ($n/p = 50\%$), performance remains satisfactory (AUPR around 50%, or more), which is critical since this situation corresponds to realistic biological studies, where subsets of genes (*i.e.* nodes) are preselected beforehand. Interestingly, $p = 500$ and $1,000$ work better than smaller values of p since the number of samples is larger to estimate the graph that generated the data. Indeed, for a given number of samples, *e.g.* $n = 50$, performance results slightly decrease from 65% ($p = 50$) and 55% ($p = 100$) to 50% ($p = 500$). GADAG is thus not considerably affected by a relative increase of dimensionality. For large graphs ($p = 500$ and $1,000$), even if $n/p \leq 10\%$, it suceeds in recovering them, which makes it a competitive algorithm with regards to other state-of-the-art approaches.

An interesting remark is that the number of edges s does not significantly change numerical results (see each row of Table 2 a), although GADAG succeeds slightly better in estimating sparser graphs. This may be due to the particular structure of our algorithm, which looks for topological ordering between nodes (genetic algorithm) and then makes the inferred graph sparse.

Concerning computational time, we can finally note that growing the dimension p clearly makes the problem harder to solve : each call to GADAG requires more than $300s$ for hundred of nodes and $1,500s$ for thousand of nodes.

5.4 DREAM data analysis

The second type of datasets we used mimic activations and regulations that occur in gene regulatory networks. It is provided by the DREAM4 challenge on "In Silico Network Challenge". Note that although plausibly simulated, DREAM4 data sets are not real biological data sets. However, the used network structures (five in total) were extracted from *E. coli* and *S. cerevisae* -two biological model organisms- trancriptional networks. These networks contain cycles, but self-loops were discarded. The gene expression observations were not simulated by an equal noise Gaussian multivariate model, stochastic differential equations were used to mimic the kinetic laws of intricate and intertwined gene regulations. In addition to the biological noise simulated from the stochastic differential equations, technical noises were added to reproduce actual gene measurement noise. All data sets were generated by the `GNW` software (Marbach et al., 2009).

Working with simulated networks, we are able to quantitatively and objectively assess the merit of competing methods in terms of true predictions (true positive TP and true negative TN) vs. incorrect predictions (false positive FP and false negative FN) edges. While the analysis of a real data set is certainly the final goal of a methodology motivated by a real problem like ours, there are only imprecise ways of validating a method when analysing a real data

Table 2: Performance results of GADAG on a toy dataset with different characteristics (number of nodes p, number of edges s and sample size n) in terms of area under the Precision vs. Recall curve (a) and computational time, in seconds (b). All results are averaged over 50 replicates (* 5 replicates only as the running time was 1 day per network).
(a)

		p=50			p=100			p=500			p=1,000*		
	s/p	100%	200%	400%	100%	200%	400%	100%	200%	400%	100%	200%	400%
	100%	0.72	0.65	0.65	0.69	0.63	0.64	0.79	0.79	0.70	0.86	0.80	0.65
n/p	50%	0.53	0.46	0.49	0.56	0.53	0.53	0.75	0.76	0.66	0.82	0.75	0.63
	10%	0.02	0.03	0.05	0.04	0.06	0.09	0.51	0.53	0.41	0.65	0.60	0.49

(b)

		p=50			p=100			p=500			p=1,000*		
	s/p	100%	200%	400%	100%	200%	400%	100%	200%	400%	100%	200%	400%
	100%	0.55	0.43	0.41	8.69	9.87	7.51	253	258	214	1,640	1,500	1,230
n/p	50%	0.48	0.48	0.41	8.63	8.90	6.62	250	258	172	1,520	1,590	1,270
	10%	0.31	0.30	0.36	6.52	5.81	5.41	183	183	165	1,590	1,550	1,290

set. Well known systems are often small and even if knowledge has accumulated on them, these can be noisy and difficult to gather to obtain a fair picture of what can adequately be considered as sets of true positive and true negative sets of edges. Even if the data generation process of the DREAM4 In Silico Network Challenge is completely understood, no existing method is able to predict all regulatory relationships, but at the price of including many false positive predictions. The DREAM4 datasets we considered have $p = 100$ nodes and only $n = 100$ observations making it a a very challenging task.

5.4.1 Comparison to state-of-the art

We compare GADAG to five state-of-the-art inference methods. Among them, the Genie3 method (Huynh-Thu et al., 2010), based on random forests, was the best performer of one of the DREAM4 sub-challenges, while the BootLasso (Allouche et al., 2013) was one of the key components of the best performing approach of one of the DREAM5 sub-challenges (Allouche et al., 2013). The two methods decompose the prediction of the network into p feature selection sub-problems. In each of the p sub-problems, one of the node is predicted from the other ones using random forests (Breiman, 2001) for Genie3 or a bootstrapped version of the Lasso (Bach, 2008) for BootLasso. For the random forest approach, parents of each node were detected as most significant explanatory variables according to a variance reduction criterion in a regression tree framework. The process was repeated on a randomized set of trees, which made up the so-called random forest. This method allowed us to derive a ranking of the importance of all variables for the target by averaging the scores over all the trees. We used the R package `randomforest` (Liaw & Wiener, 2002) for our results. The Lasso is a ℓ_1-norm penalization technique for solving linear regression. Following the works of Bach (2008), BootLasso uses

bootstrapped estimates of the active regression set based on a Lasso penalty: only those variables that are selected in every bootstrap are kept in the model. In both cases, actual coefficient values are estimated from a straightforward least square procedure. Note that we slightly relax the condition for a variable to be included in the model, a variable was selected at a given penalty level if more than 80% of bootstrapped samples led to selecting it in the model (Allouche et al., 2013).

We also compare our algorithm to three classical methods for Bayesian Networks (BNs) modelling. BNs are graphical models (Pearl, 2009) defined by a DAG and parameters that set quantitative relationships between variables. Algorithms devoted to structure and parameter learning in BNs either aim at maximising a score that reflects the fit of the data to the learnt structure, or test for independencies between variables. They are often used as references in a gene regulatory network inference context (Tsamardinos et al., 2006), athough mainly for moderate size networks. The first compared algorithm we used is the PC-algorithm (Spirtes et al., 2000), a popular constraint-based method that drastically reduces the number of conditional independence tests. It first builds the skeleton of the graph by removing edges from a complete undirected graph before determining the orientation of the edges, when possible. A large number of implementations of the PC-algorithm exists. The numerical results presented here were obtained using the pcAlg function of the R-package `pcalg`, based on standard correlation estimates for conditional indepence testing. We also ran ARACNE (Margolin et al., 2006), an improved version of minimum-weight spanning tree that uses the information inequality to eliminate the majority of indirect relationships between variables. We used the ARACNE function of the R-package `bnlearn`. We finally compare GADAG to the Greedy Equivalence Search (GSE) algorithm (Chickering, 2002), implemented in the R-package `pacalg`, which heuristically searches in the space of equivalent classes the model with the highest Bayesian score.

To compare our algorithm with these competing methods, we used the P/R curves presented in Section 5.2. As GADAG, BootLasso leads to a sparse inferred graph while controlling the level of parsimony, which builds the P/R curve. Genie3 produces as an output a ranked list of regulatory interactions, which corresponds to the edges of the inferred graph. Edges are the successively introduced with decreasing confidence scores to produce the random forest P/R curve. For the PC and the GSE algorithms, inherent parameters regulating the sparsity of the produced graphs helped us to define such curves. Note that the implementation we used for running ARACNE was only able to produce a final network prediction (interaction ranking is not available).

5.4.2 Numerical results

The P/R curves for the five DREAM problems are shown in Figure 5. Each curve corresponds to one of the five networks used in the challenge. In general, for all the problems the five methods are able to achieve a precision equal to one (that is, to include only true edges), but these correspond to overly sparse graphs (very small recall). Conversely, a recall equal to 1 can only be reached by adding a large number of FP edges, whatever the method we consider, even if some fail earlier than others. The main differences between the methods appear on the leftmost part of the P/R curves, especially those of Figure 5 B, C and D: while the precision of BootLasso, Genie3, PCalg and GSE drops rapidly with a slow increases in recall above 20% recall, it remains higher for GADAG. Hence, its first predicted edges are at least as accurate than those of the four other methods and it produces a larger set of reliable edges. For graphs of lesser sparsity, none of the five methods is really able to identify clearly reliable edges. Large number of FP edges are produced to achieve a recall higher than 60%.

For Networks 1 and to a lesser extent 5 (Figure 5 A and E), GADAG recovers with more difficulty the first true edges than other methods, with a high level of FP edges at the beginning of the curve (low

precision and low recall). However, as soon as the recall exceeds the 10%, *resp.* 15%, for graph A, *resp.* for graph E, GADAG performance is again superior to that other methods.

Table 3 gives the areas under the P/R curves for all methods and networks. For this indicator, GA significantly outperforms the state-of-the-art methods for all networks.

Table 3: Area under the Precision vs. Recall curve for all networks and methods (except ARACNE).

Method	Net 1	Net 2	Net 3	Net 4	Net 5
GADAG	**0.182**	**0.236**	**0.348**	**0.317**	**0.267**
Genie3	0.154	0.155	0.231	0.208	0.197
BootLasso	0.118	0.061	0.171	0.147	0.169
PCalg	0.116	0.089	0.171	0.149	0.130
GSE	0.101	0.089	0.170	0.153	0.133

6 Conclusion and discussion

In this paper, we proposed a hybrid genetic/convex algorithm for inferring large graphs based on a particular decomposition of the ℓ_1-penalized maximum likelihood criterion. We obtained two convergence inequalities that ensure that the graph estimator converges to the true graph under assumptions that mainly control the model structure: graph size (balance between sparsity, number of nodes and maximal degree) and signal-to-noise ratio. From an algorithmic point of view, the estimation task is split into two subproblems: node ordering estimation and graph structure learning. The first one is a non-trivial problem since we optimize over a discrete non-convex large dimensional set. It led us to use a heuristic approach we specifically tailored to achieve the optimization task. The second one is a more common problem, related to the Lasso one, for which we proposed a sound procedure with theoretical guarantees. The potential of such an approach clearly appeared in the numerical experiments, for which the behavior of our algorithm seemed to be very competitive when compared to the state-of-the-art.

Nevertheless, we see many opportunities for further improvements. First, convergence proof for the algorithm, although a challenging task, is worth investigating, for instance using the works of Cerf (1998) on genetic algorithms. An alternative would be to consider other optimization schemes for the node ordering with more established convergence proofs (*e.g.* simulated annealing (Granville et al., 1994)).

Second, other potential extensions involve algorithmic considerations in order to improve the calculation time, including a finer calibration of the algorithm parameters, an initialization step for the gradient descent, and, in general, improving the interactions between the nested and outer loops. Tackling very large datasets from several thousands of nodes may also require a particular treatment, for instance by adding local search operators to GADAG.

Finally, we would like to emphasize the graph identifiability problem: in our settings, we assume the noise variances of all graph nodes to be equal to ensure graph identifiability (that is no equivalence class of graphs). Such a hypothesis is of course restrictive and likely to be violated for real datasets. In order to infer networks for any noise variances, one solution consists in incorporating interventional data on the model. These data are obtained from perturbations of the biological system (*e.g.* gene knockouts or over-expressions) and make the equivalence class of graphs smaller (Hauser & Bühlmann, 2012). The use of additional data, informative yet very costly interventional data could be combined with observational on the MLE estimator. It was recently proposed by Hauser & Bühlmann (2015) for a BIC-score penalized MLE, or by Rau et al. (2013) for learning Gaussian Bayesian networks in the case of GRN inference. A modification of our hybrid algorithm GADAG could then lead to a more accurate identification of the true graph. Lastly, the cyclic structure framework could also be consid-

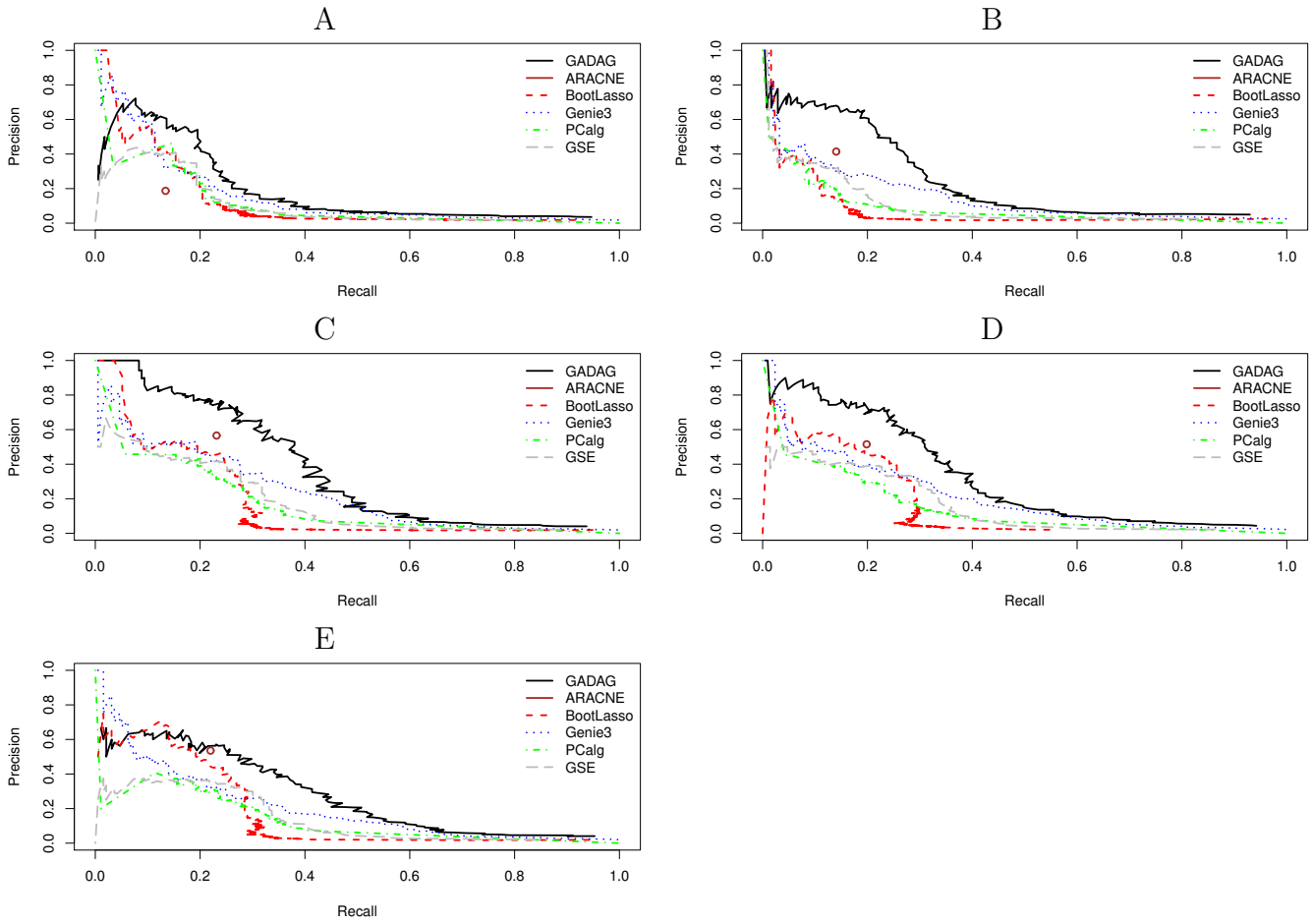


Figure 5: P/R curves for the five Dream networks and the five compared methods.

ered by using a Markov equivalence characterization (Richardson, 1997) to relax the strictly triangular assumption on our matrix T using Equation of Proposition 2.1. It would pave the way for totally new theoretical developments, and a more realistic modelling of a complex system.

References

Alander, J. (1992). On optimal population size of genetic algorithms. In *Proceedings of the IEEE Computer Systems and Software Engineering*.

Allouche, A., Cierco-Ayrolles, C., de Givry, S., Guillermin, G., Mangin, B., Schiex, T., Vandel, J. & Vignes, M. (2013). A panel of learning methods for the reconstruction of gene regulatory networks in a systems genetics. In *Gene Network Inference, Verification of Methods for Systems Genetics Data*. Springer, Heidelberg, alberto de la fuente ed., pp. 9–32.

Aragam, B., Amini, A. & Zhou, Q. (2015). Learning directed acyclic graphs with penalized neighbourhood regression. Preprint on arxiv at `https://arxiv.org/pdf/1511.08963.pdf`.

Bach, F. (2008). Bolasso: model consistent lasso estimation through the bootstrap. In *Proceedings of the Twenty-fifth International Conference on Machine Learning*.

Banerjee, O., El Ghaoui, L. & d'Aspremont,

A. (2008). Model selection through sparse maximum likelihood estimation for multivariate Gaussian or binary data. *J Mach Learn Res* **9**, 485–516.

Barabási, A. & Oltvai, Z. (2004). Network biology: understanding the cell's functional organization. *Nat Rev Genet* **5**, 101–113.

Bickel, P. & Li, B. (2006). Regularization in statistics. *Test* **15**, 271–344.

Bickel, P. J., Ritov, Y. & Tsybakov, A. B. (2009). Simultaneous analysis of lasso and Dantzig selector. *Ann Stat* **37**, 1705–1732.

Breiman, L. (2001). Random forests. *Mach Learn* **45**, 532.

Bühlmann, P. (2013). Causal statistical inference in high dimensions. *Math Method Oper Res* **77**, 357–370.

Bühlmann, P. & van de Geer, S. (2011). *Statistics for high-dimensional data*. Springer Series in Statistics. Springer, Heidelberg. Methods, theory and applications.

Cerf, R. (1998). Asymptotic convergence of genetic algorithms. *Adv Appl Probab* **30**, 521–550.

Champion, M., Picheny, V. & Vignes, M. (2017). *GADAG: A Genetic Algorithm for learning Directed Acyclic Graphs*. R package version 0.99.0.

Chen, J. & Chen, Z. (2008). Extended bayesian information criteria for model selection with large model spaces. *Biometrika* , 759–771.

Chickering, D. M. (1996). Learning Bayesian networks is NP-complete. In *Learning from data (Fort Lauderdale, FL, 1995)*, vol. 112 of *Lect. Notes Stat.* Springer, New York, pp. 121–130.

Chickering, D. M. (2002). Optimal structure identification with greedy search. *J Mach Learn Res* **3**, 507–554.

Cook, S. (1985). A taxonomy of problems with fast parallel algorithms. *Inform Control* **64**, 2–22.

Cormen, T. H., Leiserson, C. E., Rivest, R. L. & Stein, C. (2001). *Introduction to algorithms*. MIT Press, Cambridge, MA; McGraw-Hill Book Co., Boston, MA, 2nd ed.

Csiszár, I. & Tusnády, G. (1984). Information geometry and alternating minimization procedures. *Statist. Decisions* , 205–237Recent results in estimation theory and related topics.

Davis, L., ed. (1991). *Handbook of Genetic Algorithms*. Van Nostrand Reinhold.

De Smet, R. & Marchal, K. (2010). Advantages and limitations of current network inference methods. *Nature Reviews Microbiology* **8**, 717–729.

Dondelinger, F., Lèbre, S. & Husmeier, D. (2013). Non-homogeneous dynamic bayesian networks with bayesian regularization for inferring gene regulatory networks with gradually time-varying structure. *Machine Learning* **90**, 191–230.

Dréo, J., Pétrowski, A., Siarry, P. & Taillard, E. a. (2006). *Metaheuristics for hard optimization*. Springer-Verlag, Berlin. Methods and case studies, Translated from the 2003 French original by Amitava Chatterjee.

Duchi, J., Gould, S. & Koller, D. (2008). Projected subgradient methods for learning sparse gaussians. In *Proceedings of the 24th Conference on Uncertainty in Artificial Intelligence*.

Efron, B., Hastie, T., Johnstone, I. & Tibshirani, R. (2004). Least angle regression. *Ann Stat* **32**, 407–499.

El Karoui, N. (2008). Spectrum estimation for large dimensional covariance matrices using random matrix theory. *Ann Stat.* **36**, 2757–2790.

Ellis, B. & Wong, W. (2008). Learning causal Bayesian network structures from experimental data. *J Am Stat Assoc* **103**, 778–789.

Evtushenko, Y., Malkova, V. & Stanevichyus, A. (2009). Parallel global optimization of functions of several variables. *Comput. Math. Math. Phys.* **49**, 246–260.

Friedman, J., Hastie, T. & Tibshirani, R. (2007). Sparse inverse covariance estimation with the graphical lasso. *Biostatistics* **9**, 432–441.

Friedman, N. & Koller, D. (2003). Being Bayesian about network structure: a Bayesian approach to structure discovery in Bayesian networks. *Mach Learn* **50**, 95–125.

Fu, F. & Zhou, Q. (2013). Learning sparse causal Gaussian networks with experimental intervention: regularization and coordinate descent. *J Am Stat Assoc* **108**, 288–300.

Giraud, C. (2015). *Introduction to high-dimensional statistics*, vol. 139 of *Monographs on Statistics and Applied Probability*. CRC Press, Boca Raton, FL.

Granville, V., Krivanek, M. & Rasson, J.-P. (1994). Simulated annealing: A proof of convergence. *IEEE T Pattern Anal* **16**, 652–656.

Grefenstette, J., Gopal, R., Rosmaita, B. & Van Gucht, D. (1985). Genetic algorithms for the traveling salesman problem. In *Proceedings of the first International Conference on Genetic Algorithms and their Applications*.

Grzegorczyk, M. & Husmeier, D. (2008). Improving the structure MCMC sampler for bayesian networks by introducing a new edge reversal move. *Mach Learn* **71**.

Guyon, I., Aliferis, C. & Cooper, G. (2010). *Causation and Prediction Challenge*, vol. 2. Microtome Publishing. Challenges in Machine Learning.

Hauser, A. & Bühlmann, P. (2012). Characterization and greedy learning of interventional Markov equivalence classes of directed acyclic graphs. *J Mach Learn Res* **13**, 2409–2464.

Hauser, A. & Bühlmann, P. (2015). Jointly interventional and observational data: estimation of interventional Markov equivalence classes of directed acyclic graphs. *J R Stat Soc. Series B* **77**, 291–318.

Hogben, L., ed. (2007). *Handbook of linear algebra*. Discrete Mathematics and its Applications (Boca Raton). Chapman & Hall/CRC, Boca Raton, FL. Associate editors: Richard Brualdi, Anne Greenbaum and Roy Mathias.

Holland, J. H. (1975). *Adaptation in natural and artificial systems*. University of Michigan Press, Ann Arbor, Mich. An introductory analysis with applications to biology, control, and artificial intelligence.

Horst, R. & Pardalos, P. M., eds. (1995). *Handbook of global optimization*. Nonconvex optimization and its applications. Dordrecht, Boston: Kluwer Academic Publishers.

Hoyle, R., ed. (1995). *Structural equation modeling*. Thousand Oaks [u.a.]: Sage Publ.

Husmeier, D. & Werhli, A. (2007). Bayesian integration of biological prior knowledge into the reconstruction of gene regulatory networks with bayesian networks. *Comput Syst Bioinformatics* **6**, 85–95.

Huynh-Thu, V. A., Irrthum, A., Wehenkel, L. & Geurts, P. (2010). Inferring regulatory networks from expression data using tree-based methods. *PLoS ONE* **5**, e12776.

Jones, D. R., Perttunen, C. D. & Stuckman, B. E. (1993). Lipschitzian optimization without the lipschitz constant. *J Optimiz Theory App* **79**, 157–181.

Kahn, A. B. (1962). Topological sorting of large networks. *Commun. ACM* **5**, 558–562.

Kahre, K., Oh, S. & Rajaratnam, B. (2015). A convex pseudolikelihood framework for high dimensional partial correlation estimation with convergence guarantees. *JRSS B* **77**, 803–825.

Kalisch, M. & Bühlmann, P. (2007). Estimating high-dimensional directed acyclic graphs with the PC-algorithm. *J Mach Learn Res* **8**, 613–636.

Koivisto, M. & Sood, K. (2004). Exact bayesian structure discovery in Bayesian networks. *J Mach Learn Res* **5**, 549–573.

Ledoit, O. & Wolf, M. (2015). Spectrum estimation: A unified framework for covariance matrix estimation and pca in large dimensions. *J Multivariate Anal* **139**, 360384.

Liaw, A. & Wiener, M. (2002). Classification and regression by randomforest. *R News* **2**, 18–22.

Liu, H., Wang, L. & Zhao, T. (2014). Sparse covariance matrix estimation with eigenvalue constraints. *J Comput Graph Stat* **23**, 439–459.

Lounici, K., Pontil, M., Tsybakov, A. & van de Geer, S. (2009). Taking advantage of sparsity in multi-task learning. In *Proceedings of the 22nd Conference on Learning Theory*.

Maathuis, M., Colombo, D., Kalisch, M. & Bühlmann, P. (2010). Predicting causal effects in large-scale systems from observational data. *Nature Methods* **7**, 247–248.

Marbach, D., Schaffter, T., Mattiussi, C. & Floreano, D. (2009). Generating realistic in silico gene networks for performance assessment of reverse engineering methods. *J Comput Bio* **16**, 229–239.

Margolin, A., Nemenman, I., Basso, K., Wiggins, C., Stolovitzky, G., Dalla Favera, R. & Califano, A. (2006). Aracne: An algorithm for the reconstruction of gene regulatory networks in a mammalian cellular context. *BMC Bioinformatics* **7**, S7.

Meinshausen, N. & Bühlmann, P. (2006). High-dimensional graphs and variable selection with the lasso. *Ann Stat* **34**, 1436–1462.

Mestre, X. (2008). Improved estimation of eigenvalues and eigenvectors of covariance matrices using their sample estimates. *IEEE T Inform Theory* **54**, 5113–5129.

Michalewicz, Z. (1994). *Genetic algorithms + data structures = evolution programs*. Springer-Verlag, Berlin, 2nd ed.

Mordelet, F. & Vert, J. (2008). Sirene: Supervised inference of regulatory networks. *Bioinformatics* **24**, i76–i82.

Newman, M. (2003). The structure and function of complex networks. *SIAM Rev* **45**, 157–256.

Pearl, J. (2009). *Causality*. Cambridge University Press, Cambridge, 2nd ed. Models, reasoning, and inference.

Perrin, B., Ralaivola, L., Mazurie, A., Bottani, S., Mallet, J. & d'Alché Buc, F. (2003). Gene networks inference using dynamic bayesian networks. *Bioinformatics* **Supp2**, ii138–148.

Peters, J., Mooij, J., Janzing, D. & Schölkopf, B. (2014). Causal discovery with continuous additive noise models. *J Mach Learn Res* **15**, 2009–2053.

Peters, J., Mooij, J. M., Janzing, D. & Schölkopf, B. (2011). Identifiability of causal graphs using functional models. In *27th Conference on Uncertainty in Artificial Intelligence (UAI 2011)*.

Piszcz, A. & Soule, T. (2006). Genetic programming: Optimal population sizes for varying complexity problem. In *Proceedings of the Genetic and Evolutionary Computation Conference*.

R Core Team (2017). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.

Rau, A., Jaffrézic, F. & Nuel, G. (2013). Joint estimation of causal effects from observational and intervention gene expression data. *BMC Syst Biol* **7**, 111.

Richardson, T. (1997). A characterization of markov equivalence for directed cyclic graphs. *Int J Approx Reason* **17**, 107–162.

Ridge, E. (2007). *Design of Experiments for the Tuning of Optimisation Algorithm*. Ph.D. thesis, The University of York, Department of Computer Science.

Robinson, R. W. (1973). Counting labeled acyclic digraphs , 239–273.

Sachs, K., Itani, S., Fitzegerald, J., Wille, L., Schoeberl, B., Dahleh, M. & Nolan, G. (2009). Learning cyclic signaling pathway structures while minimizing data requirements. In *Proceedings of the Pacific Symposium on Biocomputing*.

Schaffer, J. D., Caruana, R., Eshelman, L. J. & Das, R. (1989). A study of control parameters affecting online performance of genetic algorithms for function optimization. In *Proceedings of the Third International Conference on Genetic Algorithms*.

Schwarz, G. E. (1978). Estimating the dimension of a model. *The Annals of Statistics* **6**, 461–464.

Sergeyev, Y. (1995). An information global optimization algorithm with local tuning. *SIAM J Optimiz* **5**, 858–870.

Sergeyev, Y. & Kvasov, D. (2006). Global search based on efficient diagonal partitions and a set of lipschitz constants. *SIAM J Optimiz* **16**, 910–937.

Shojaie, A., Jauhiainen, A., Kallitsis, M. & Michailidis, G. (2014). Inferring regulatory networks by combining perturbation screens and steady state gene expression profiles. *PLoS ONE* **9**, e82393.

Shojaie, A. & Michailidis, G. (2010). Penalized likelihood methods for estimation of sparse high-dimensional directed acyclic graphs. *Biometrika* **97**, 519–538.

Silander, T. & Myllymäki, T. (2006). A simple approach for finding the globally optimal bayesian network structure. In *Proceedings of the Twenty-second Conference on Uncertainty in Artificial Intelligence*.

Souma, W., Fujiwara, Y. & Aoyama, H. (2006). Heterogeneous economic networks. In *The complex networks of economic interactions*, vol. 567 of *Lecture Notes in Econom. and Math. Systems*. Springer, Berlin, pp. 79–92.

Spirtes, P. (1995). Directed cyclic graphical representations of feedback models. In *Proceedings of the 11th Annual Conference on Uncertainty in Artificial Intelligence*.

Spirtes, P., Glymour, C. & Scheines, R. (2000). *Causation, prediction, and search*. Adaptive Computation and Machine Learning. Cambridge, MA: MIT Press, 2nd ed. With additional material by David Heckerman, Christopher Meek, Gregory F. Cooper and Thomas Richardson, A Bradford Book.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *J R Stat Soc Series. B* **58**, 267–288.

Tsamardinos, I., Brown, L. & Aliferis, C. (2006). The max-min hill-climbing bayesian network structure learning algorithm. *Mach Learn* **65**, 31–78.

van de Geer, S. & Bühlmann, P. (2013). ℓ_0-penalized maximum likelihood for sparse directed acyclic graphs. *Ann Stat* **41**, 536–567.

Verma, T., Araújo, N. & Herrmann, H. (2014). Revealing the structure of the world airline network. *Scientific Reports* **5**.

Verma, T. & Pearl, J. (1991). Equivalence and synthesis of causal models. In *Proceedings of the 6th Annual Conference on Uncertainty in Artificial Intelligence*.

Verzelen, N. (2012). Minimax risks for sparse regressions: Ultra-high dimensional phenomenons. *Electron J Stats* **6**, 38–90.

Wainwright, M. (2009). Information-theoretic limits on sparsity recovery in the high-dimensional and noisy setting. *IEEE T Info Theory* **55**, 5728–5741.

Witten, D., Freidman, J. & Simon, N. (2011). New insights and faster computations for the graphical lasso. *J COmput Graph Stat* **20**, 892–900.

Wright, S. (1921). Corelation and causation. *J Agric Res* , 558–585.

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

Zhou, Q. (2011). Multi-domain sampling with applications to structural inference of Bayesian networks. *J Am Stat Assoc* **106**, 1317–1330.