Master Thesis

Radboud University Nijmegen

Network Deconvolution

An analysis

Author:
Emma Gerritse

Supervisor:
Johannes Textor

Second reader:
Tom Claassen

November 21, 2018

1 Abstract

Many datasets contain measures of associations, such as correlations, between different variables. We are often interested in which of these associations reflect direct effects of the variables on each other, and which ones reflect merely an indirect relation. However, discriminating between direct and indirect effects can be very difficult and expensive.

For example, consider a social network consisting of a large group of people, where we wish to determine which individuals are friends with each other. We could ask everyone to list all their friends, but that would take a lot of time. Instead, it would be easier to analyze associations between these people on social media. People who are friends in real life are probably friends on social media, but friends in a social network could also simply be colleagues or have met each other at a party of mutual friends. Here, the information available on social networks would not be enough to answer our question. This raises the question of whether we could use the social network information to somehow extract direct relations from possibly indirect associations.

Network Deconvolution (ND) is a recently proposed method to infer direct relations from association measures. Given a network of mutual *associations* in the form of a symmetric, real-valued matrix, ND computes another matrix representing a network depicting supposedly direct *interactions*. In our social media example, the association network would be friendships on social media, and the interaction network would be real-life friendships between people. The novelty in this method lies in the way direct interactions are calculated. Unlike other methods, ND is based on the *transitive reduction* – the inverse of the transitive closure of a network. Thus, ND assumes that the association network is the transitive closure of the interaction network. Under this assumption, the transitive reduction of the association network indeed recovers the interaction network. This differs from other methods to recover interaction networks, such as the precision matrix, which are often purely based on statistics. Because the transitive closure does not converge if the eigenvalues of the interaction matrix are too large, we may need to normalize the input by scaling. Otherwise, we could not have gotten the association network as a result of transitive closure.

The paper that first introduced ND showed empirically that it performs well on some datasets. However, an empirical or theoretical comparison to related methods such as the precision matrix has not been performed yet. In particular, the assumption of transitivity on which the method is based does not appear to hold for many real-world interactions – two people having a mutual friend do not need to be friends on social media. It is unclear under what conditions ND might work when the transitivity assumption is violated.

In this research, we considered ND in the setting where the interaction networks are Gaussian-linear Bayesian networks (also known as structural equation models), and the association networks are their implied covariance matrices. We first analytically solved network deconvolution on networks describing transitive and intransitive cases, and found that both ND and the precision matrix gave similar results and cannot correctly retrieve the non-transitive interaction network. Based on these insights, we then proved that ND without normalization and the precision matrix are mathematically equivalent. However, we have also shown that the normalization, which we did not account for in our proof, removed this equivalence and had substantial impact on ND results.

We then performed an empirical analysis in which we compare ND to other methods on real-world datasets, and found similar results for all methods. To get better insights in which cases ND works well and in which it does not, we then used simulated datasets with different levels of the sparsities of the generating Bayesian Networks, different amount of nodes and different levels of association strengths. In this analysis, ND outperformed the precision matrix, especially on datasets generated through dense networks, despite their only difference being normalization. Indeed, when systematically varying ND normalization factors in our simulated datasets, we found that the chosen value of the ND method was always optimal. We wondered if it was possible to apply normalization to any given method. It turned out that the normalization method used in ND can be applied to some other methods, like the partial correlation, improving their results. Lastly, we wondered if we could assign any interpretation to the ND normalization step other than presented in the original paper. We found that the ND normalization can be represented in a Bayesian Network as an incorporation of measurement error in the variables, which shrinks estimated (partial) correlations towards 0. Thus, the ND normalization can be seen as a form of shrinkage.

We can conclude that the empirical performance of ND, even when transitivity does not hold, can be explained by its strong relation to the precision matrix. Moreover, ND even outperforms the precision matrix and other methods because of the additional normalization step, especially on dense matrices. We have also shown empirically that the parameter chosen for normalization is optimal. Therefore, our results suggest that even when the underlying transitivity assumption of ND does not hold, it can still be a good choice to consider when attempting to recover direct interaction strengths from association data.

Contents

2 Introduction

2.1 Motivation

When having some system of interacting variables, we can calculate some measures of association with for example a covariance matrix. However, we do not know if these associations reflect direct or indirect effects between these variables. Often, it is important to know if certain variables interact with each other directly. Different methods have been proposed already to calculate these, for example calculating the precision matrix [Dob04], partial correlation matrix [Mad94] or graphical lasso [FHT07]. In some cases, methods specific to the dataset have been developed, as can be seen in the example below.

In this thesis, we discuss one general method for extracting direct effects called Network Deconvolution (ND). But first, let us discuss a problem to which ND can be applied.

Protein structure prediction is the task of predicting how a protein folds from its one-dimensional amino-acid sequence. This is an important task in biotechnology and medicine. For example, some medicines work because they bind to a certain protein. If we know its structure, we can design a medicine more efficiently to bind to that protein. Protein folding is a physical process. Determining the structure of a protein can also be done in a laboratory, but this is labor-intensive and expensive.

The same protein can occur in multiple organisms, having a slightly different chain of amino acids in each one. However, since the protein has the same function, the structure must also be the same. Thus the structure of proteins can be predicted by which pairs of amino acids are more prone to be changed together, since they would probably be close to each other in the 3D structure. In figure 1 we can see an example of that, where the method to discover direct effects for protein structure prediction has been proposed in [MPL+11]. We can calculate the correlation between two columns of amino acids, which we will call the Mutual Information (MI). Then, using the MI between all pairs, we can calculate the Direct Information (DI). In figure 2, we can see the predictions made using that method, with **A** the top 20 pairs of MI, and **B** the DI, with red indicating a distance of less than 8 ångström and green a distance greater than that.

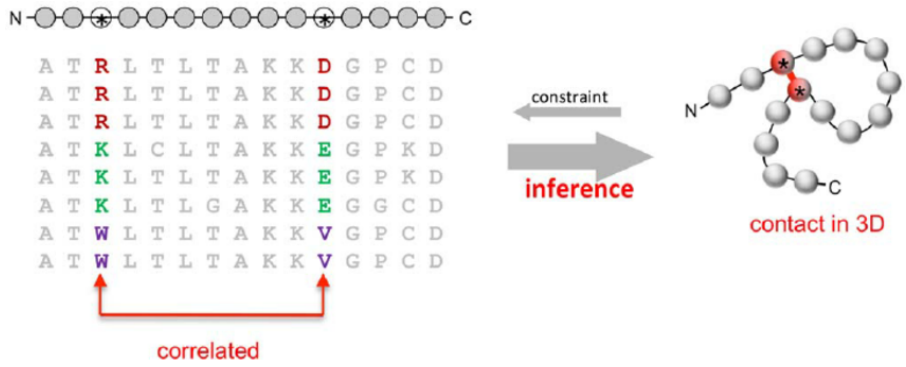


Figure 1: Multiple sequence alignment inferencing contact points, as can be found in [MCS+11]

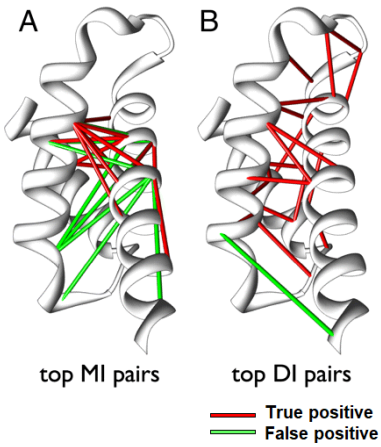


Figure 2: Contact points: Direct relations, edited from image in [MPL+11]

2.2 State of the Art

Several methods for recovering direct relations have been proposed. One of the most basic methods for this task is the precision matrix [Dob04], which is simply the matrix inverse. Partial correlation is also similar to this method, but adds a scaling based on the square root of the diagonal elements of the precision matrix [Mad94]. Another method is graphical lasso [FHT07], which is an algorithm used to estimate the precision matrix using lasso regression. It works especially well on sparse matrices.

In this research, we focus on ND, a method proposed by [FMMK13]. This method is, unlike the other three methods mentioned above, not necessarily based in statistics. The authors claim that the observed matrix is the geometric sum of the original matrices, and this should follow out of taking the transitive closure of a network. Then, the original network can be retrieved by ND, which is the inverse formula of the geometric sum. They show that the method performs well on three different datasets. The method has been used in many other pieces of research as well, such as plankton interaction networks [LMFH$^+$15] and co-occurrence of soil microbiota [MWD$^+$16].

However, we found some things to be lacking in the original paper. First of all, the methods were proved to perform well, but were not compared to other state of the art methods. Even though the method works well, it does not work perfectly in their paper. An analysis in which cases the method succeeds and in which it fails was also not given. Most importantly, ND assumes transitivity. However, many networks are not transitive. For example, when basing friendships on some levels of interaction, two people who have mutual friends could have never met each other. This is why we thought it would be interesting to investigate the method in more detail, attempting to understand why it works, and on which cases it works best.

2.3 Structure of this Thesis

This thesis is structured in the following way. In chapter 4, we perform a mathematical analysis of ND. ND applies normalization to ensure that the eigenvalues of the input matrix are not too large. To simplify our problem, we first consider ND without this normalization, and apply this simplified ND to problems involving few variables. To precisely define the difference between interactions and associations, we use Direct Acyclic Graphs (DAG) to construct covariance matrices. We start on DAGs consisting of only 3 variables. In particular, one of these DAGs, the collider $X \to M \leftarrow Y$, represents a non-transitive interaction structure. On these simple examples, we compare simplified ND to the precision matrix. After having seen the similarities in results between these two methods, we give some form of mathematical equivalence. Then, we take a better look at normalization, and how this changes the structure of the resulting matrix.

In chapter 5.1, we evaluate empirically how well ND works on the co-author dataset as used in [FMMK13], and compare these results to the precision matrix and partial correlation.

In chapter 5.2, we perform a systematic empirical evaluation using datasets simulated using structural equation models to evaluate how ND behaves on datasets with different structures. Here, we see that ND works better than the precision matrix, despite differing only in normalization. Normalization in ND is performed using a scaling factor that is determined automatically from the eigenvalues of the input matrix. Asking whether this value is optimal, we evaluate if other values give a better score. We also ask whether normalization can be applied more widely to other methods as well, and test the normalization on graphical lasso and the partial correlation. Our results suggest seems that normalization improves the performance of the partial correlation, especially on covariance matrices generated through dense graphs.

Chapter 6 aims to present an intuitive explanation of why normalization appears to perform well. To this end, we construct a representation of the normalization step using Bayesian networks, which shows that the normalization can be understood as a form of shrinkage.

The appendix (chapter 8) contains further analyses that cover additional cases not considered in the main text.

There are further analytical DAG investigations, with more different structures of DAGs, but also with either 0 or 1 on the diagonal of the input matrix.

Lastly, we provide listings of both the `Sage` code and the `R` code used to conduct the analytical and empirical research reported in this thesis. The `Sage` code was used to help calculate symbolic covariance matrices, and the `R` code was used for the simulations.

3 Background

In this chapter, we will provide definitions of the relevant concepts that Network Deconvolution (ND) is based on.

Definition 3.1 (Covariance)**.** For two random variables x and y, the covariance is defined by

$$cov(x, y) = E_{x,y}[xy] - E[x]E[y] \tag{1}$$

Definition 3.2 (Correlation)**.** For two random variables x and y, the correlation is defined by

$$corr(x, y) = \frac{cov(x, y)}{\sigma_x \sigma_y} \tag{2}$$

Definition 3.3 (Covariance Matrix)**.** For a set of variables $X = [x_1, \ldots, x_n]$, the *covariance matrix* Σ is the matrix where

$$\Sigma_{i,j} = cov(x_i, x_j) \tag{3}$$

Definition 3.4 (Correlation Matrix)**.** For a set of variables $X = [x_1, \ldots, x_n]$, the *correlation matrix* can be written as

$$cor(X) = diag(\Sigma)^{-\frac{1}{2}} \Sigma diag(\Sigma)^{-\frac{1}{2}} \tag{4}$$

where $diag(\Sigma)$ is the diagonal of the covariance matrix Σ.

Definition 3.5 (Precision Matrix)**.** The *precision matrix* P is the inverse of the covariance matrix C, giving $P = C^{-1}$

Definition 3.6 (Eigenvector and Eigenvalues)**.** For a matrix A, an *eigenvector* v and *eigenvalue* λ are a vector and value such that $Av = \lambda v$

Theorem 3.7. For a *geometric sequence* with $|r| < 1$, we know that

$$\sum_{n=1}^{\infty} r^n = r + r^2 + r^3 + \cdots = \frac{r}{1 - r} \tag{5}$$

Proof. The proof of this basic theorem can be found in many textbooks on basic calculus or real analysis, for example [Tao16], *lemma 7.3.3*. □

Definition 3.8 (Directed Acyclic Graph)**.** A graph consists of nodes and edges. A *Directed Acyclic Graph* (DAG) is a graph in which each edge is directed, and there is no directed path from a node to itself.

Definition 3.9 (Bayesian Network)**.** A *Bayesian Network* for a set of variables V is a tuple (G, f) where G is a directed acyclic graph and f is a joint density for the variables V that factorizes as $f(V) = \prod_{v \in V} f(v \mid \text{Pa}(v))$, where $\text{Pa}(v)$ denotes the parents of v in G.

Definition 3.10 (Structural Equation Model)**.** A *Structural Equation Model (SEM)* is a Bayesian network (G, f) in which each variable is a linear combination of its parents in G and a Gaussian variable ϵ_v with zero mean and fixed variance. Throughout this thesis, we assume that the variance of ϵ_v is set such that the variance of v is 1. Then we can parameterize an SEM using one number per edge $u \to v$, which is the linear factor of u in v. We call this number the *edge strength.*

Definition 3.11 (Graphical Lasso)**.** The *Graphical lasso* is a method developed by [FHT07] to estimate sparse graphs by applying a lasso penalty. For N multivariate normal observations of dimension p with mean μ and covariance Σ, we want to estimate $\Theta = \Sigma^{-1}$. Let S be the empirical covariance matrix, then the graphical lasso is the estimate $\hat{\Theta}$ such that

$$\hat{\Theta} = argmin_{\Theta \geq 0} \log\det(\Theta) - \text{tr}(S\Theta) - \rho \|\Theta\|_1 \tag{6}$$

Definition 3.12 (Partial Correlation)**.** Given the correlation matrix C, its inverse $P = C^{-1}$, we can define the *partial correlation* ρ between elements i, j as

$$\rho_{ij} = -\frac{p_{ij}}{\sqrt{p_{ii} p_{jj}}} \tag{7}$$

Definition 3.13 (Primitive ND)**.** ND is a general method for inferring direct effects (interactions) G_{dir} from an observed association matrix G_{obs}. This matrix can contain both direct and indirect effects. The direct matrix G_{dir} is defined as the matrix representing the effects of the graph of the true network. Then, the observed matrix G_{obs} of this effect is assumed to have been obtained from G_{dir} by forming the transitive closure

$$G_{\text{obs}} = G_{\text{dir}} + G_{\text{dir}}^2 + G_{\text{dir}}^3 + \ldots \tag{8}$$

This transitive closure is an infinite sum, but it has a closed form that can be obtained in a similar way as in equation 5, but then with matrices instead of numbers:

$$G_{\text{obs}} = G_{\text{dir}}(I - G_{\text{dir}})^{-1} \,. \tag{9}$$

By rearranging the above equation, we obtain an expression that recovers the interaction matrix G_{dir} from the association matrix G_{obs}:

$$G_{\text{obs}} = G_{\text{dir}}(I - G_{\text{dir}})^{-1} \tag{10}$$
$$G_{\text{obs}}(I - G_{\text{dir}}) = G_{\text{dir}} \tag{11}$$
$$G_{\text{obs}} - G_{\text{obs}}G_{\text{dir}} = G_{\text{dir}} \tag{12}$$
$$G_{\text{obs}} = G_{\text{dir}} + G_{\text{obs}}G_{\text{dir}} \tag{13}$$
$$G_{\text{obs}} = (I + G_{\text{obs}})G_{\text{dir}} \tag{14}$$
$$G_{\text{dir}} = G_{\text{obs}}(I + G_{\text{obs}})^{-1} \tag{15}$$

Now we can define *primitive ND* for a matrix C as simply the application of the above equation, i.e.:

$$D(C) = C(C + I)^{-1} \tag{16}$$

Here, the input matrix C is assumed to have 0 on its diagonal (meaning that the direct relationship, or interaction, between a variable and itself is always defined to be 0).

Definition 3.14 (Normalization)**.** To make sure that the equality in formula 9 still holds, we can normalize the matrix using eigenvalues in the following way. If the unscaled eigenvalues for the dependency matrix $G^{\text{us}}_{\text{obs}}$ are too large, we scale them with a factor α such that $G_{\text{obs}} = \alpha G^{\text{us}}_{\text{obs}}$. Suppose that $\lambda_+^{\text{obs(us)}}$ and $\lambda_-^{\text{obs(us)}}$ are the smallest negative and the largest positive eigenvalues. Then by using

$$\alpha = \min\left(\frac{\beta}{(1-\beta)\lambda_+^{\text{obs(us)}}}, \frac{-\beta}{(1+\beta)\lambda_-^{\text{obs(us)}}}\right), \tag{17}$$

it is guaranteed that the largest absolute eigenvalue of G_{dir} will be smaller than $\beta < 1$. The proof of this is given in the supplementary material of [FMMK13].

Definition 3.15 (Normalized Network Deconvolution)**.** We can now define *normalized ND* as

$$\tilde{D}(C) = D(\alpha C) \tag{18}$$

with α as found in equation 17. If we mention the term ND, we will always mean normalized ND.

The same normalization operation could be applied to any other method that uses an association matrix, such as the precision matrix.

Definition 3.16 (Off-diagonal Scaling)**.** We will also apply the normalization step of ND to other methods, but it will then only apply to the off-diagonal elements of an input matrix. We therefore introduce a form of *off-diagonal scaling* as

$$c \oslash A = \begin{cases} c \cdot a_{ij} & \text{if } i \neq j \\ a_{ii} & \text{otherwise} \end{cases}$$

Example 3.17. Suppose we have a matrix

$$A = \begin{pmatrix} 1 & x_1 & x_2 \\ x_1 & 1 & x_3 \\ x_2 & x_3 & 1 \end{pmatrix}$$

then the result of the normalization operator is

$$\alpha \oslash A = \begin{pmatrix} 1 & \alpha x_1 & \alpha x_2 \\ \alpha x_1 & 1 & \alpha x_3 \\ \alpha x_2 & \alpha x_3 & 1 \end{pmatrix}$$

4 Analytical Investigation of ND

In this chapter, we will perform an in-depth analysis of ND when applied to covariance matrices generated through different small Gaussian-linear Bayesian networks. We will compare ND to the precision matrix, and we will show its equivalent to the precision matrix.

4.1 Examples of ND

After having defined primitive ND, we want to know how it behaves. Can we obtain a general understanding of which properties are recovered, and which ones are not? We are going to test this by applying ND to covariance matrices with certain properties. To provide meaning to these covariance matrices, we are going to generate them through SEMs. In this manner, we know which relations are direct and which are indirect. We can also calculate the implied covariance matrix from a given SEM. Note that the nodes in this case are variables, and the values on the edges represent the strengths of direct causal effects between the variables. This value will be between -1 and 1.

Example 4.1 (Example with primitive ND: Fork case)**.** First, we will look at the fork case, which is as follows:

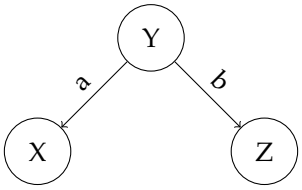


Figure 3: Fork

This will give us the following covariance matrix:

$$C = \begin{array}{c} \begin{matrix} X & Y & Z \end{matrix} \\ \begin{pmatrix} 1 & a & a \cdot b \\ a & 1 & b \\ a \cdot b & b & 1 \end{pmatrix} \end{array} \begin{matrix} X \\ Y \\ Z \end{matrix}$$

Note that for ND, we need to have a 0 diagonal, so we will have the following input matrix:

$$C' = C - I = \begin{array}{c} \begin{matrix} X & Y & Z \end{matrix} \\ \begin{pmatrix} 0 & a & a \cdot b \\ a & 0 & b \\ a \cdot b & b & 0 \end{pmatrix} \end{array} \begin{matrix} X \\ Y \\ Z \end{matrix}$$

Then we can calculate the primitive deconvolved matrix, which will give us the following results:

$$D(C') = \begin{array}{c} \begin{matrix} X & Y & Z \end{matrix} \\ \begin{pmatrix} \frac{a^2}{a^2-1} & -\frac{a}{a^2-1} & 0 \\ -\frac{a}{a^2-1} & \frac{\left(2a^2-1\right)b^2-a^2}{(a^2-1)b^2-a^2+1} & -\frac{b}{b^2-1} \\ 0 & -\frac{b}{b^2-1} & \frac{b^2}{b^2-1} \end{pmatrix} \end{array} \begin{matrix} X \\ Y \\ Z \end{matrix}$$

So we see that this gives us a 0 at the right place, the nodes X and Z are not connected. So this will give us the correct results.

Now let us regard the same matrix, but then calculate the precision matrix. Then we need to use the original implied covariance matrix C, so this will give us:

$$P(C) = \begin{array}{c} \begin{matrix} X & Y & Z \end{matrix} \\ \begin{pmatrix} \frac{1}{1-a^2} & \frac{a}{a^2-1} & 0 \\ \frac{a}{a^2-1} & \frac{1-a^2b^2}{(a^2-1)(b^2-1)} & \frac{b}{b^2-1} \\ 0 & \frac{b}{b^2-1} & \frac{1}{1-b^2} \end{pmatrix} \end{array} \begin{matrix} X \\ Y \\ Z \end{matrix}$$

Example 4.2 (Example with primitive ND: Collider case)**.** Since colliders behave differently than the previous case, we want to look at them in particular as well. We know with colliders, that the covariance matrix will be non transitive. Since ND is based on taking the transitive closure, we wonder if it will still work in this case.

Let us regard the following DAG:

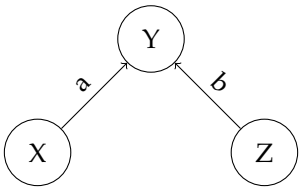


Figure 4: Collider

This will give the following implied covariance matrix:

$$C = \begin{array}{c} \begin{matrix} X & Y & Z \end{matrix} \\ \begin{pmatrix} 1 & a & 0 \\ a & 1 & b \\ 0 & b & 1 \end{pmatrix} \end{array} \begin{matrix} X \\ Y \\ Z \end{matrix}$$

This will give us the input matrix as seen below:

$$C' = C - I = \begin{array}{c} \begin{matrix} X & Y & Z \end{matrix} \\ \begin{pmatrix} 0 & a & 0 \\ a & 0 & b \\ 0 & b & 0 \end{pmatrix} \end{array} \begin{matrix} X \\ Y \\ Z \end{matrix}$$

Then we can calculate the primitive deconvolved matrix, which will give us the following results:

$$D(C') = \frac{1}{a^2+b^2-1} \begin{array}{c} \begin{matrix} X & Y & Z \end{matrix} \\ \begin{pmatrix} a^2 & -a & ab \\ -a & a^2+b^2 & -b \\ ab & -b & b^2 \end{pmatrix} \end{array} \begin{matrix} X \\ Y \\ Z \end{matrix}$$

This will give us zero values if $a \cdot b = 0$, but this can only happen if $a = 0 \vee b = 0$, so this will always give us a non-zero interaction between X and Z.

Now let us regard the same matrix, but then calculate the precision matrix. For this, we will use the original implied covariance matrix as input. So this will give us:

$$P(C) = \frac{1}{a^2+b^2-1} \begin{array}{c} \begin{matrix} X & Y & Z \end{matrix} \\ \begin{pmatrix} b^2-1 & a & ab \\ a & 1 & b \\ -ab & b & a^2-1 \end{pmatrix} \end{array} \begin{matrix} X \\ Y \\ Z \end{matrix}$$

We see that this gives a non zero value as relation between X and Z, the same as $D(C)$.

These results illustrate that primitive ND and taking the precision matrix yield similar results. We show more examples of this in appendix 8.1, and also consider the case where ND would be applied without normalizing the diagonal. Indeed, we will now provide a simple lemma that explains why the results of ND and the precision matrix are so similar: they are mathematically strongly related.

4.2 Mathematical Analysis of ND

Lemma 4.3. *For a correlation matrix C, we know that we get the precision matrix by calculating $P(C) = C^{-1}$. We wonder what the relation is between the precision matrix and the primitive deconvolved matrix. When subtracting the diagonal from the covariance matrix, we get $C' = C - I$. Then:*

$$D(C') = I - P(C) \tag{19}$$

Proof.

$$\begin{aligned} D(C') &= C'(C' + I)^{-1} \\ &= (C - I)(C - I + I)^{-1} \\ &= (C - I)(C)^{-1} \\ &= I - C^{-1} \\ &= I - P(C) \end{aligned}$$

□

So when not normalizing, deconvolving will be very similar to computing the precision matrix. Before explaining how normalization destroys this correspondence, we first give an example that explains the motivation behind the normalization step applied in ND.

Example 4.4. The geometric sequence in formula 9 does not converge if the eigenvalues are too large. This is the same idea that the geometric sequence in formula in 5 would not converge if the value of r is too large. If this is the case, although we could still apply formula 5, the equality of the direct formula would not hold. For example:

$$\sum_{n=1}^{\infty} 2^n = 2 + 2^2 + 2^3 + \cdots = \infty \neq \frac{2}{1-2} = -2$$

To make sure this inequality does not happen, we need to use ND with normalization, as defined in definition 3.15.

Example 4.5 (Compatibility under multiplication)**.** We want to see what kind of effects normalization has on the results. For example, could we introduce false negatives? Suppose we have a direct effect matrix. Then, we can calculate the indirect effect matrix by taking the reverse of primitive ND. After that we can apply normalization and then use primitive ND again. Can we say anything about the results? In other words, would the following relation hold:

$$D(q \cdot D^{-1}(E)) \overset{?}{\sim} q \cdot E$$

Suppose we have a direct effect matrix:

$$E = \begin{pmatrix} a & b & 0 \\ c & d & e \\ 0 & f & g \end{pmatrix}$$

If we want to get indirect effect matrix, we simply have to reverse the process of ND. This will give us:

$$\begin{aligned} D(C) &= C \cdot (C + I)^{-1} \\ D(C) \cdot (C + I) &= C \\ D(C) \cdot C + D(C) &= C \\ D(C) \cdot C - C &= -D(C) \\ (D(C) - I)C &= -D(C) \\ C &= (D(C) - I)^{-1} \cdot -D(C) \end{aligned}$$

So now we know that

$$D^{-1}(C) = (C - I)^{-1} \cdot -C \tag{20}$$

To determine an analytic expression for $D^{-1}(E)$, we used the computer algebra system `Sage` [The18] (code found in appendix 8.2.1), and we will get the following matrix for $D^{-1}(E)$:

$$D^{-1}(E) = \frac{1}{\det} \begin{pmatrix} \det - (1 - d - ef - g + dg) & bg - b & be \\ cg - c & \det - (a-1)(g-1) & (a-1)e \\ cf & (a-1)f & \det - ((a-1)d - a - bc + 1) \end{pmatrix}$$

where det $= (a-1)ef - bc + (a-1)d + (bc - (a-1)d + a - 1)g - a + 1$.

Now, if we multiply $D^{-1}(E)$ with a factor q and then deconvolve, we get the following matrix:

$$D(q \cdot D^{-1}(E)) = \frac{1}{det} \cdot \begin{pmatrix} \dots & -bgq^2 + (bg - b)q & beq^2 - beq \\ -cgq^2 + (cg - c)q & \dots & -aeq^2 + (a-1)eq \\ cfq^2 - cfq & -afq^2 + (a-1)fq & \dots \end{pmatrix}$$

We can clearly see that for $q \notin \{-1, 0, 1\}$, zero elements in the unnormalized deconvolved matrix values will turn into nonzero elements in the deconvolved matrix. So normalizing introduces nonzero values, and $D(q \cdot D^{-1}(E)) \nsim q \cdot E$.

Now, let us consider this result, but with $a = d = g = 0$, and $c = f = b = e$. Then we get as a result:

$$\frac{1}{det} \cdot \begin{pmatrix} \dots & -b \cdot q & b^2 \cdot q^2 - b^2 \cdot q \\ -b \cdot q & \dots & -b \cdot q \\ (b^2 \cdot q^2 - b^2 \cdot q) & -b \cdot q & \dots \end{pmatrix}$$

Then

$$\begin{aligned} b^2q^2 - b^2q &< 0 \\ b^2(q^2 - q) &< 0 \\ q^2 - q &< 0 \\ q^2 &< q \end{aligned}$$

We here see that the connection between the first variable and third variable becomes negative when $q^2 \leq q \implies 0 < q < 1$.

ND seems to find true negatives and false negatives in the same cases as the precision matrix does. This led to a proof of the mathematical equivalence between these two methods. However, this did not regard normalization. Normalization seems to give different results entirely, more than simply scaling. Normalization does not seem to give consistent true negatives. To see how ND behaves when including normalization, we need to test it on actual datasets.

5 Emperical Evaluation

5.1 Evaluation on Co-author Dataset

Having shown that ND is in fact mathematically equivalent to the precision matrix except for the normalization step, we now turn our attention to an empirical evaluation of how ND performs and how this compares to alternative methods. In [FMMK13], Network Deconvolution is tested on three datasets. One of these datasets is a co-author network of 1589 scientific authors, where two authors are connected if they where co-authors in at least one scientific paper [1]. The direct interactions in this dataset are defined as the interaction between authors which are larger than 0.5, after normalizing the input matrix between 0 and 1. We want to know if two authors have a weak or a strong connection to each other. The input of this data is not a covariance matrix, but an adjacency matrix.

We wondered how ND would compare to the precision matrix and another method, the partial correlation (see def 3.12) on this dataset. The precision matrix and partial correlation both require covariance matrices as input. However, adjacency matrices are not covariance matrices. So to make the input valid, we multiplied the whole matrix with 0.1, which is an arbitrary value to make sure that the input data set could have been a covariance matrix. For ND, the input has 0 as diagonal and for the other two methods, the matrix has 1 as diagonal. We will compare the three methods using the Area Under Curve (AUC), which is calculated through the Receiver Operating Characteristic (ROC), for which we take the connectedness of two nodes as a binary label, and the absolute value of the interaction strength in the resulting matrix as the prediction.

This resulted in figure 5. As we can see here, the AUC of ND is the same as for the partial correlation applied to the modified input matrix, whereas the AUC of the precision matrix is slightly lower.

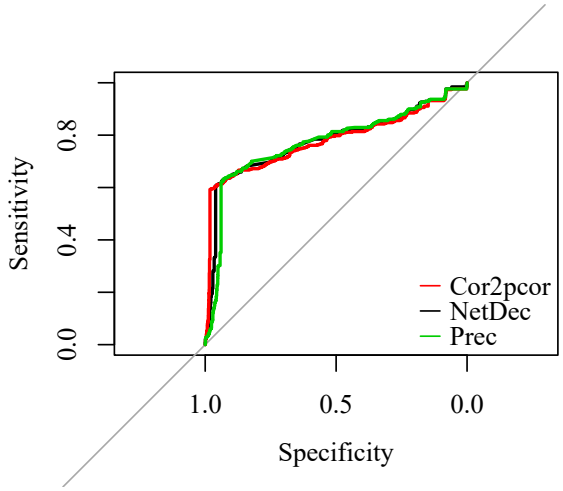


Figure 5: ROC-curve of ND, precision and Cor2PCor. AUC scores are: Network Deconvolution: 0.779; partial correlation: 0.779; precision: 0.772

We have seen that the AUC of all three methods are roughly the same on this dataset. This is what we would expect, given that all three methods are similar. However, we cannot conclude anything after testing these methods on just one dataset. That is why we will compare these methods on more datasets in the next subsection.

[1] Dataset can be found at `http://compbio.mit.edu/nd`

5.2 Simulated Data

Our empirical evaluation on real-world data in the previous section showed little difference performance-wise between ND and the other methods we considered. In this section, we aim to understand better if certain properties of the input network would have any influence on which method performs the best. To test that, we will evaluate the methods on simulated data. The code for this can be found in appendix 8.2.2.

In each simulation, we generated 100 random DAG structures each for 100 different sparsity levels. For each of those points, we compared the area under curve. We generated the random DAGs using the function `randomDAG` from the `R` package `dagitty` [TvG+16]. This function has two parameters: N, the number of nodes, and p, the chance that two random nodes are connected to each other. `SimulateSEM` is used from the same `R` package to simulate an observed covariance matrix. Here we have the following parameters: β is the effect strength corresponding to each arrow. We set the parameters `standardized` to `FALSE`, this means that we will not standardize all variables to have a variance of 1, and `empirical` to `TRUE`, which will assert that the empirical covariance matrix is equal to the population covariance matrix.

The output matrix of the tested function and the adjacency matrix can now be used to evaluate how well that method performed. Given the adjacency matrix A and the predicted matrix P, we can get the following two arrays: $A' = \langle a_{ij} \rangle \forall i > j$, and $P' = \langle p_{ij} \rangle \forall i > j$. Then we can calculate the ROC curve using A' and P'. The code for this is the function `roc.for.matrix` in the codelisting in appendix 8.2.2. Then we can use the function `auc` in `R` to calculate the area under the curve.

First, we compare ND, the precision matrix, partial correlation (from the package `corpcor`) and graphical lasso (from the `R` package `glasso`).

Now, we wonder if the sparseness of the DAGs has any influence on the performance of different algorithms. For this, we made a graph with p as variable. In this, we can see that for higher values of p, network deconvolution performs better than the other methods. The results of this can be seen in figure 6.

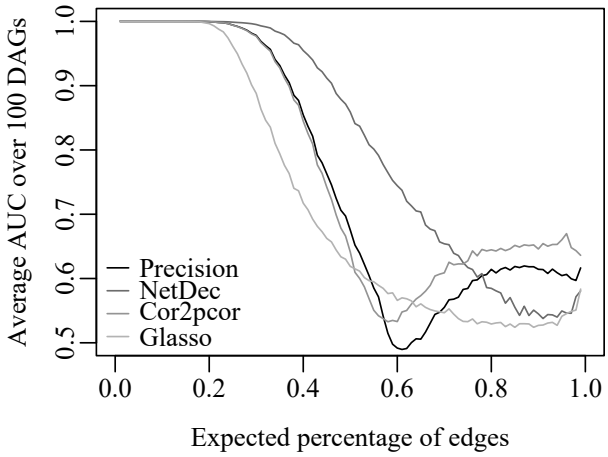


Figure 6: Calculating the AUC score over 100 different random DAGs per value of p.

After this we see that ND performs better than other methods, especially when the SEMs used to generate the data sets have more edges. This is interesting, because the only difference between ND and the precision matrix is the normalization. This means that normalization must be the main reason why ND performs well.

5.3 Normalization

In the previous section, we have seen that ND empirically outperformed the precision matrix. We know that these two methods are mathematically equivalent to each other, except for the normalization step performed in ND. Here we therefore ask: can we apply the normalization step of ND to other methods as well? Further, is the scaling value used for normalization in ND the optimal value, or is it possible to improve ND performance even more by using a different, manually tuned value?

Normalization is performed by multiplying the off-diagonal elements of the input matrix with a scalar based on the eigenvalues, as can be seen in definition 3.15. We wondered if this automatically chosen value leads to optimal performance. To investigate this, we ran ND on data generated from structural equation models at different sparsity levels and with different normalization factors α, as shown in figure 7. The other settings are the same as used in the simulation in subsection 5.2. Our results show that normalization factors between 0.4 and 0.6 work well for different levels of sparsity.

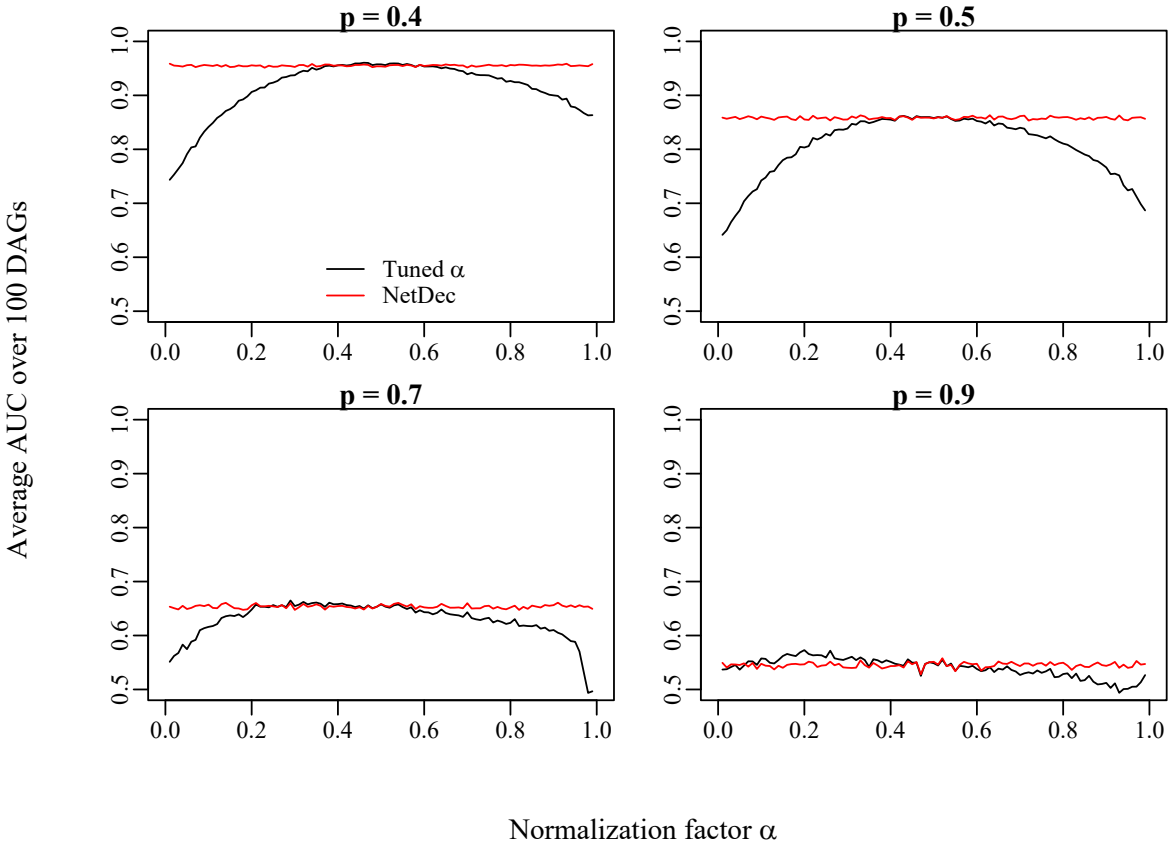


Figure 7: Performance of ND using varying normalization values α at four different sparsity levels. Note that the red lines are not dependent on α, they are supposed to be a horizontal line. However, due to randomness, they are not completely flat.

Now we compare ND to precision, partial correlation and graphical lasso applied to $0.5 \oslash C$, where C is the correlation matrix as input. As shown in figure 8, the scores of `cor2pcor` and the precision matrix increased after normalization, with normalization performing even better than ND. Note that the lines of `Glasso` and `Glasso` $\oslash$ 0.5 are overlapping, and so are the lines of ND and Precision $\oslash$ 0.5. The other settings are the same as in subsection 5.2.

In figure 9, we look at the precision matrix with $\alpha \oslash C$, $\alpha \in \{0.1, 0.3, 0.5, 0.7, 0.9.\}$, varying the parameter $p \in \{0.2, 0.4, 0.5\}$, and fixing $\beta = 0.15, N = 40$. Here we see that for $\alpha = 0.5$, the precision matrix performs as well as ND, and for other values, it performs worse.

We look at different values of β (see figure 10), with $\beta \in \{0.1, 0.3, 0.5\}$, with $N = 40, p = 0.4$. Here we also see that in all cases, the precision of $0.5 \oslash C$ performs similarly to ND.

Finally, in figure 11, we look at graphs with varying graph size $N \in \{15, 25, 35\}$ nodes, fixing $\beta = 0.15, p = 0.4$. Here we see that the precision matrix on $0.5 \oslash C$ and on $0.4 \oslash C$ performs similarly to ND.

All the underlying histograms are created in the manner way as the simulations in section 5.2, but with only three point from the x-axis as bars. This is because too many lines would make the plot unclear.

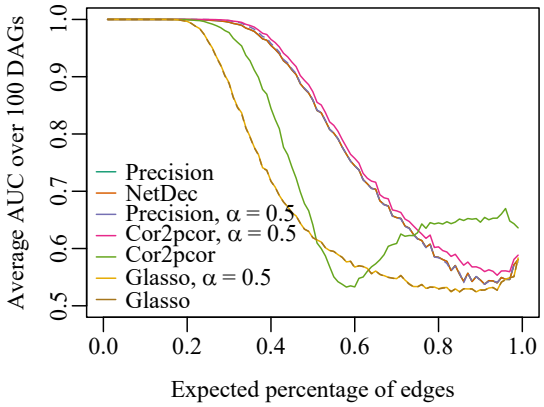


Figure 8: Network Deconvolution compared to partial correlation, graphical lasso and precision matrix.

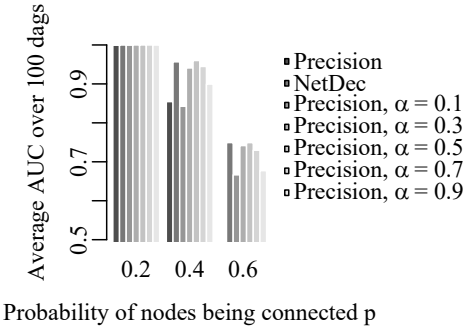


Figure 9: Network Deconvolution and precision matrix with variable network density p.

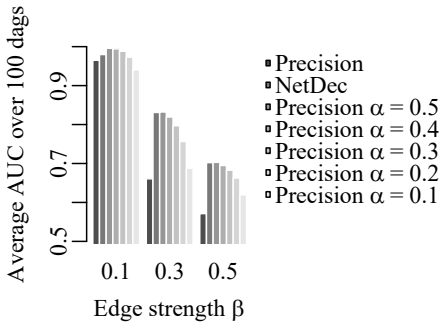


Figure 10: Network Deconvolution and precision matrix with variable interaction strength β.

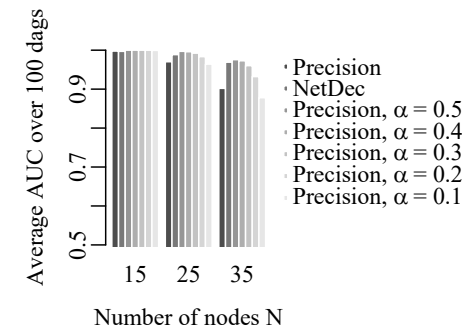


Figure 11: Network Deconvolution and precision matrix with variable network size N.

Note that on all graphs we generated, the value for normalization calculated with the formula in definition 3.14 ranged between 0.5 and 0.55. This would explain why the best value for normalization always was around 0.5. We did not test ND on larger DAGs, and we also did not use varying edge strengths. This might change the value for the calculated α. However, handpicking the value of α does not seem to perform better than simply using ND. Therefore, based on our results, we would recommend using the automatically determined value. In this chapter, we have seen that normalization can be used for other methods, improving their results. Especially for partial correlation and the precision matrix, results seem to improve.

6 Relation to Shrinkage

In the previous sections, we saw empirically that normalization substantially improves the performance of ND. The reason for the normalization step in ND is to assert that the transitive closure could not have converged to the observed matrix if the eigenvalues were too large. However, we have seen that ND without normalization would have been a valid method, because of its equivalence to the precision matrix. In this section, we aim to provide an alternative explanation of what normalization through off-diagonal scaling of the input matrix represents. Therefore, we will use Bayesian networks to represent this operations.

Specifically, we will construct two networks, one represents the process generating the correlation matrix and the other represents the normalized form of the matrix. Let us first regard the following network:

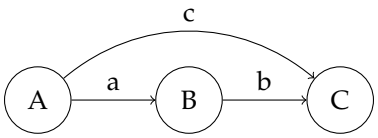


Figure 12: Complete DAG

Now, the correlation matrix will be

$$M = \begin{array}{c} \\ A \\ B \\ C \end{array} \begin{array}{c} \begin{array}{ccc} A & B & C \end{array} \\ \begin{pmatrix} 1 & a & a \cdot b + c \\ a & 1 & a \cdot c + b \\ a \cdot b + c & a \cdot c + b & 1 \end{pmatrix} \end{array}$$

If we expand this DAG to the following DAG:

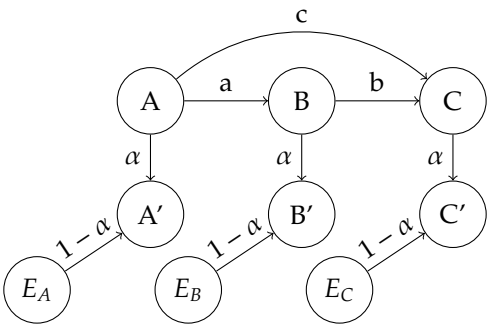


Figure 13: Added variables

Where A', B', C' represent the 'observed' equivalents of A, B, C and E_A, E_B, E_C represent measurement noise that affects A, B, C.

We will get the following correlation matrix for A, B, C, A', B', C':

$$M = \begin{array}{c} \\ A \\ B \\ C \\ A' \\ B' \\ C' \end{array} \begin{array}{c} \begin{array}{cccccc} A & B & C & A' & B' & C' \end{array} \\ \begin{pmatrix} 1 & a & a \cdot b + c & \alpha & \alpha a & \alpha(a \cdot b + c) \\ a & 1 & a \cdot c + b & \alpha a & \alpha & \alpha(a \cdot c + b) \\ a \cdot b + c & a \cdot c + b & 1 & \alpha(a \cdot b + c) & \alpha(a \cdot c + b) & \alpha \\ \alpha & \alpha a & \alpha(a \cdot b + c) & 1 & \alpha^2 a & \alpha^2(a \cdot b + c) \\ \alpha a & \alpha & \alpha(a \cdot c + b) & \alpha^2 a & 1 & \alpha^2(a \cdot c + b) \\ \alpha(a \cdot b + c) & \alpha(a \cdot c + b) & \alpha & \alpha^2(a \cdot b + c) & \alpha^2(a \cdot c + b) & 1 \end{pmatrix} \end{array}$$

If we restrict this matrix to only A', B', C', we will get the following matrix:

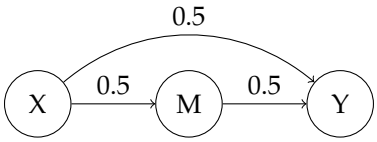


Figure 14: Example DAG.

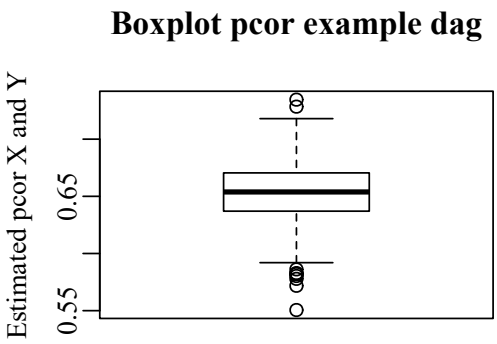


Figure 15: Estimating $X \sim Y$: Boxplot after generating 1000 SEMs

$$M' = \begin{array}{c} \\ A' \\ B' \\ C' \end{array} \begin{array}{c} \begin{array}{ccc} A' & B' & C' \end{array} \\ \begin{pmatrix} 1 & \alpha^2 a & \alpha^2(a \cdot b + c) \\ \alpha^2 a & 1 & \alpha^2(a \cdot c + b) \\ \alpha^2(a \cdot b + c) & \alpha^2(a \cdot c + b) & 1 \end{pmatrix} \end{array}$$

And here we can see that $\alpha^2 \oslash M = M'$.

So the off-diagonal multiplication with factor α is equivalent to assuming that every variable is affected by a measurement error of strength $1 - \alpha$, and correcting the estimated correlations by this assumed measurement error. However, this insight in itself does not yet explain why this multiplication leads to improved performance – in our simulated data, for example, we do not introduce measurement error into our variables.

It is known that estimates of partial correlations can be positively biased, take for example figure 14, of which we simulate the Structural Equation Model (SEM) shown in Figure 14 1000 times using `simulateSEM` from the `Dagitty` package, and then test the partial correlation between X and Y with M as mediator using the function `pcor.test`. This will give us the boxplot in figure 15. We can see that the median of the estimated correlations is about 0.65, even when it should be 0.5. Hence, the estimate is biased. The code for generating this boxplot can be found in appendix 8.2.3.

Shrinkage describes estimators that improve a simple estimate (such as a maximum likelihood estimate) by combining it with other information. This is done to reduce the effects from sampling variation. For several methods, it is known that shrinkage improves the results. For `cor2pcor`, shrinkage is said to amongst others improve the results and also make sure that the input matrices are always positive definite [SORZ+17]. However, shrinkage as used here is done by shrinking towards the identity matrix. Also, to use shrinkage, one needs to set a shrinkage parameter λ. An attractive feature of shrinkage such as implemented by ND is that it comes with an automated procedure to estimate this parameter from the eigenvalues of the input matrix.

7 Conclusion

Network deconvolution (ND) is a method for recovering direct interactions, given an input matrix of associations between variables. It was originally proposed that this method works by calculating the transitive reduction; the inverse of the transitive closure. However, this motivation appears at odds with the fact that many forms of interactions are non-transitive. In this research, we show that ND indeed works well on different datasets even when this assumption is violated, but because of a different reason than previously thought.

First, we tested ND on small DAGs, one which is transitive and one which is not. This confirmed our expectation that primitive ND, like the precision matrix, works best on non-collider cases; i.e. cases where relations are transitive. With this result, we proved primitive ND to be equivalent to the precision matrix. We also saw that the effect of normalization is different from linear scaling, and can introduce false negatives. We tested ND on the co-authorship dataset as used in [FMMK13], and compared it to the precision matrix and partial correlation, and we saw that all three methods performed very similarly; their AUC scores showed no substantial difference.

We then compared ND to other methods on datasets that we generated through Gaussian-linear Bayesian networks. These generating networks had varying levels of sparsity, varying edge strengths and varying numbers of nodes. Here we saw that ND outperformed other methods on datasets generated through dense networks. Given that we proved that the only difference between ND and the precision matrix is the normalization step, this superior performance of ND must be due to the normalization step alone.

We therefore tested if we could also apply normalization to partial correlation and graphical lasso. For the partial correlation this indeed improved the results, especially for matrices generated through dense networks. For graphical lasso, the results did not change substantially.

We found empirically that the normalization factor chosen through ND was optimal for all cases we tested. We did this by trying various different normalization factors on datasets generated through networks of varying sparsity. There were no manually chosen factors that outperformed the ND-determined factor, but the best value was not unique. We discovered that a normalization factor of around 0.5 seemed to work as well as the factor calculated with ND, in all cases which we tested. However, the normalization factor determined by ND ranged between 0.5 and 0.55 in the DAGs simulated. We do not yet know if this is due to the structure of our generated DAGs, or perhaps due to the structure of correlation matrices in general. The value calculated in ND depends on the largest positive or the smallest negative eigenvalue. Since correlation matrices have certain properties, like being positive definite and every value being between -1 and 1, this could maybe be an indicator of the range of eigenvalues.

Alternatively to the transitive reduction idea, ND can be explained using a Bayesian network in which the normalized input can be visualized as the correlation matrix of nodes which are directly connected to nodes of the original network. In such a network, the variables of the input matrix would be assumed to be noisy versions of the underlying original variables. When determining partial correlations, measurement error would lead to overestimate the partial correlation, which needs to be corrected for; this idea is similar to shrinkage.

We think ND can also be applied to Mendelian randomization [DSH14] (MR), which is a method to determine causal effects between variables when there are unknown co-founding factors. MR is basically an instrumental variable method where genetic variants are used as instruments. MR is thought to mimic a randomized trial in cases where this is not possible. Because causal effects are transitive, we would expect ND to work well on an input matrix generated from pairwise MR estimates of total causal effects, even though the role of normalization in this case would need to be clarified in future work.

For usage, our recommendation would be that especially for datasets where the underlying direct interaction networks are expected to be dense, ND would be a good choice. Not only does it appear to perform better than alternative methods like the graphical lasso, it is also computationally more efficient, since it only consists of elementary matrix operations. We would not recommend to manually tune the normalization parameter, since the generated value performs well, so tuning would only take extra time and the specific choice of the parameter would be hard to justify.

Our theoretical analysis in this work did not yet reach very far. It would be interesting to prove mathematically that the chosen factor for normalization is indeed optimal. In [SS05] it is shown mathematically that their method of shrinkage is optimal. This would be a good addition for further understanding this method. A limitation of our empirical research is the graph structures we used to test the methods on. We do not know how representative Bayesian Networks with equal edge strength for all edges, and with randomly generated structure, are for datasets encountered in real life. It would be interesting to extend our analysis with more variation in the generated DAGs. For example, negative edge weights could be used, different edge weights

could be applied to different edges in the DAG. Another thing we could test is how well these methods perform when tested on correlation matrices generated through different network structures, for example scale-free networks [BA99] or small-world networks [WS98]. Lastly, it would be interesting to compare ND to algorithms created for a specific structure in a dataset, for example Direct-Coupling Analysis as proposed in [MPL+11], a method for predicting protein structures.

References

[BA99] Albert-László Barabási and Réka Albert. Emergence of Scaling in Random Networks. *Science*, 286(5439):509–512, 1999.

[Dob04] Annette Dobson. 1. the oxford dictionary of statistical terms. yadolah dodge (ed.), oxford university press, oxford, 2003, hardcover. no. of pages: 506. price: GBP 25.00. ISBN 0-19-850994-4. *Statistics in Medicine*, 23(11):1824–1825, 2004.

[DSH14] George Davey Smith and Gibran Hemani. Mendelian randomization: genetic anchors for causal inference in epidemiological studies. *Human molecular genetics*, 23(R1):R89–R98, 2014.

[FHT07] J. Friedman, T. Hastie, and R. Tibshirani. Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*, 9(3):432–441, dec 2007.

[FMMK13] Soheil Feizi, Daniel Marbach, Muriel Médard, and Manolis Kellis. Network deconvolution as a general method to distinguish direct dependencies in networks. *Nature Biotechnology*, 31(8):726–733, jul 2013.

[LMFH+15] G. Lima-Mendez, K. Faust, N. Henry, J. Decelle, S. Colin, F. Carcillo, S. Chaffron, J. C. Ignacio-Espinosa, S. Roux, F. Vincent, L. Bittner, Y. Darzi, J. Wang, S. Audic, L. Berline, G. Bontempi, A. M. Cabello, L. Coppola, F. M. Cornejo-Castillo, F. d'Ovidio, L. De Meester, I. Ferrera, M.-J. Garet-Delmas, L. Guidi, E. Lara, S. Pesant, M. Royo-Llonch, G. Salazar, P. Sanchez, M. Sebastian, C. Souffreau, C. Dimier, M. Picheral, S. Searson, S. Kandels-Lewis, G. Gorsky, F. Not, H. Ogata, S. Speich, L. Stemmann, J. Weissenbach, P. Wincker, S. G. Acinas, S. Sunagawa, P. Bork, M. B. Sullivan, E. Karsenti, C. Bowler, C. de Vargas, and J. Raes and. Determinants of community structure in the global plankton interactome. *Science*, 348(6237):1262073–1262073, may 2015.

[Mad94] David Madigan. Graphical models in applied multivariate statistics, by j. whittaker, john wiley & sons, new york, 1990, 448 pp. *Networks*, 24(2):125–125, mar 1994.

[MCS+11] Debora S. Marks, Lucy J. Colwell, Robert Sheridan, Thomas A. Hopf, Andrea Pagnani, Riccardo Zecchina, and Chris Sander. Protein 3d structure computed from evolutionary sequence variation. *PLoS ONE*, 6(12):e28766, dec 2011.

[MPL+11] F. Morcos, A. Pagnani, B. Lunt, A. Bertolino, D. S. Marks, C. Sander, R. Zecchina, J. N. Onuchic, T. Hwa, and M. Weigt. Direct-coupling analysis of residue coevolution captures native contacts across many protein families. *Proceedings of the National Academy of Sciences*, 108(49):E1293–E1301, nov 2011.

[MWD+16] Bin Ma, Haizhen Wang, Melissa Dsouza, Jun Lou, Yan He, Zhongmin Dai, Philip C Brookes, Jianming Xu, and Jack A Gilbert. Geographic patterns of co-occurrence network topological features for soil microbiota at continental scale in eastern china. *The ISME Journal*, 10(8):1891–1901, jan 2016.

[SORZ+17] Juliane Schafer, Rainer Opgen-Rhein, Verena Zuber, Miika Ahdesmaki, A. Pedro Duarte Silva, and Korbinian Strimmer. *corpcor: Efficient Estimation of Covariance and (Partial) Correlation*, 2017. R package version 1.6.9.

[SS05] Juliane Schäfer and Korbinian Strimmer. A shrinkage approach to large-scale covariance matrix estimation and implications for functional genomics. *Statistical Applications in Genetics and Molecular Biology*, 4(1), jan 2005.

[Tao16] Terence Tao. *Analysis I*. Springer Singapore, 2016.

[The18] The Sage Developers. *SageMath, the Sage Mathematics Software System (Version 8.1)*, 2018. `http://www.sagemath.org`.

[TvG+16] Johannes Textor, Benito van der Zander, Mark S Gilthorpe, Maciej Liśkiewicz, and George TH Ellison. Robust causal inference using directed acyclic graphs: the R package 'dagitty'. *International Journal of Epidemiology*, 45(6):1887–1894, 2016.

[WS98] Duncan J. Watts and Steven H. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393(6684):440–442, 1998.

8 Appendix

8.1 Analytical Investigation of ND

Here, we solve ND analytically for further simple covariance matrices that are generated from SEMs; see Section 4

Network Deconvolution is generally done on a matrix with 0 on the diagonal. Here we look at some examples with the original implied covariance matrix as input matrix, which has a 1 on the diagonal. At first, it was not clear from the ND paper whether there should be a 1 or a 0 on the diagonal, so we tested both. However, later, we found out that the authors used 0 on the diagonal in their own code, so that is what we used in the main text.

8.1.1 Collider Case; 1 on Diagonal

For the SEM with graph

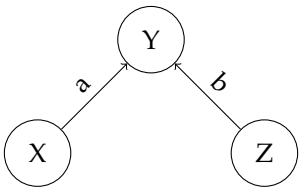


Figure 16: Collider

with the implied covariance matrix

$$C = \begin{array}{c} \begin{matrix} X & Y & Z \end{matrix} \\ \begin{pmatrix} 1 & a & 0 \\ a & 1 & b \\ 0 & b & 1 \end{pmatrix} \end{array} \begin{matrix} X \\ Y \\ Z \end{matrix}$$

we find, after primitive devoncolution, the following matrix:

$$D(C) = \frac{1}{2(a^2+b^2-4)} \begin{array}{c} \begin{matrix} X & Y & Z \end{matrix} \\ \begin{pmatrix} 2a^2+b^2-4 & 2a & ab \\ -2a & 2(a^2+b^2-2) & -2b \\ ab & -2b & a^2+2b^2-4 \end{pmatrix} \end{array} \begin{matrix} X \\ Y \\ Z \end{matrix}$$

This matrix has a zero between X and Y when $a \cdot b$ is zero, thus when $a = 0 \vee b = 0$. So we see that using the covariance matrix C as input, instead of $C - I$, does not improve results. These results are the same as when having 0 on the diagonal.

8.1.2 Fork Case; 1 on Diagonal

For the SEM with graph with covariance matrix

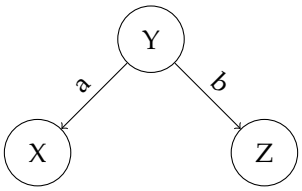


Figure 17: Fork

$$C = \begin{array}{c} \begin{matrix} X & Y & Z \end{matrix} \\ \begin{pmatrix} 1 & a & a \cdot b \\ a & 1 & b \\ a \cdot b & b & 1 \end{pmatrix} \end{array} \begin{matrix} X \\ Y \\ Z \end{matrix}$$

we found the following matrix with primitive deconvolution:

$$D(C) = \frac{1}{2\,(a^2 + b^2 - 4)} \begin{array}{c} \begin{matrix} X & Y & Z \end{matrix} \\ \begin{pmatrix} 2\,a^2 + b^2 - 4 & ab^2 - 2\,a & -ab \\ ab^2 - 2\,a & -\left(\left(a^2 - 2\right)b^2 - 2\,a^2 + 4\right) & \left(a^2 - 2\right)b \\ -ab & \left(a^2 - 2\right)b & a^2 + 2\,b^2 - 4 \end{pmatrix} \end{array} \begin{matrix} X \\ Y \\ Z \end{matrix}$$

This matrix has zeroes when $a = \sqrt{2}$ between Y and Z or at $b = \sqrt{2}$ between X and y; however, these are not valid values in a covariance matrix. So we see that using the covariance matrix C as input, instead of $C - I$, makes the results worse. We would find a true negative between X and Z when using $C - I$ as input, as seen in example 4.2. However, when using C with a 1 on the diagonal, we do not find this zero value anymore.

8.1.3 Complete DAG; 0 on Diagonal

For the Bayesian network with graph

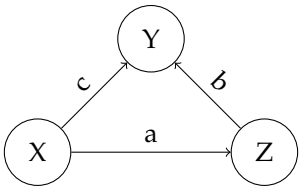


Figure 18: Complete DAG

we found the following covariance matrix:

$$C = \begin{array}{c} \begin{matrix} X & Y & Z \end{matrix} \\ \begin{pmatrix} 1 & a & a \cdot b + c \\ a & 1 & a \cdot c + b \\ a \cdot b + c & a \cdot c + b & 1 \end{pmatrix} \end{array} \begin{matrix} X \\ Y \\ Z \end{matrix}$$

For this, we found the following primitive deconvolved matrix:

$$D(C) = \frac{1}{det} \begin{array}{c} \begin{matrix} X & Y & Z \end{matrix} \\ \begin{pmatrix} 2\,(a^3 - 2\,a)bc + (a^2 - 2)c^2 - 2\,a^2 - b^2 + 4 & -(ab^2 + (a^2 + 1)bc + ac^2 - 2\,a) & ab - (a^2 - 2)c \\ -(ab^2 + (a^2 + 1)bc + ac^2 - 2\,a) & (a^2 - 2)b^2 + 2\,(a^3 - 2\,a)bc - 2\,a^2 - c^2 + 4 & -((a^2 - 2)b - ac) \\ ab - (a^2 - 2)c & -((a^2 - 2)b - ac) & 2\,(a^3 - 3\,a)bc - a^2 - 2\,b^2 - 2\,c^2 + 4 \end{pmatrix} \end{array} \begin{matrix} X \\ Y \\ Z \end{matrix}$$

where $det = 2\left(\left(a^3 - 3\,a\right)bc - a^2 - b^2 - c^2 + 4\right)$

In this matrix we can find multiple roots, for example for example at

$$(ab - (a^2 - 2)c)$$

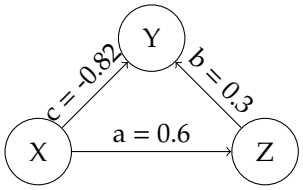
, the numerator for the effect between X and Z, we find roots at $c = \frac{ab}{-2+a^2}$.
In the numerator of the effect between X and Z, we find multiple roots, among which

$$c = -\frac{(1 + a^2)b + \sqrt{b^2 + a^4b^2 - 2a^2(-4 + b^2)}}{2a}$$

For the numerator of the effect between Y and Z, we find the root

$$c = \frac{(-2 + a^2)b}{a}$$

One example of this is $a = 0.6; b = 0.3, c = -0.82$, which will give 0 as deconvolved value between Y and Z. This will give the following network: with the following covariance matrix:



$$C = \begin{array}{c} \begin{matrix} X & Y & Z \end{matrix} \\ \begin{pmatrix} 1 & 0.6 & -0.64 \\ 0.6 & 1 & -0.192 \\ -0.64 & -0.192 & 1 \end{pmatrix} \end{array} \begin{matrix} X \\ Y \\ Z \end{matrix}$$

This will give us the following deconvolved matrix:

$$C = \begin{array}{c} \begin{matrix} X & Y & Z \end{matrix} \\ \begin{pmatrix} 0.61 & -1.65 \cdot 10^{-01} & 1.78 \cdot 10^{-01} \\ -0.16 & 5.49 \cdot 10^{01} & -8.33 \cdot 10^{-17} \\ 0.18 & -1.94 \cdot 10^{-16} & 5.57 \cdot 10^{-01} \end{pmatrix} \end{array} \begin{matrix} X \\ Y \\ Z \end{matrix}$$

However, the deconvolved matrix after normalization will be:

$$\tilde{D}(C) = \begin{array}{c} \begin{matrix} X & Y & Z \end{matrix} \\ \begin{pmatrix} 0.72 & -0.14 & 0.15 \\ -0.14 & 0.69 & 0.02 \\ 0.15 & 0.015 & 0.70 \end{pmatrix} \end{array} \begin{matrix} X \\ Y \\ Z \end{matrix}$$

Where the zeroes seem to disappear.

8.1.4 Complete DAG; 0 on Diagonal

Let us use figure 18 again, and consider using the matrix

$$C' = C - I = \begin{array}{c} \begin{matrix} X & Y & Z \end{matrix} \\ \begin{pmatrix} 0 & a & a \cdot b + c \\ a & 0 & a \cdot c + b \\ a \cdot b + c & a \cdot c + b & 0 \end{pmatrix} \end{array} \begin{matrix} X \\ Y \\ Z \end{matrix}$$

as an input matrix, then we will get

$$D(C') = \begin{array}{c} \begin{matrix} X & Y & Z \end{matrix} \\ \begin{pmatrix} \frac{2a^3bc+a^2b^2+\left(2a^2-1\right)c^2-a^2}{(a^2-1)b^2+2(a^3-a)bc+(a^2-1)c^2-a^2+1} & -\frac{ab^2+\left(a^2+1\right)bc+ac^2-a}{(a^2-1)b^2+2(a^3-a)bc+(a^2-1)c^2-a^2+1} & -\frac{c}{2abc+b^2+c^2-1} \\ -\frac{ab^2+\left(a^2+1\right)bc+ac^2-a}{(a^2-1)b^2+2(a^3-a)bc+(a^2-1)c^2-a^2+1} & \frac{2a^3bc+a^2c^2+\left(2a^2-1\right)b^2-a^2}{(a^2-1)b^2+2(a^3-a)bc+(a^2-1)c^2-a^2+1} & -\frac{b}{2abc+b^2+c^2-1} \\ -\frac{c}{2abc+b^2+c^2-1} & -\frac{b}{2abc+b^2+c^2-1} & \frac{2abc+b^2+c^2}{2abc+b^2+c^2-1} \end{pmatrix} \end{array} \begin{matrix} X \\ Y \\ Z \end{matrix}$$

as deconvolved matrix. We here see that the effect of a on c becomes zero when c is zero. The effect of a on b is zero when

$$a = -\frac{b^2 + c^2 + \sqrt{b^4 + c^4 - 2\,(b^2+1)c^2 - 2\,b^2 + 1} - 1}{2\,bc}$$

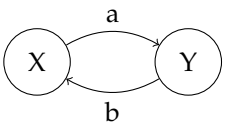
or when

$$a = -\frac{b^2 + c^2 - \sqrt{b^4 + c^4 - 2\,(b^2+1)c^2 - 2\,b^2 + 1} - 1}{2\,bc}\,.$$

This will be the case when, for example, $a = 0.3907, b = 0.5, c = 0.4$.

8.1.5 Small Cyclic Case; 0 on Diagonal

Let us consider a smaller cyclic case, for example:



This will give us the following covariance matrix (minus one):

$$C' = C - I = \begin{array}{c} \begin{matrix} X & Y \end{matrix} \\ \begin{pmatrix} 0 & \frac{a+b}{ab+1} \\ \frac{a+b}{ab+1} & 0 \end{pmatrix} \end{array} \begin{matrix} X \\ Y \end{matrix}$$

And this will result in the following primitive deconvolved matrix:

$$D(C) = \begin{array}{c} \begin{matrix} X & Y \end{matrix} \\ \begin{pmatrix} -\frac{(a+b)^2}{(a^2-1)(b^2-1)} & \frac{(a+b)(ab+1)}{(a^2-1)(b^2-1)} \\ \frac{(a+b)(ab+1)}{(a^2-1)(b^2-1)} & -\frac{(a+b)^2}{(a^2-1)(b^2-1)} \end{pmatrix} \end{array} \begin{matrix} X \\ Y \end{matrix}$$

8.1.6 Cyclic Case

For the Bayesian network with graph

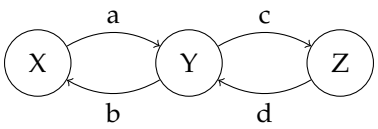


Figure 19: Cyclic case

we found the following covariance matrix by first defining the adjacency matrix

$$L = \begin{array}{c} \begin{matrix} X & Y & Z \end{matrix} \\ \begin{pmatrix} 0 & a & 0 \\ b & 0 & c \\ 0 & d & 0 \end{pmatrix} \end{array} \begin{matrix} X \\ Y \\ Z \end{matrix}$$

Then we calculate $\Lambda = (I - L)^{-1}$ and $\Phi = (((\Lambda^T)^2)^{-1}) \cdot \begin{pmatrix} 1 & 1 & 1 \end{pmatrix}$. Then $\Sigma = \Lambda^T \cdot \Phi \cdot \Lambda$, in which Σ is the covariance matrix. This covariance matrix is calculated with `Sage`, and is too large to fit to fit on the page.

Next we calculate the primitive deconvolved matrix by $D = \Sigma \cdot (\Sigma + 1)^{-1}$. Again, this matrix is very large and does not fit within the margins of the page. We solved for zero values using `Sage`. This matrix contains zeroes when for example

$$c = \frac{a \cdot b - 1}{d}$$

or

$$a = \frac{c \cdot d + 1}{b}$$

so for example when $a = 0.65; b = 0.8; c = -0.6; d = 0.8$, or when $a = 0.5, b = 0.5, c = 0.8; d = -0.9375$, we will find a 0 for the relation between X and Z.

8.2 Code Listings

8.2.1 Sage Code

Code for example 4.1:

```
var('a b')
# For the fork case
C = matrix([[0,a,a*b],[a,0,b],[a*b,b,0]])
D = (C*~(C+matrix.identity(3))).apply_map(lambda x: x.simplify_full())

# For the collider case
C = matrix([[0,a,0],[a,0,b],[0,b,0]])
D = (C*~(C+matrix.identity(3))).apply_map(lambda x: x.simplify_full())
```

Code for example 4.5:

```
var('a b c d e f g q')
M = matrix([[a,b,0],[c,d,e],[0,f,g]])
C = (~(M - matrix.identity(3))*-M).apply_map(lambda x: x.simplify_full())
D2 = (q*C*~(q*C+matrix.identity(3))).apply_map(lambda x: x.simplify_full())
```

8.2.2 R Code

R code used in chapter 5.2. This code is used to generate figures like figure 6. Note that in this script, we only compare deconvolution and the precision matrix. If the reader wants to compare more methods, please add these yourself in the place specified.

```
library(corpcor)
library(dagitty)
library(pROC)

adjacencyMatrix <- function(g) {
  M <-
    sapply(names(g), function(x)
      (names(g) %in% adjacentNodes(g, x)))
  rownames(M) <- names(g)
  return(M)
}

dir <- function(x)
  x %*% solve(x + diag(nrow(x)))

normalize_beta <- function(x, beta = 0.95) {
  # Normalization according to chapter 1.6 of the supplementary notes.
  xe <- eigen(x)
  max_e <- max(xe$values)
  min_e <- min(xe$values)
  alpha <- min(beta / ((1 - beta) * max_e),-beta / ((1 + beta) * min_e))
  return(x * alpha)
}

deconvolve <- function(C,
                       normalize = TRUE,
                       alpha = 1) {
  if (normalize &&
      alpha == 1)
    convmatrix <- dir(normalize_beta(C - diag(nrow(C))))
  else if (alpha == 1)
    convmatrix <- dir(C - diag(nrow(C)))
  else
    convmatrix <- dir(alpha * (C - diag(nrow(C))))
  return(convmatrix)
```

```
}

roc.for.matrix <-
  function(network.weights,
           adjacency.matrix,
           filter.matrix = NULL) {
    require(pROC)
    if (is.null(filter.matrix)) {
      filter.matrix <- upper.tri(network.weights)
    }
    x <- abs(network.weights[filter.matrix])
    y <- adjacency.matrix[filter.matrix]
    roc(y, x)
  }

M = 100
K = 100
# Different values for probabilities, with min = 0.01 and max = 1
p_values = seq(0.01, 1, length.out = M)
# Creating a matrix to store everything in
# n_col = amount of lines plotted/functions tested
ncol = 2
average_auc = matrix(NA, nrow = M, ncol = n_col)

for (j in c(1:M)) {
  aucs = matrix(NA, nrow = K, ncol = n_col)
  for (i in c(1:K)) {
    # Amount of nodes
    N <- 40
    # Generate the DAG
    g <- randomDAG(N, p_values[j])
    # Simulate the strucural equation model
    X <- simulateSEM(
      g,
      b.default = 0.15,
      standardized = FALSE,
      empirical = TRUE
    )
    # Calculate the correlation matrix
    C <- cor(X)

    # Here we compare MD to MC

    # Deconvolve
    MD <- deconvolve(C)
    # Precision
    MC <- solve(C)


    AM <- adjacencyMatrix(g)

    # Skip cases in which everything is connected
    if (all(AM[upper.tri(AM)]))
      next

    # If you want to test more methods, please append
    # auc(roc.for.matrix(YOURMETHOD(C), AM)) to the following list:

    aucs[i, ] = c(auc(roc.for.matrix(MC, AM)), auc(roc.for.matrix(MD, AM)))
  }
  average_auc[j, ] = colMeans(aucs, na.rm = T)
  cat(paste(c("At␣number␣", j, "\n")))
}

# Plotting
par(
  mfrow = c(1, 1),
  mar = c(4.1, 4.1, 0.1, 0.1),
  xpd = TRUE,
  mgp = c(2, .5, 0),
  family = 'serif'
)
plot(
  p_values,
  average_auc[, 1],
  type = 'l',
```

```
  xlab = "Expected␣percentage␣of␣edges",
  ylab = "Average␣AUC␣over␣100␣DAGs",
  xlim = c(0, 1),
  ylim = c(0.5, 1),
  col = 1,
  lty = 1
)
lines(p_values, average_auc[, 2], col = 2)
legend(
  "bottomright",
  inset = c(0.3, 0),
  c("Precision", "NetDec"),
  col = c(1, 2),
  lty = 1,
  bty = "n",
  y.intersp = 1.5
)
```

8.2.3 R Code for Boxplot

The following code is used to create the boxplot in section 6.

```
library(dagitty)
library(ppcor)
# Simulating the DAG X -> M <- Y; X -> Y
pc <- function() {g <- simulateSEM("dag␣{␣x␣->␣m␣[beta=0.5]␣m␣->␣y␣[beta=0.5]␣x␣->␣y␣[beta=.5]␣}" )

# Estimating the partial correlation of X and Y through M
pcor.test(g$x, g$y, g$m)$estimate}

# Doing this 1000 times and creating a boxplot
boxplot(replicate( 1000, pc() ),
ylab  = "Estimated␣pcor␣X␣and␣Y",
main = "Boxplot␣pcor␣example␣dag")
```