UCLouvain

FACULTE DES SCIENCES

Faculté des sciences

Group differences and similarities with multiple graph models

Author : Abduelhakem Ben Addi
Supervisor : Prof. Eugen Pircalabelu

Reader : Prof. Christian Hafner
Academic year 2021-2022

Masters degree in Statistics (orientation générale, à finalité spécialisée)

Preface

This dissertation has been prepared in partial fulfilment of the requirements for the Masters degree in Statistics delivered by the UCLouvain.

Acknowledgement

Computational resources have been provided by the *Consortium des Équipements de Calcul Intensif* (CÉCI), funded by the *Fonds de la Recherche Scientifique de Belgique* (F.R.S.-FNRS) under Grant No. 2.5020.11 and by the Walloon Region.

Many people have contributed to the completion of this dissertation.
I would like to first thank my supervisor Prof Eugen Pircalabelu for his guidance, extensive review of this manuscript and help throughout this master thesis. I deeply appreciate the encouragement during the difficult times. This work literally wouldn't have been possible without you.

Je suis reconnaissant envers Marion pour sa patience et son soutient. Beaucoup de sacrifices ont été nécessaires tout au long de ce master mais tout bonne chose à une fin. Nous pouvons réfléchir au master suivant pour plus de fun.

Enfin, j'aimerais remercié ma famille et mes amis pour leur soutient sur cette longue période d'étude.

Executive summary

The understanding of any large and complex system is bound to the unravelling of its network of relations. While some networks are readily uncovered others necessitate the processing of probabilistic data. The study of relationships through (probabilistic) graphical models has gained a lot of attraction due to its intuitive way to model structure dependencies.
The complexity of the task is greatly increased when multiple classes of graphs must be estimated and compared. These groups of networks might be globally similar but also differ to some extent. To add challenge to the complexity, large networks must often be estimated using high-dimensional data where the number of parameters p to estimate is much larger than the sample size n.

Robust statistical methodologies must be developed to address these many challenges. In the multivariate Gaussian framework, the precision matrix bears the useful property of encoding the conditional independence structure, *i.e.* off-diagonal zeros exclude a direct relation between two features of the networks. In high-dimensional settings, the maximum likelihood estimate of a covariance matrix can not be inverted to produce an estimate of the concentration matrix. Many penalized log-likelihood methods have been proposed in the literature in order to address those issues. However, none have fully addressed the outstanding issue: either the optimization problem was not convex or the emphasis was put on forcing the similarity between classes. An interesting approach proposed in the literature was the use, in the optimization problem, of one penalty to control the sparsity pattern and a second one to force similarity. We took advantage of that idea in this master thesis.

We introduced a new family of penalties and tested the performances of 4 of its members, of which one was published under the acronym GGL in Danaher *et al.* (2014). We studied the performances of our proposals on synthetic data simulated from underlying scale-free networks which mimic biological networks. Our simulation results showed our Proposals 1 and 3 were comparable to the GGL penalty in terms of accuracy, (differential) edges detection sensitivity and specificity. We confirmed that the GGL penalty also contributed to the sparsity while our proposals did not encourage it. Proposal 2 was less reliable in terms of algorithmic convergence and showed worse performances.
The regularization methods were also tested on a real-data set of microarray gene expression generated from lung epithelial cells obtained from 90 healthy patients and 97 cancer stricken patients. The results confirmed the similarities between Proposal 1 and 3 as most of their visualized sub-networks were identical. Despite, many differences in terms of network pattern, the vast majority of the edges modulated between classes were common to the GGL penalty, Proposals 1 and 3.

Contents

Abbreviations

Acronym	**Meaning**
ADMM	Alternating Direction of Multipliers Method
AIC	Akaike Information Criterion
BIC	Bayesian Information Criterion
CD	Coordinate Descent
cDNA	complementary DeoxyriboNucleic Acid
DE	Differential Edge
DGM	Directed Graphical Model
dKL	Kullback-Leibler divergence
DNA	DeoxyriboNucleic Acid
EL	Entropy Loss
FDR	False Discovery Rate
FGL	Fused Graphical Lasso
FL	Frobenius Loss
FNR	False Negative Rate
FPDE	False Positive Differential Edge
FPE	False Positive Edge
FPR	False Positive Rate
GM	Graphical Model
GGL	Grouped Graphical Lasso
GGM	Gaussian Graphical Model
LARS	Least Angle Regression and Shrinkage
lasso	Least absolute shrinkage and selection operator
MAD	Median Absolute Deviation
mRNA	messenger RiboNucleic Acid
MSE	Mean Squared Error
PGM	Probabilistic Graphical Models
RoC	Receiver operator Characteristic (curve)
SSE	Sum of Squared Errors
s.t.	subject to
TPDE	True Positive Differential Edge
TPE	True Positive Edge
TPR	True Positive Rate

Notations

Notation	Description
$\mathbf{M}^T$	Transpose of matrix $\mathbf{M}$
$\|\|\mathbf{M}\|\|_{\mathbf{1}}$	ℓ_1-norm of matrix $\mathbf{M} \in \mathbb{R}^{A \times B}$, *i.e.* $\sum_{ab} \|M_{a,b}\|$
$\|\|\mathbf{M}\|\|_{\mathbf{2}}$	(Spectral) ℓ_2-norm of matrix $\mathbf{M} \in \mathbb{R}^{A \times B}$, *i.e.* $\sqrt{\max \mathrm{eig}(\mathbf{M}^T\mathbf{M})}$
$\|\|\mathbf{M}\|\|_{\infty}$	ℓ_∞-norm of matrix $\mathbf{M} \in \mathbb{R}^{A \times B}$ *i.e.* $\max_{ab} \|M_{a,b}\|$
$\|\|\mathbf{M}\|\|_{\mathbf{F}}$	Frobenius norm of matrix $\mathbf{M} \in \mathbb{R}^{A \times B}$, *i.e.* $\sqrt{\mathrm{tr}(\mathbf{M}^T\mathbf{M})}$
$\mathrm{Sgn}(c)$	Sign of the scalar value c
$(c)_+$	Positive part, *i.e.* $\max(c, 0)$
$\mathrm{Soft}(c, \lambda)$	Soft thresholding of coefficient scalar c, *i.e.* $\mathrm{Sgn}(c)(\|c\| - \lambda)_+$
$\mathrm{tr}(\mathbf{M})$	Trace of matrix $\mathbf{M}$, *i.e.* $\sum_a M_{a,a}$
$\mathrm{vec}(\mathbf{M})$	vectorization of matrix $\mathbf{M}$
$\mathbf{M} \otimes \mathbf{N}$	Kronecker product of matrices $\mathbf{M}$ and $\mathbf{N}$

Chapter 1

Introduction

The last century has witnessed an accelerated progression of science achievements which materialized in improved and innovative technologies. These in turn allowed for enhanced capabilities to generate and store large amounts of data as well as the increase of computational power. A variety of applied fields such as system biology and image processing have been benefiting from these developments (James *et al.*, 2013; Kolaczyk, 2009).

The rapid growth of large dataset collections over the last decades in many scientific areas created a plethora of challenges for the fields of statistics and data science. Among those challenges, the study of high dimensional data, that is datasets containing a large number of variables for a limited number of observations, is of particular interest.

In this master thesis, we focused on the study of biological systems, in particular the genes expression circuits. Cellular pathways, *i.e.* cascades of events within the cells which regulate biological functions, are another field of interest in the community of biologists. The Krebs cycle involving series of chemical reactions to release stored energy, is one such example. Modelling the circuit has proved useful in understanding the minimal sub-networks fulfilling each a specific task, the direction of information flow in the network and the response to perturbations (Oliveira *et al.*, 2003).
Among the approaches to study biological functions is the investigation of individual gene activity, using different methods which are labour intensive and lack in global perspective. During the past few decades, a widespread approach was the generation of vast amount of information, through high throughput technologies, which permitted to study the biological systems in their (almost) entirety. Nevertheless, the statistical approach was mostly limited to gene expression profiling, which helped develop diagnostic tools (Spira *et al.*, 2007). The same approach has been exploited to identify cellular pathways involved in biological dis-regulation (Gustafson *et al.*, 2010). However, given the interconnectedness of the network, differentially expressed genes do not readily indicate which sub-network is affected. In that line of research, amongst others, the science of network or study of relations between objects has gained much attention in recent decades for its potential to generate working hypotheses to unravel the underlying mechanisms of biological functions.

The statistical approach to unravel these networks is the Graphical modelling.

The study of biological networks suffers many challenges. Among others is the need to study multiple classes (groups) of networks to identify defective sub-networks. It is expected that classes of comparable samples, *e.g.* collected from healthy and disease stricken patients, would be overall comparable in their networks and would differ on a small fraction of sub-networks. It is therefore of the utmost importance to estimate these networks as to preserve both the affected **and** unaffected sub-networks. The generation of an accurate picture of the networks and their differences would greatly facilitate the formulation of working hypotheses and therefore the study of underlying mechanisms responsible for biological dysfunction. Solutions to the joint estimation of multiple classes have been provided in the literature (Danaher *et al.*, 2014; Zhao *et al.*, 2014). However, these methods were either focused on preserving the similarities between networks, only allowing comparisons of networks by pairs or only providing a picture of the differential network and not the actual ones.

One objective of this master thesis, was to improve the existing methods of networks estimation as to discriminate well between two or multiple classes while maintaining the similarities. The steps taken to address our objective were the following:

- develop graph **estimation methods** that best describe similarities and differences between multiple classes;
- develop and implement an algorithm to estimate the model parameters;
- **evaluate the estimation method empirically** on simulated and real data to illustrate the benefits of the new procedure.

The manuscript is organised as follows. An overview of the Graphical models (GM) concepts is provided in Chapter 2. Probabilistic Graphical models, a category of GM which addresses biological networks, are presented in Section 2.2. Gaussian Graphical models which are the main focus of this work are discussed in Section 2.3. The two-step process of learning Graphical models is laid out in Chapter 3. Three proposals to address the statistical issue and their analytical solution are developed in Chapter 4. The performance of the proposals are investigated on synthetic datasets in Section 5.3.2. An application of the proposals to a real lung gene expression dataset, from the field of cancer research, is presented in Section 5.4. The discussion follows in Chapter 6.

Chapter 2

Introduction to Graphical Models

2.1 Concepts, definitions and notation

The relationships between objects are often of interest and sometimes complex to untangle. The study of relationships through graphical models has gained a lot of attraction due to their intuitive way to model structure dependencies.

A graph is an ordered pair $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where $\mathcal{V}$ is a set of vertices or nodes, $X_0, X_1, \ldots, X_J$, and $\mathcal{E}$ is a set of edges, $E_e : (X_j, X_{j'})$ with $j \neq j'$, that represent binary relations. A sub-graph $\mathcal{G}' = (\mathcal{V}', \mathcal{E}')$ of $\mathcal{G}$, is a graph such that $\mathcal{V}' \subseteq \mathcal{V}$ and $\mathcal{E}' \subseteq \mathcal{E}$, in which each edge in $\mathcal{E}'$ is incident on vertices in $\mathcal{V}'$.

Graphical models are categorised according to two features (Sucar, 2015):

1. Causality: a graph is said causal or directed if the orientation of the relations between its nodes is important as opposed to undirected (or non-causal) graphs in which relations are symmetric.

2. Staticity: a graph is said to be static if its set of nodes only captures the state of the system at a certain point in time as opposed to dynamic graphs.

This manuscript will be focused on non-causal, static graphical models.

Causal relationships, which will not be explored in this manuscript, are defined as follows: X_j is a cause of $X_{j'}$, with $j \neq j'$, if $X_j \rightarrow X_{j'}$. Conversely, $X_{j'}$ is a cause of X_j if $X_{j'} \rightarrow X_j$ or $X_j \leftrightarrow X_{j'}$ if the relation is bi-directional. These graphs are referred to as directed graphical models (DGM) or digraphs. Directed GM are also misleadingly referred to as Bayesian networks although statistical inference based on these graphs can be performed using either frequentist or Bayesian methods.

On the other hand, undirected graphs represent symmetric relations between features,

i.e. either no causal relationship exist between the objects or the causalities are not reported.

In a graph, the relationships among a set of objects are illustrated by networks of vertices and edges linking the nodes, as depicted in Figure 2.1.1. The vertices are represented by circles while lines or arrows materialize the dependencies of the objects to each other.

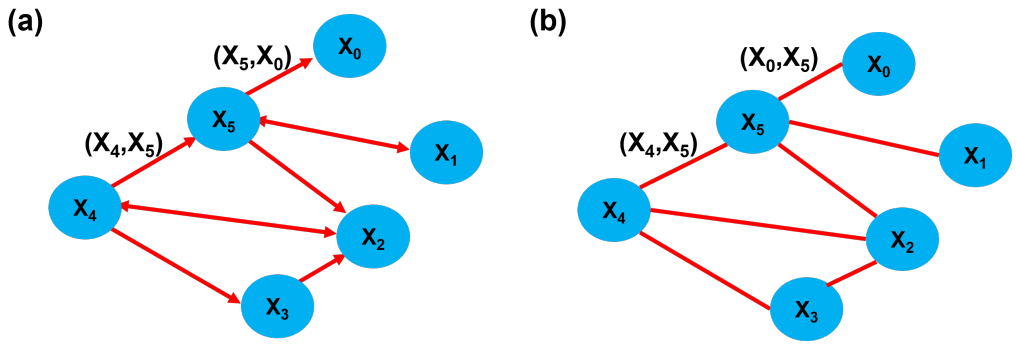

Figure 2.1.1: **Generic example of small directed (a) and undirected (b) graphical models**. Six objects depicted by circles (*i.e.* vertices X_0 to X_5) connected through 7 lines (b) or arrows (a), *i.e.* edges: $(X_j, X_{j'})$ with $j \neq j'$, materializing their causal (a) and undirected (b) relationships.

Two vertices X_j and $X_{j'}$, with $j \neq j'$, are said to be adjacent, denoted $X_j \sim X_{j'}$, if there is an edge between X_j and $X_{j'}$ in $\mathcal{G}$, *i.e.* if either $X_j - X_{j'}$ (undirected), $X_j \rightarrow X_{j'}$ and $X_j \leftarrow X_{j'}$ (unidirectional) or $X_j \leftrightarrow X_{j'}$ (bidirectional) (Højsgaard *et al.*, 2012). In Figure 2.1.1, node X_5 is adjacent to vertices X_0, X_1, X_2 and X_4.

In an undirected graph, the degree of a node is the number of edges that are incident to that node. In Figure 2.1.1 panel (b), node X_5 has a degree of 4 and vertex X_0 has a degree of 1.

The density of a graph, or edge density, is the proportion of existing edges with regards to the number of possible edges. The graph in Figure 2.1.1 panel (b) could accommodate a total of 15 edges ($J(J-1)/2$ with $J = 6$) but contains 7 edges; the graph density is then 0.466 ($= 7/15$).

A path, or trajectory, is a sequence of edges, $\{E_1, \ldots, E_E\}$, between distinct nodes $\{X_1, \cdots, X_J\}$ such that the final vertex of each edge coincides with the initial vertex of the next edge in the sequence $(X_{l-1}, X_l) \in \mathcal{E}$ for $l = 1, \ldots, J$. In Figure 2.1.1 (b), X_0, X_5, X_4, X_3, X_2 is a path of length 4, *i.e.* the number of edges linking the 5 distinct nodes. The distance between two nodes is the number of edges in the shortest path (also called a graph geodesic), *e.g.* in Figure 2.1.1 (b) the distance between node pair (X_0, X_3) is 3; consisting of edges

$(X_0, X_5), (X_5, X_4)$ and (X_4, X_3). For any graph, the diameter equals the number of edges in the longest shortest path between any two nodes (Sudderth, 2006), *i.e.* the longest graph geodesic is the graph diameter. In Figure 2.1.1 **(*b*)**, the graph is of diameter 3 as the longest path amongst the shortest ones is between pair (X_0, X_3) or pair (X_1, X_3).

An n-cycle, or n-circuit, is a path which starts and ends with the same node $X_0 = X_T$, and for which all internal nodes $(i_1, \ldots, i_{T-1})$ are distinct. The prefix "n" indicates the number of distinct nodes in the cycle, *e.g.* the graph illustrated in Figure 2.1.2 **(*c*)** contains a 4-cycle path (*i.e.* $\{X_0, X_1, X_2, X_3, X_0\}$) and two 3-cycle paths (*i.e.* $\{X_0, X_1, X_3, X_0\}$ and $\{X_1, X_2, X_3, X_1\}$.
An edge joining two non-consecutive vertices within a cycle is called a *chord*, *i.e.* a chord is not part of the cycle but joins 2 nodes that belong to the cycle. A cycle is said chordless if it has no chord edge. The graph in Figure 2.1.1 **(*b*)** has two chordless 3-cycle, (X_4, X_2, X_3, X_4) and (X_4, X_2, X_5, X_4), while its 4-cycle $(X_4, X_5, X_2, X_3, X_4)$ contains the chord edge (X_2, X_4).

As depicted in Figure 2.1.2 **(*a*)**, an edge which is incident on one vertex only is a *loop*. Edges associated to the same pair of nodes are said to be *parallel* edges (Figure 2.1.2 **(*b*)**).

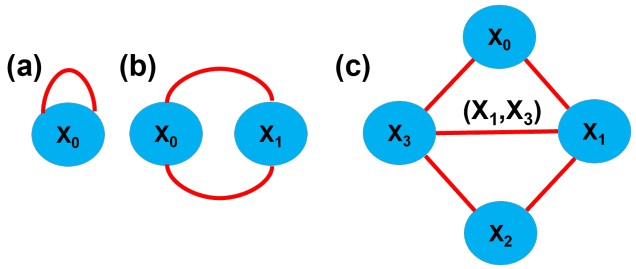

Figure 2.1.2: **Cycles and types of edge**. **(*a*)** Loop edge starting and ending on the same vertex. **(*b*)** Parallel edges starting and ending on the same pair of nodes. **(*c*)** Chord edge (X_1, X_3) joins two non-consecutive nodes, X_1 and X_3, in the 4-cycle path $\{X_0, X_1, X_2, X_3, X_0\}$.

In addition to the two basic categories of graphs, *i.e.* directed and undirected, there exist other types of graphs, illustrated in Figure 2.1.3, such as:

(a) Chain graphs: hybrid graphs that have directed and undirected edges.

(b) Simple graphs: graphs that do not include loops and parallel edges.

(c) Multigraphs: graphs with several components (sub-graphs), such that each component has no edges to the other components, *i.e.* they are disconnected. A vertex that is not an endpoint to any edge is an isolated vertex.

(d) Complete graphs: graph that have an edge between each pair of vertices as opposed to an incomplete graphs.

(e) Bipartite graphs: graphs in which the vertices can be divided in two subsets, $\mathcal{G}_1$ and $\mathcal{G}_2$ with $\{\mathcal{G}_1,\mathcal{G}_2\} \subseteq \mathcal{G}$, such that all edges connect a vertex in $\mathcal{G}_1$ with a vertex in $\mathcal{G}_2$; that is, there are no edges between nodes within each subset.

(f) Undirected tree graphs: there is a simple trajectory between each pair of vertices. These are composed of two types of nodes: leaf and internal vertices; with the former having degree one and the latter having degree greater than one. When the edges of a directed acyclic[1] graph are replaced by undirected ones, the undirected version of the graph is tree–structured.

(g) Weighted graphs: graphs that have weights associated to their edges and/or vertices.

(h) Triangulated graphs: have no chordless cycle of length greater than or equal to four.

A clique, $\mathcal{C}$, is a maximally complete subset graph of $\mathcal{G}$ such that there is no other complete set in $\mathcal{G}$ that contains $\mathcal{C}$. The graph in Figure 2.1.3 panel $(\boldsymbol{b})$ contains 2 cliques: $\{X_0, X_1, X_2\}$ and $\{X_0, X_2, X_3\}$.

The boundary of a subset $\mathcal{V}_1 \subseteq \mathcal{V}$ is defined as those vertices in $\mathcal{V} \setminus \mathcal{V}_1$ that are adjacent to a vertex in $\mathcal{V}_1$ (Edwards, 2000). Nodes X_1 and X_2 form the subset that is the boundary of subgraph $\mathcal{G}_1$ in Figure 2.1.3 panel $(\boldsymbol{e})$.

If $\mathcal{V}_1, \mathcal{V}_2$ and $\mathcal{V}_3 \subseteq \mathcal{V}$ are three disjoint subsets of nodes in an undirected graph $\mathcal{G}$, then $\mathcal{V}_3$ is said to separate $\mathcal{V}_1$ from $\mathcal{V}_2$ if every path between a node in $\mathcal{V}_1$ and a node in $\mathcal{V}_2$ contains at least one node in $\mathcal{V}_3$.

A graph is decomposable if it is complete or if and only if it is triangulated. More generally, a graph is decomposable if it can be decomposed into decomposable subgraphs (Højsgaard *et al.*, 2012). Graphs $(\boldsymbol{d})$ and $(\boldsymbol{h})$ of Figure 2.1.3 are decomposable since the former is complete and the latter triangulated. On the contrary graph $(\boldsymbol{a})$ of Figure 2.1.4 has a chordless cycle of length 4 hence is not decomposable.
Triangulated graphs are of great value in this research area as they allow considerable computational simplification[2].

[1] Cycles are defined later in the text.
[2] This statement is explicitly addressed in Section 2.2.

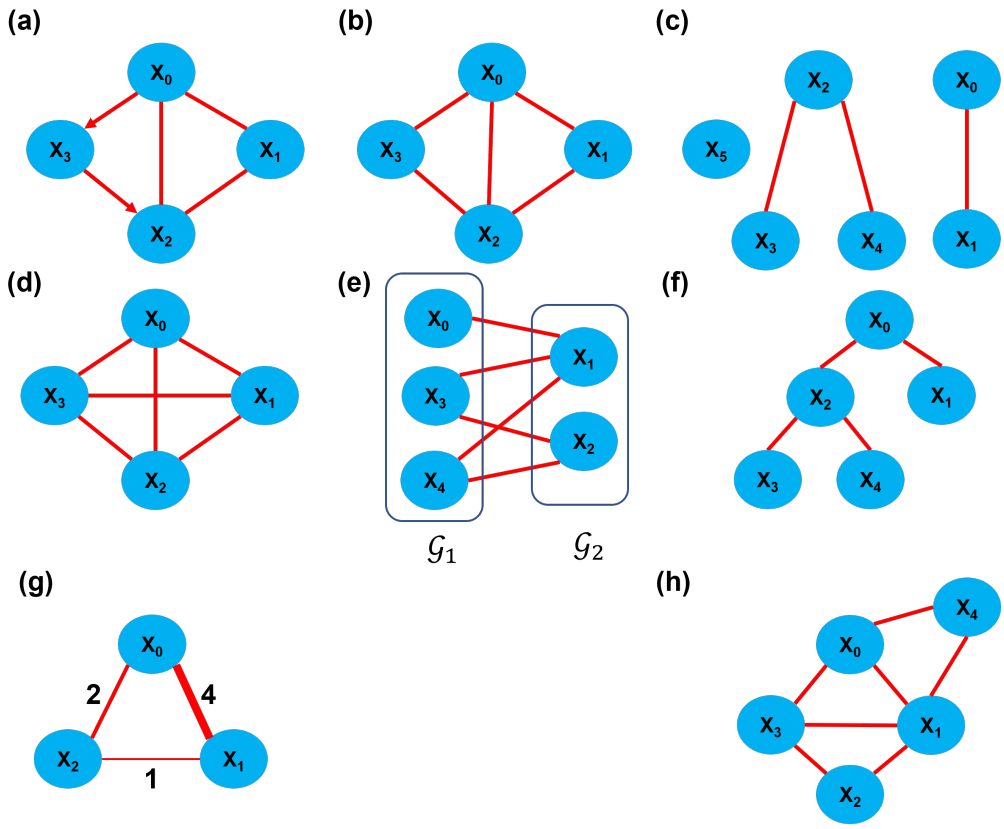

Figure 2.1.3: **Generic examples of different graph types.** (a) Chain graph: the relations (X_0, X_3) and (X_3, X_2) are causal while the remaining ones are undirected. (b) Simple graph: no vertices pair is incident to the same pair of edges and no edge loops on the same node. (c) Multigraph: three subgraphs composed of nodes $\{X_5\}$, $\{X_2, X_3, X_4\}$ and $\{X_0, X_1\}$. (d) Complete graph: each node is linked to all the other vertices of the graph. (e) Bipartite graph: $\mathcal{G}_1$ and $\mathcal{G}_2$ indicate 2 graph subsets of $\mathcal{G}$. (f) Undirected tree graph: X_1, X_3 and X_4 are the leaf nodes; X_0 and X_2 are the internal nodes. (g) Weighted graph: the values on the lines indicate the weights of the edges. (h) Triangulated graph: cycle X_0, X_1, X_2, X_3, X_0, of size 4, within the graph has a chord edge (X_1, X_3).

Graphs are said to be isomorphic if there is a one to one correspondence between their vertices and edges, so that the incidences are maintained, as illustrated in Figure 2.1.4.

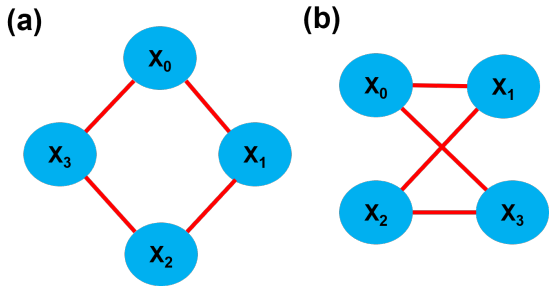

Figure 2.1.4: **Graph isomorphism**. Graph in panels (a) and (b) display the exact same nodes and edges paths.

2.2 Probabilistic Graphical Models (PGM)

Objects and their interconnectedness loosely fall into one or more of the following four network categories: technological (*e.g.* the Internet web, network of rail roads), social (*e.g.* bonding through the Facebook social media, interactions among a group of chimpanzees), informational (*e.g.* World Wide Web, citations between academic journals) and biological (*e.g.* metabolic pathway, neuronal network). Objects in these networks are constituted by any element (*e.g.* a person, a train station, a computer) around which a web of relations can be drawn (Newman, 2003).

Of the first three categories of networks, (*i.e.* technological, social and informational), part or the entirety of their topology is often available. Some of these are indeed human constructions for which either the blueprints exist (*e.g.* rail roads network) or their interconnectedness is readily uncovered through simple data collection (*e.g.* "who is friend with whom on Facebook"?).
The research topics around these categories of networks tend to focus on their nature and structure. Among other outstanding questions, one might study how information is spread across the network, where or when bottlenecks happen in the network, or whether subnetworks facilitating the spread of the Covid-19 mutants exist.

Biological network research topics include, among other things, the search for meaningful clusters (*e.g.* discovery of new genes circuit), the inference of roles in cellular function (*e.g.* gene function prediction) and impact of external stimuli (*e.g.* the response to new anti-Covid treatments).

Most often, the exact topology of biological networks is unknown. The 'mapping' of the underlying biological system involves the collection of data that are:

- of relational nature;

- high-dimensional;
- probabilistic.

The latter account for the uncertainty on the interconnectedness between objects, thus leading to randomness in the graphical networks, thusfore referred to as probabilistic graphical models (PGM). These combine the rigour of a probabilistic approach with the intuitive representation of relationships given by graphs. As such, PGM are natural and powerful tools to unravel the complex structure of data. These were initially developed in the 70's (Darroch *et al.*, 1972; Wermuth, 1976) as special subclasses of log-linear models which are interpretable in terms of conditional independencies[3]. PGM are commonly used in probability theory, Bayesian statistics and machine learning (Kolaczyk, 2009).

PGM are constituted by two main elements (Scutari *et al.*, 2011):

1. a vector $X = (X_1, ..., X_p)^T$ of p random variables describing the quantities of interest. The statistical distribution of X, $P(X)$, is called the global distribution of the data, while the components it factorises into are called local distributions, $P(X_j \mid X_{j'})$.
2. a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ as defined in Section 2.1.

The most common classes of PGM and their type are listed in Table 2.2.1.
Certain systems, *e.g* metabolic pathways, can be modelled as a series of nodes that can take different values and are influenced probabilistically by the states of their neighbours. A *configuration* is a particular assignment of values to each variable in the model. These models are known as Markov random fields (MRF) or Markov networks. A MRF can be categorized as regular, when the random variables form a grid, if not, they are irregular.

Graphically, MRF are undirected graphical models consisting of a set of random variables, $\mathcal{V}$, and a set of undirected edges $\mathcal{E}$. This type of graph is the main focus of this manuscript. MRF satisfies the Markov property, or locality property: a variable X_j is independent of all other variables in the field given its neighbours (Blake *et al.*, 2018).

A graph $\mathcal{G}$ is a dependency map, or D-map, of the probabilistic dependence structure P of X if there is a one-to-one correspondence between the random variables in X and the nodes $\mathcal{V}$ of $\mathcal{G}$, such that for all disjoint subsets A, B, C of X

$$A \perp\!\!\!\perp_P B \mid C \Rightarrow \mathcal{V}_A \perp\!\!\!\perp_G \mathcal{V}_B \mid \mathcal{V}_C$$

where subset C is said to separate subsets A and B in the graph $\mathcal{G}$. In Markov networks, graphical separation is easily defined due to the lack of direction in the relation.

[3]Conditional independence is defined in Section 2.3.

Type	**Directed (*D*) Undirected (U)**	**Dynamic (*D*) Static (S)**
Bayesian classifiers	D/U	S
Markov chains	D	D
Hidden Markov models	D	D
Markov random fields	U	S
Bayesian networks	D	S
Dynamic Bayesian networks	D	D

Table 2.2.1: **Most common probabilistic graphical models (PGM) types.** The graph models are classified according to 2 properties: causality and staticity.

Similarly, $\mathcal{G}$ is an independency map (or I-map) of the probabilistic dependence structure of X if $\mathcal{V}_A \perp\!\!\!\perp_G \mathcal{V}_B \mid \mathcal{V}_C$ also holds in P, *i.e.*

$$A \perp\!\!\!\perp_P B \mid C \Leftarrow \mathcal{V}_A \perp\!\!\!\perp_G \mathcal{V}_B \mid \mathcal{V}_C.$$

An I-map graph does not guarantee that every conditional independence relationship present in P is reflected in the graph; instead these graphs are referred to as minimal I-maps.
$\mathcal{G}$ is said to be isomorphic to, or the perfect map (P-map) of, the probabilistic dependence structure P of X if it is both a D-map and an I-map, that is

$$A \perp\!\!\!\perp_P B \mid C \Leftrightarrow \mathcal{V}_A \perp\!\!\!\perp_G \mathcal{V}_B \mid \mathcal{V}_C.$$

Considering the probabilistic dependence structure $X_0 \perp\!\!\!\perp_P X_2 \mid (X_1, X_3)$, graphs $(\boldsymbol{b})$ and $(\boldsymbol{c})$ in Figure 2.2.1 are D-maps as the conditional independence between node X_0 and X_2 is captured, unlike graph $(\boldsymbol{a})$. Similarly, graphs $(\boldsymbol{b})$ and $(\boldsymbol{c})$ are both, al least, minimal I-maps. Now considering the extra conditional independence $X_2 \perp\!\!\!\perp_P X_3 \mid (X_1, X_0)$ pair in the probabilistic dependence structure, only graph $(\boldsymbol{c})$ is a P-map. Both graphs $(\boldsymbol{b})$ and $(\boldsymbol{c})$ are still (minimal) I-maps but the former is no longer a D-map.

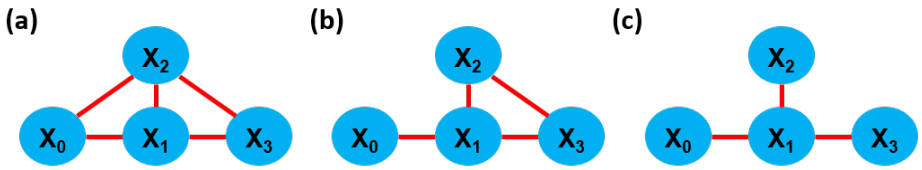

Figure 2.2.1: **Dependency map**. The graphs in panel $(\boldsymbol{a})$, $(\boldsymbol{b})$ and $(\boldsymbol{c})$ display the exact same nodes but different edges. Classification into a D-map, an I-map or an isomorphic map will depend on the underlying probabilistic dependence structure.

When the dependency structure P of X can be expressed by multiple graphs, one must use the one with the minimum number of edges; if any further edge is removed then the

graph is no longer an I-map of P.

The Markov property defines the decomposition of the global distribution, or joint distribution, of the data into a set of local distributions (Barber, 2012; Scutari *et al.*, 2011). The chain rule of probability takes the form

$$f(X) = \frac{1}{Z} \prod_{l=1}^{L} \psi_l(\mathcal{C}_l)$$

were Z is a normalizing constant ensuring that the density integrates to 1 and $\mathcal{C}_l$ indicates the cliques. The non-negative functions ψ_l, or Gibbs' potentials, are the relative mass of probability of each clique or their density functions if the graph is decomposable.
In that case, the global distribution factorises again according to the chain rule as follows

$$f(X) = \frac{\prod_{l=1}^{L} f(C_l)}{\prod_{l=1}^{L} f(S_l)}$$

where S_l are the nodes of C_l which are also part of any other clique and $f(S_l)$ is their joint density (Pearl, 1988). In Figure 2.2.2, the global distribution of $X \in \mathbb{R}^3$ factorises into:

$$P(X) = \frac{P(X_0, X_1)P(X_0, X_2)}{P(X_0)} = P(X_1 \mid X_0)P(X_2 \mid X_0)P(X_0),$$

provided that $P(X_0) > 0$.

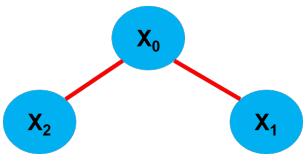

Figure 2.2.2: **Markov network composed of two cliques.** $\mathcal{C}_1 = \{X_0, X_1\}$ and $\mathcal{C}_2 = \{X_0, X_2\}$, separated by X_0. To match the probabilistic and graphical model notation, nodes were associated with components of the random vector X.

The Markov blanket of a node X_j, is defined as the set of vertices that completely separates X_j from the rest of the graph, *i.e.* it is the set of nodes that includes all the knowledge needed to perform inference on X_j. In Markov networks, the Markov blanket coincides with the neighbours of X_j as illustrated in Figure 2.2.3. If the number of edges is small compared to the number of nodes, *i.e.* sparse network, the interpretation of each Markov blanket becomes a useful tool in understanding the data.

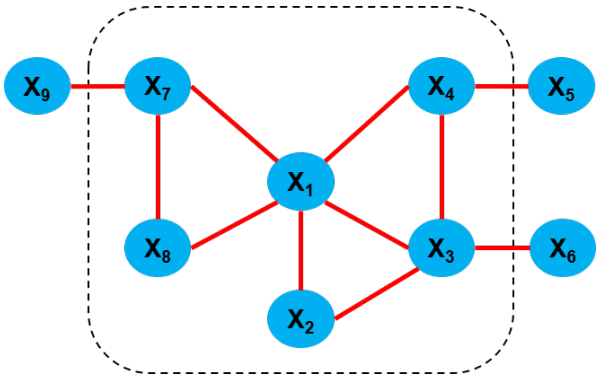

Figure 2.2.3: **Markov blanket.** The Markov blanket of node X_1 within the Markov network is delimited by the dashed rectangle and includes vertices $\{X_1, X_2, X_3, X_4, X_7, X_8\}$.

2.3 Gaussian Graphical Models (GGM)

Quantities that are the sum of independent inputs approximately follow normal distribution laws, as established by the central limit theorem. For instance, the expression level of genes might be the outcome of the combined actions of various transcription factors and epigenetic modifications (Markowetz *et al.*, 2007; Pavlopoulos *et al.*, 2011).

In multivariate normal settings, the joint multivariate Gaussian distribution of the random vector $X = (X_1, \ldots, X_p)^T \in \mathbb{R}^p$, with $X \sim \mathcal{N}_p(\mu, \mathbf{\Sigma})$, where $\mu \in \mathbb{R}^p$ and $\mathbf{\Sigma} \succ 0$, *i.e.* a positive definite, $(p \times p)$ matrix

$$\mathbf{\Sigma} = \begin{pmatrix} \sigma_{11}^2 & \sigma_{12} & \cdots & \sigma_{1p} \\ \vdots & \vdots & \ddots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{p1} & \sigma_{p2} & \cdots & \sigma_{pp}^2 \end{pmatrix},$$

is given by

$$f_{\mu,\mathbf{\Sigma}}(x) = (2\pi)^{-\frac{p}{2}} (\det \mathbf{\Sigma})^{-\frac{1}{2}} \exp\left\{ -\frac{1}{2}(x-\mu)^T \mathbf{\Sigma}^{-1}(x-\mu) \right\}, \quad x \in \mathbb{R}^p,$$

where $\det(\cdot)$ is the determinant operator and $\mathbf{\Sigma}^{-1}$ denotes the inverse of the covariance matrix.
An illustration of a bivariate normal density is provided in Figure 2.3.1.

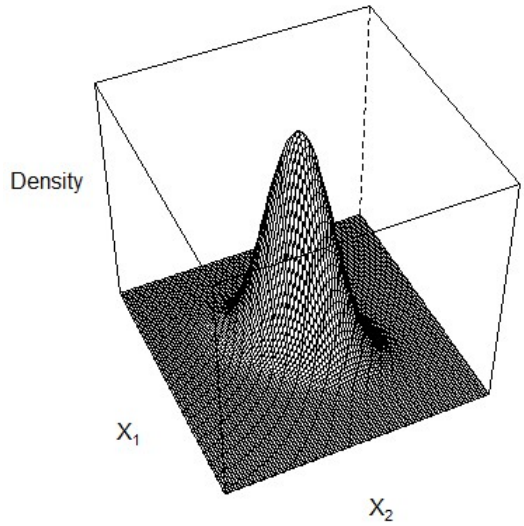

Figure 2.3.1: **Bivariate Gaussian probability function**. Joint normal density of the random normal vector $X = (X_1, X_2)^T$ with $\mu = (0, 0)^T$ and $\mathbf{\Sigma} = \begin{pmatrix} 1 & .5 \\ .5 & 1 \end{pmatrix}$.

Let X_A and X_B be two subsets, of dimensions q and r with $q+r = p$, of the random vector $X \sim \mathcal{N}_p(\mu, \mathbf{\Sigma})$. The covariance matrix $\mathbf{\Sigma}$ is re-formulated in block matrices

$$\mathbf{\Sigma} = \begin{pmatrix} \mathbf{\Sigma}_{AA} & \mathbf{\Sigma}_{AB} \\ \mathbf{\Sigma}_{BA} & \mathbf{\Sigma}_{BB} \end{pmatrix}$$

where $\mathbf{\Sigma}_{AA}$ is the covariance of the subset X_A, *i.e.* $\text{Cov}(X_A)$, and $\mathbf{\Sigma}_{AB}$ is the covariance of subsets X_A and X_B, *i.e.* $\text{Cov}(X_A, X_B)$.

Taking the Schur complement[4] of the block $\mathbf{\Sigma}_{AA}$, one obtains the conditional covariance of the subset X_B given $X_A = \{X_1^A, ..., X_q^A\}$

$$\mathbf{\Sigma}_{B|A} = \mathbf{\Sigma}_{BB} - \mathbf{\Sigma}_{BA}\mathbf{\Sigma}_{AA}^{-1}\mathbf{\Sigma}_{AB}.$$

Considering the Cholesky factorization of the covariance matrix $\mathbf{\Sigma}$ (Whittaker, 1990),

$$\mathbf{\Sigma} = \begin{pmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{\Sigma}_{BA}\mathbf{\Sigma}_{AA}^{-1} & \mathbf{I} \end{pmatrix} \begin{pmatrix} \mathbf{\Sigma}_{AA} & \mathbf{0} \\ \mathbf{0} & \mathbf{\Sigma}_{B|A} \end{pmatrix} \begin{pmatrix} \mathbf{I} & \mathbf{\Sigma}_{AA}^{-1}\mathbf{\Sigma}_{AB} \\ \mathbf{0} & \mathbf{I} \end{pmatrix},$$

one can invert[5] $\mathbf{\Sigma}$ to obtain

$$\mathbf{\Sigma}^{-1} = \begin{pmatrix} \mathbf{\Sigma}_{AA}^{-1} + \mathbf{B}^T\mathbf{\Sigma}_{B|A}^{-1}\mathbf{B} & -\mathbf{B}^T\mathbf{\Sigma}_{B|A}^{-1} \\ -\mathbf{\Sigma}_{B|A}^{-1}\mathbf{B} & \mathbf{\Sigma}_{B|A}^{-1} \end{pmatrix}$$

[4]Development provided in Appendix Section 7.2
[5]Proof provided in the Appendix sections 7.3 and 7.4

where $\mathbf{B} = \mathbf{\Sigma}_{BA}\mathbf{\Sigma}_{AA}^{-1}$. Permutation, in the original random vector, and selection of variables allows this results to be true for any variables combination.

Let us now introduce the new notation

$$\mathbf{\Theta} = \begin{pmatrix} \theta_{1,1} & \theta_{1,2} & \cdots & \theta_{1,p} \\ \vdots & \vdots & \ddots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ \theta_{p,1} & \theta_{p,2} & \cdots & \theta_{p,p} \end{pmatrix} \equiv \mathbf{\Sigma}^{-1} = \begin{pmatrix} \sigma_{1,1}^2 & \sigma_{1,2} & \cdots & \sigma_{1,p} \\ \vdots & \vdots & \ddots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{p,1} & \sigma_{p,2} & \cdots & \sigma_{p,p}^2 \end{pmatrix}^{-1}$$

where the indexes in the precision matrix $\mathbf{\Theta}$ match the ones in matrix $\mathbf{\Sigma}$. Each off-diagonal element $(\mathbf{\Theta})_{a,b}$ of the matrix quantifies the (in)dependence of the variables pair $(X_j, X_{j'})$, with $j \neq j'$, conditionally on all other variables (Bühlmann *et al.*, 2011; Lauritzen, 1996).

This property of the multivariate Gaussian distribution has an important implication with respect to the interpretation of zeros in the precision matrix as zeros correspond to conditional independence relations (Dempster, 1972).

Probabilistic graphical models encoding the conditional independence structure for the Gaussian distribution are called "Gaussian Graphical Models" (GGM) or covariance selection models.

A random vector $X \in \mathbb{R}^p$ is said to satisfy the Gaussian graphical model with graph $\mathcal{G}$, if it has a multivariate Gaussian distribution $\mathcal{N}(\mu, \mathbf{\Sigma})$ with

$$(\mathbf{\Sigma}^{-1})_{a,b} \equiv (\mathbf{\Theta})_{a,b} = 0 \quad \text{for all } (a,b) \notin \mathcal{E}, \text{ with } a \neq b.$$

The graph $\mathcal{G}$ describes the sparsity pattern of the precision matrix; also referred to as the concentration graph.
In the graph model depicted in Figure 2.3.2, node X_3 is independent of X_1 conditionally on X_2.

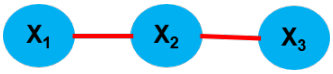

Figure 2.3.2: **Conditional independence**. Three nodes, *i.e.* X_1, X_2 and X_3, connected by 2 edges materializing their undirected relationships. Node X_3 is independent of X_1 conditionally on X_2.

In terms of model representation, this graph would be encoded as follows

$$\mathbf{\Theta} = \begin{pmatrix} \theta_{1,1} & \theta_{1,2} & 0 \\ \theta_{2,1} & \theta_{2,2} & \theta_{2,3} \\ 0 & \theta_{3,2} & \theta_{3,3} \end{pmatrix}$$

where $\theta_{a,b}$ are the parameters[6] of the graphical model which need to be estimated when they are considered not equal to zero.

[6]Sometimes referred to as coefficients in this manuscript.

Chapter 3

Learning Gaussian Graphical Models

The fitting of graphical models, also called *learning*, is a two-step process (Scutari *et al.*, 2011).

The first step, called structure learning, aims at finding the graph structure that best reflects the conditional independencies structure of the random vector X. At this stage of the process, the graph structure $\mathcal{G}$ should be a minimal I-map of the probabilistic dependence structure of X, that is:

$$A \perp\!\!\!\perp_P B \mid C \Leftarrow \mathcal{V}_A \perp\!\!\!\perp_G \mathcal{V}_B \mid \mathcal{V}_C$$

where A, B and C are subsets of X. Of note, any conditional independence relationship present in an I-map graph must hold in $P(X)$, the probabilistic dependence structure of X.
The structure learning takes a number of assumptions:

- there must be a one-to-one correspondence between the nodes of the graph and the random variables included in the model;
- there must be no unobserved, also called latent or hidden, variables;
- observations must be stochastically independent;
- every combination of the variable values must represent a valid, observable event;
- if the variables in X are continuous, the global distribution and the local distributions are usually assumed to follow Gaussian distributions;
- by definition, all the relations in the network must be conditional independences.

The second step, called parameter learning, aims at estimating the parameters of the local distributions associated with the cliques of the graph. Once the graph structure has been

defined, estimating the global distribution parameters is greatly simplified by the application of the Markov property.

Statistical learning relies heavily on an appropriate sampling from the population of interest. As stated in the structure learning assumptions, the entities of the sample must be stochastically independent but they also must be representative of the population. From this point onward, we will consider the sample matrix $\mathbf{X} \in \mathbb{R}^{n \times p}$ were n is the sample size and p is the number of variables (*i.e.* features).

In multivariate normal settings, the graph parameters are classically estimated via the Gaussian log-likelihood[1]:

$$\ell_n(\boldsymbol{\Theta}) \approx \frac{n}{2} \left[\log(\det \boldsymbol{\Theta}) - \operatorname{tr}(\mathbf{S}\boldsymbol{\Theta})\right]$$

where $\mathbf{S}$ denotes the empirical $p \times p$ covariance matrix of the centred random vectors of observations X_i,

$$\mathbf{S} = \frac{1}{n} \sum_{i=1}^{n} X_i X_i^T, \tag{3.0.1}$$

$\boldsymbol{\Theta} \equiv \boldsymbol{\Sigma}^{-1}$, and $\operatorname{tr}(\cdot)$ is the trace operator.

In low dimensional settings, where the number of observations n is larger or equal to the dimension p of $\mathbf{X}$, one can identify the structure of the graph $\mathcal{G}$ by multiple hypotheses testing of its $p(p-1)/2$ partial correlation elements (Drton *et al.*, 2007; Edwards, 2000). A common problem in graphical models is the high dimensionality of the data compared to the small sample size, *i.e.* $p \gg n$. Typically, in such graphical models, the number of parameters outnumbers the data points and the maximum likelihood estimators are no longer fit for purpose.

3.1 Gaussian Graphical Lasso

As mentioned in the previous section, the estimation of high-dimensional graphs, *i.e.* $p \gg n$, is a challenging task. As such, regularized (or penalized) models have been at the forefront of the estimation process. Put simply, the goal behind these models is to shrink estimates and, for some, drive (as many of) them to zero (Bühlmann *et al.*, 2011).

In high-dimensional settings, three main issues might occur when the maximum likelihood estimator $\hat{\boldsymbol{\Sigma}}_{MLE}$ is used to obtain $\hat{\boldsymbol{\Sigma}}^{-1}$ by inversion:

- given the large number, *i.e.* $p(p+1)/2$, of unknown parameters to estimate, $\hat{\boldsymbol{\Sigma}}$ is **not a stable estimator of** $\boldsymbol{\Sigma}$;

[1] The reformulation of the Gaussian log-likelihood is provided in Appendix Section 7.1.

- $\hat{\mathbf{\Sigma}}$ is **singular** and thus cannot be inverted to yield an estimate of $\mathbf{\Sigma}^{-1}$;
- **unconnected pairs** in the graph model, will in general **not be equal to zero** in the estimated precision matrix.

3.2 Regularization methods

Due to the issues mentioned above and to avoid the over-fitting of the data, a simplification of the models through regularization is considered. Models regularization is achieved through penalisation of some objective function. The general principle of penalized optimization takes the form

$$\min_{\beta}\{\text{loss}(\beta) + \lambda h(\beta)\} \tag{3.2.1}$$

where β is the vector of coefficients to be estimated and $\lambda > 0$ is a regularization parameter chosen *a priori*. The larger the value of λ, the greater the extend of shrinkage. In expression 3.2.1, the loss (also called cost or objective) function, *i.e.* $loss(\cdot)$, drives the optimal fitting of the model to the data while the penalization function, *i.e.* $h(\cdot)$, limits over-fitting. Among the many possible penalizing functions, the ℓ_s "regularizer" class, *i.e.* $\left(\sum_{j=1}^{p} |\beta_j|^s\right)^{1/s}$, is the most used and studied. Within that class, the two most common choices are the ℓ_1 and ℓ_2 regularization norms.

The ℓ_2 regularization norm, also called the Ridge penalty (Hoerl *et al.*, 1970), takes the form:

$$\hat{\beta}^{ridge} = \arg\min_{\beta}\{\text{loss}(\beta) + \lambda||\beta||_2^2\}$$

where $||\beta||_2$ is the Euclidean norm: $\sqrt{\sum_{j=1}^{p} \beta_j^2}$. In the context of Ridge linear regression, the coefficients are shrunk thus alleviating the issues of collinearity. However, the Ridge estimates get very close but never reach the value zero.

The least absolute shrinkage and selection operator (*lasso*) estimation method, based on the ℓ_1 regularization norm, has been first introduced in the context of linear regression (Tibshirani, 1996). The aim of a lasso optimization is to produce shrinkage and most importantly sparsity through coefficients zeroing. The *lasso* optimization takes the form

$$\hat{\beta}^{lasso} = \arg\min_{\beta}\{\text{loss}(\beta) + \lambda||\beta||_1\}$$

where $||\beta||_1$ is the Taxicab norm in reference to the Manhattan distance, defined for a vector as

$$||\beta||_1 = \sum_{j=1}^{p} |\beta_j|$$

Under an orthogonal design, *i.e.* $X^T X = I$, *lasso* regularization translates estimates by a constant factor, λ, until a threshold is reached where estimates are cut down to the value zero, as illustrated in Figure 3.2.1. This progressive shrinking is called "soft-thresholding" (Donoho *et al.*, 1994) as opposed to the "hard-thresholding" involving brutal estimates zeroing upon threshold attainment.

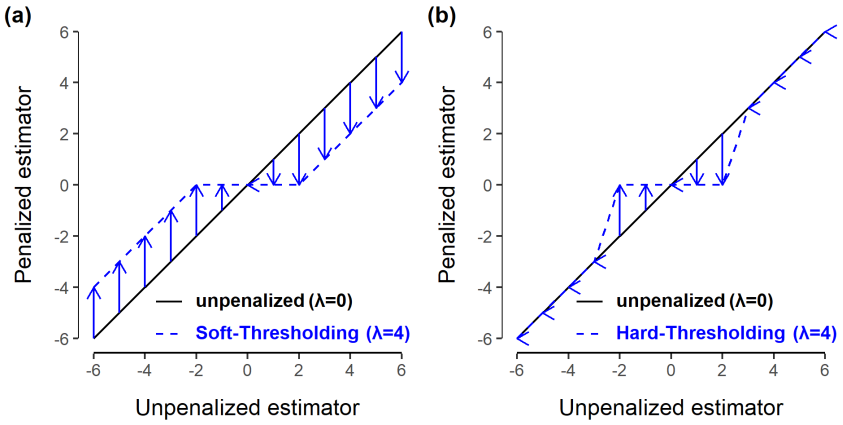

Figure 3.2.1: ***lasso* Soft-thresholding (*left*) and Hard-thresholding (*right*)**[2]. The plain black and dashed blue lines indicate the value of the unpenalized ($\lambda = 0$) or penalized ($\lambda > 0$) coefficients, respectively. (**a**) The coefficients are gradually shrunk and zero-levelled upon reaching the threshold. (**b**) The penalty only takes effect upon reaching the threshold where the coefficients are zeroed.

Importantly, ℓ_1 penalized minimization problems have a unique *lasso* solution with probability one, regardless of the sizes of n and p over a wide range of loss functions (Tibshirani, 2012). Figure 3.2.2 provides a geometrical visualization of the ℓ_1 and ℓ_2 regularization norms optimization principles. Starting from the "true" or optimal estimator value (black dot), one can move across the space of solutions, along the iso-surfaces (red ellipses where the loss function takes the same value), to shrink the estimates of the un-penalized original optimization problem. This is equivalent to optimizing under the constraint imposed upon the ℓ_1- and ℓ_2-norms (blue shapes). The size of the constraint is relative to the size of λ. The solution to the penalized optimization problem must therefore sit at the crossing-point between the iso-surface and the blue shape (constraint). A large λ ultimately results in the simultaneous shrinking of all estimates to values close to but never equal to zero in the ℓ_2 penalized optimization problem. The ℓ_1-norm constraint, however, leads to estimates zeroing due to its geometry, allowing for optimal solutions on the corner of the blue diamond shape.

[2]Figure adapted from Pr.Pircalabelu (LSTAT2400: *Introduction to high-dimensional statistics*, UCLouvain, 2018-2019).

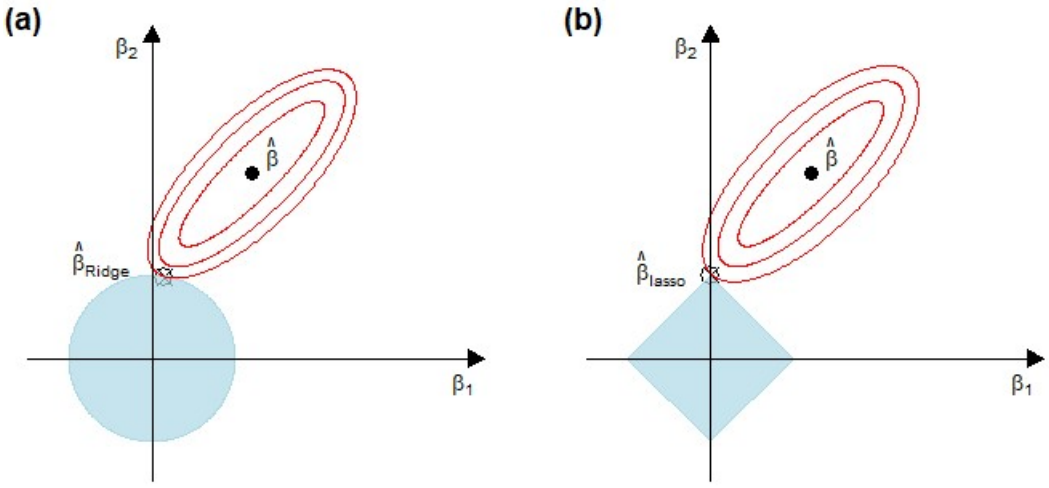

Figure 3.2.2: **Geometry of the ℓ_2 *(a)* and ℓ_1 *(b)* norms**. $\hat{\beta}$ is the vector of optimal coefficients. The red ellipses materialize the iso-surface of the loss function. The blue surface indicates the constraint imposed by the penalty on the coefficients. The black circle locate the optimal value of the coefficients. The crossed circle locates the crossing-point between the iso-surface and the constraint. *Adapted from Tibshirani, 1996.*

A convex loss functions combined to an ℓ_1 constraint results in an optimization problem that is convex which makes the optimization problem more likely to be solved. As illustrated in Figure 3.2.3, a function is convex (concave) if the line segment connecting two points, $(x_1, f(x_1))$ and $(x_2, f(x_2))$, sits on or above (below) the graph of the function f. Also, a function f is concave if its conjugate, *i.e.* $-f$, is convex. Conversely, a non-convex function is neither convex nor concave (*e.g.* sine and cosine functions). Mathematically, a function is convex if the following inequality holds:

$$f(\alpha x_1 + (1-\alpha)x_2) \leq \alpha f(x_1) + (1-\alpha)f(x_2)$$

with $\alpha \in (0, 1)$. When that relation is strict, *i.e.* the sign is "$<$", then the function f is strictly convex.

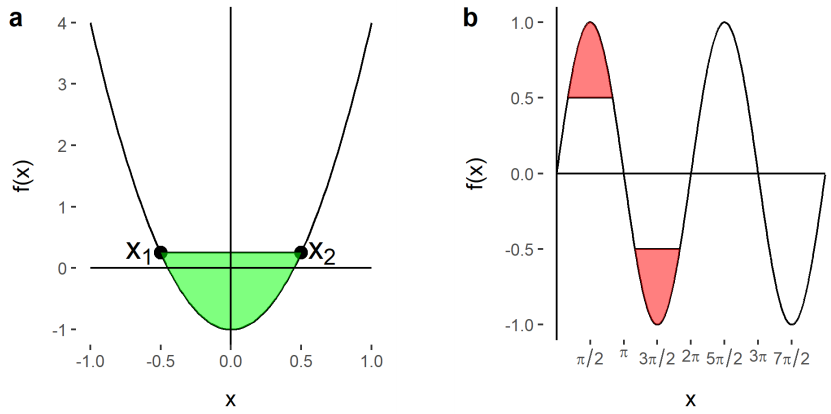

Figure 3.2.3: **Strictly convex (*a*) and non-convex (*b*) functions**. (**a**) As per definition the line segment between any two points on the graph of the function lie above the graph. (**b**) The areas centred around $x = \pi/2$ and $3\pi/2$ lie below and above the graph, respectively

3.3 Regularization methods applied to GGM

A straightforward application of the *lasso*-penalization to estimate the inverse covariance matrix $\boldsymbol{\Theta}$ was proposed by Meinshausen and Bühlmann (2006). The neighbours (variables) selection was performed through an ℓ_1-penalized linear regression model on a response variable X_j, associated to the node $\mathcal{V}_j \in \mathcal{V}$, and the X_A variables, *i.e.* the neighbouring nodes $\mathcal{V}_A \in \mathcal{V} \setminus \mathcal{V}_j$ forming its Markov blanket, as predictors.
Their approach involves the optimization of a convex function for each node in the graph. Let us consider $\beta \in \mathbb{R}^p$ the vector of the predictors coefficients. The solution to the penalized ordinary least squares estimators is

$$\hat{\beta}_j = \arg\min_{\beta}(n^{-1}||X_j - \boldsymbol{X}_{\setminus j}\beta_{\setminus j}||_2^2 + \lambda||\beta_{\setminus j}||_1)$$

where $\boldsymbol{X}_{\setminus j}$ is the data matrix, of dimension $n \times (p-1)$, excluding the response variable j, $X_j \in \mathbb{R}^n$ and $\beta_{\setminus j}$ is the vector of coefficients excluding β_j.
The neighbourhood estimate of a node $\mathcal{V}_j$ is then defined by the non-zero coefficient estimates of the ℓ_1-penalized regression. This relatively simple approach was viewed as an approximation to the exact problem, for one because it does not consider the positive definiteness and symmetry constraints on $\boldsymbol{\Theta}$. Although this optimization problem is convex, it is non-smooth, *i.e.* the function is non-differentiable and/or discontinuous in parts of the domain, and has an unbounded constraint set, *i.e.* it has no limit in some directions of the feasible set, therefore may not have an optimum. Despite the solution to this optimization problem not be unique, the neighbourhood still holds asymptotically.

In multivariate settings, the more favoured and studied approach is the log-likelihood

optimization. Its generic (penalized) Lagrangian formulation is as follows:

$$\underset{\boldsymbol{\Theta} \succ 0}{\text{maximize}} \{\log(\det \boldsymbol{\Theta}) - \text{tr}(\mathbf{S}\boldsymbol{\Theta}) - \lambda\, h(\boldsymbol{\Theta})\}$$

where the term $\{\log(\det \boldsymbol{\Theta}) - \text{tr}(\mathbf{S}\boldsymbol{\Theta})\}$ is the loss function, $h(\boldsymbol{\Theta})$ is a generic penalizing function and the $\mathbf{S}$ matrix denotes the empirical second moment of the distribution as defined in expression 3.0.1.

Somewhat related to the *lasso*-penalized log-likelihood, the optimization method proposed by Yuan *et al* (2007) considers an off-diagonal "garrote" penalty of the form:

$$h(\boldsymbol{\Theta}) = \sum_{a \neq b} \frac{\theta_{a,b}}{\tilde{\theta}_{a,b}}$$

s.t. $\frac{\theta_{a,b}}{\tilde{\theta}_{a,b}} \geq 0$, where $\tilde{\theta}_{a,b}$ is an acceptable estimate of $\theta_{a,b}$, with (a, b) the row and column, respectively, index of the parameters in the precision matrix. An illustration of the *garrote* penalty effect is provided in Figure 3.3.1(b). Comparatively to a *lasso* penalization, the *garrote* penalty shrinks the parameters more heavily for the same tuning parameters values. Asymptotically, this method selects the right graph with probability tending to one. However, the drawback of this approach is that an acceptable estimator of $\boldsymbol{\Theta}$ must be available.

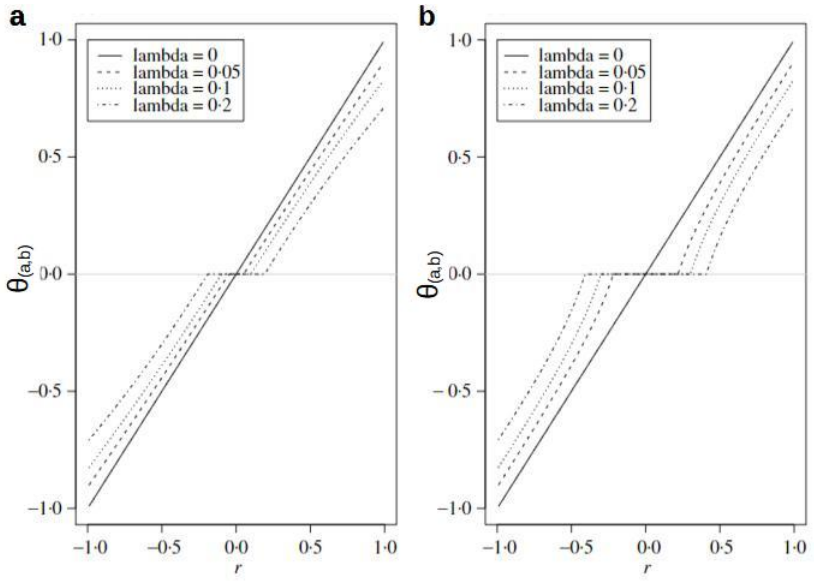

Figure 3.3.1: ***Lasso* (*a*) and *garrote* type (*b*) parameter soft-thresholding**. The coefficient of correlation r between any two variables and the tuning parameter are provided. *Figure from Yuan et al. (2007).*

The ℓ_1-norm penalized conjugate function proposed by Banerjee *et al.* (2008) takes the

form

$$h(\boldsymbol{\Theta}) = \max_{||\mathbf{U}||_\infty \leq 1} \operatorname{tr}(\boldsymbol{\Theta}\mathbf{U})$$

where $\mathbf{U}$ is a symmetric matrix and $||\cdot||_\infty$ is the vector ℓ_∞-norm, *i.e.* $\max_{ab}|U_{a,b}|$. The sparse maximum likelihood optimization problem is then reformulated as follows:

$$\hat{\boldsymbol{\Sigma}} = \arg\max\{\log\det \mathbf{W}\text{: } ||\mathbf{W} - \mathbf{S}||_\infty \leq \lambda\}$$

where $\mathbf{W} = (\mathbf{S} + \mathbf{U}) \succ \mathbf{0}$ is the estimate of the covariance matrix.
Using strong convex duality (Osborne *et al.*, 2000), the optimization problem is equivalent to

$$\underset{\boldsymbol{\Theta}}{\text{minimize}}\, ||\mathrm{Q}\theta_j - b||_2^2 + \lambda||\theta_j||_1$$

where matrix $\mathrm{Q} = \mathrm{W}_{\setminus j \setminus j}^{1/2}$ with its subscript indicating that the column and the row of variable j are excluded, $b = \frac{1}{2}\mathrm{Q}^{-1}S_j$ with $S_j \in \mathbb{R}^{(p-1)}$ being the j^{th} column vector of the empirical covariance matrix minus the diagonal element S_{jj}.
This problem is similar to a penalized regression of variable j against all the others, reminiscent of the Meinshausen and Bühlmann (2006) approach. An important result is that for any $j \in 1, ..., p$, if $\lambda \geq |S_{jj'}|$, *i.e.* the off-diagonal elements of the empirical covariance matrix estimate, for all $j \neq j'$, then one can show that column and row j of the estimate $\hat{\boldsymbol{\Sigma}}$ are zero, except for the diagonal element.
The solution to this optimization problem is obtained by a block coordinate descent[3] procedure in a way that guarantees that each penalized regression problem has a unique solution.
Of important note, the here-above method aims at estimating the covariance matrix $\hat{\boldsymbol{\Sigma}}$, rather than the precision matrix $\hat{\boldsymbol{\Theta}}$.

The ℓ_1-penalized log-likelihood optimization problem proposed by Friedman *et al.* (2008), which is very similar to Banerjee *et al.* (2008), is of the form

$$\underset{\boldsymbol{\Theta}}{\text{maximize}}\, \{\log(\det \boldsymbol{\Theta}) - \operatorname{tr}(\mathbf{S}\boldsymbol{\Theta}) - \lambda||\boldsymbol{\Theta}||_1\}\,.$$

The solution to this optimization problem, referred to as the ***graphical lasso***, is solved using a block coordinate descent procedure.
If needed, their procedure allows regularization for each variable, or even element, to be penalized differently using a component-wise penalizing function

$$h(\boldsymbol{\Theta}) = ||\boldsymbol{\Theta}\lambda_{a,b}||_1.$$

Of note, the penalized Gaussian likelihood estimators are still valid for some mildly non-Gaussian data since maximizing a penalized likelihood can be interpreted as minimizing a penalized log-determinant Bregman divergence (Ravikumar *et al.*, 2011).

[3] See section 3.4.

3.4 Algorithmic optimization methods

In the framework of convex problems optimization, two algorithms for solving large-scale statistical tasks are popular for their simplicity, stability and relative computational speed (Benfenati *et al.*, 2018):

- the Coordinate Descent (CD) procedure
  (Friedman *et al.*, 2007; Tong *et al.*, 2008; Wright, 2015);
- the Alternating Direction of Multipliers Method (ADMM)
  (Boyd *et al.*, 2010).

Unlike the *gradient descent* method, the *coordinate descent* algorithm moves along the function parameters axes, *i.e.* it explores a minimization path along one of the function parameters while keeping all the others fixed. The global problem is subdivided into one-dimensional sub-problems, which therefore can be solved with relative ease.
The single-coordinate descent method can be extended into the block coordinate descent, which adjusts groups of parameters, therefore reducing the dimension of the minimization problem, *i.e.* searching along a coordinate hyperplane rather than the full-size p-dimensional hyperspace.
The pseudo-code of a coordinate descent algorithm is provided in **Algorithm 1**. At the initial step, *i.e* $m = 0$, and for a specified tuning parameter λ, a starting point of the precision matrix is set to zero or any estimate. Within each iteration, *i.e* $m = 1, 2, \ldots,$ the precision matrix from the previous step is considered and one column of parameters is selected for optimization while maintaining all the others fixed. The precision matrix is then updated with the optimized column parameters. At each iteration, the algorithm cycles trough all the columns. At the end of each iteration, the estimated precision matrix is tested against some convergence and stopping criteria. If neither are reached, a new iteration is started and the latest update of the precision matrix is carried on.
The coordinate descent can be applied to problems in which parameters are involved in only a limited number of constraints such as two-dimensional fused *lasso*, an ℓ_1-norm penalizing both the parameters and their successive differences (Tibshirani *et al.*, 2005). In non separable problems, *e.g.* the general fused *lasso* problem, the CD might remain away from the optimal solution.
In large *lasso* optimization problems, the coordinate descent approach is very competitive even against the fastest procedures such as the Least Angle Regression and Shrinkage (LARS). Throughout the iterations, the minimization of many parameters does not require changes, hence, speeding up the procedure. For an orthonormal matrix $\mathbf{M}$ and a penalizing function of the form $h(\mathbf{M}) = \sum_{a \neq b} |M_{a,b}|$, no explicit search is needed as a closed-form solution of the sub-problems exists, hence, speeding up the procedure.

Algorithm 1: (*Lasso*) Coordinate descent algorithm

1: Select a scalar $\lambda > 0$
2: Initialize $\hat{\boldsymbol{\Theta}}^{(m=0)}$ to $\mathbf{0}$ or $\mathbf{I}$
3: **for** $m = 1, 2, \ldots$ **do**
4: **for** $j = 1, 2, \ldots, p$ **do**
5: Solve $\hat{\Theta}_j^{(m)} = \arg\min_{\boldsymbol{\Theta}} \{loss(\hat{\boldsymbol{\Theta}}^{(m-1)}) + \lambda h(\hat{\boldsymbol{\Theta}}^{(m-1)})\}$
6: Update $\hat{\boldsymbol{\Theta}}^{(m)}$ using $\hat{\Theta}_j^{(m)}$;
7: **end for**
8: Check convergence conditions
9: **end for**

The ADMM is a variant of the augmented Lagrangian constrained optimization problem using un-constrained problems and adding a penalty term to the loss function. The optimization of small local sub-problems are coordinated to find a solution to a large global problem.
The scaled augmented Lagrangian takes the form:

$$L_\rho(\boldsymbol{\Theta}, \mathbf{Z}, \mathrm{U}) = \mathrm{tr}(\mathbf{S}\boldsymbol{\Theta}) - \log(\det \boldsymbol{\Theta}) + h(\mathbf{Z}) + \frac{\rho}{2}||\boldsymbol{\Theta} - \mathbf{Z} + \mathbf{U}||_F^2 \tag{3.4.1}$$

where $\rho > 0$ is the step size, $\mathbf{Z}$ is the matrix of auxiliary variables, $\mathbf{U}$ is the matrix of dual variables, $h(\cdot)$ is a penalizing function, the operator $||\cdot||_F$ is the Frobenius norm and $\mathbf{S}$ is the empirical covariance matrix.
In the ADMM algorithm, which is bond to converge to the global optimum, the L_ρ is sequentially and iteratively optimized for $\boldsymbol{\Theta}$, $\mathbf{Z}$ and $\mathbf{U}$ under the constraints that $\boldsymbol{\Theta} \succ 0$ and $\boldsymbol{\Theta}^{(m)} = \mathbf{Z}^{(m)}$ for $m = 1, 2, \cdots$; as synthesised in **Algorithm 2**. The solutions of the sequential optimization steps will be discussed in greater details in Section 4.1.

Algorithm 2: Alternating Direction of Multipliers Method (ADMM) algorithm

1: Initialize $\hat{\boldsymbol{\Theta}}^{(m=0)} = \mathbf{I}, \hat{\mathbf{U}}^{(m=0)} = \mathbf{0}, \hat{\mathbf{Z}}^{(m=0)} = \mathbf{0}$
2: Select a scalar $\rho > 0$ and a scalar $\lambda > 0$
3: **for** $m = 1, 2, \ldots$ **do**
4: update $\hat{\boldsymbol{\Theta}}^{(m)}$ as $\arg\min_{\boldsymbol{\Theta}} \left\{ \mathrm{tr}(\mathbf{S}\boldsymbol{\Theta}) - \log(\det \boldsymbol{\Theta}) + \frac{\rho}{2}||\boldsymbol{\Theta} - \hat{\mathbf{Z}}^{(m-1)} + \hat{\mathbf{U}}^{(m-1)}||_F^2 \right\}$;
5: Update $\hat{\mathbf{Z}}^{(m)}$ as $\arg\min_{\mathbf{Z}} \left\{ \frac{\rho}{2}||\mathbf{Z} - \left(\hat{\boldsymbol{\Theta}}^{(m-1)} + \hat{\mathbf{U}}^{(m-1)}\right)||_F^2 + h(\mathbf{Z}) \right\}$;
6: Update $\hat{\mathbf{U}}^{(m)}$ as $\hat{\mathbf{U}}^{(m-1)} + \hat{\boldsymbol{\Theta}}^{(m)} - \hat{\mathbf{Z}}^{(m)}$;
7: Check convergence conditions
8: **end for**

3.5 Available tools for GGM estimation

Several R packages have been developed to estimate sparse inverse covariant matrices in the context of GGM, among them:

- the *glasso* package (Friedman *et al.*, 2019), based on the work of Friedman *et al.* (2008) uses the coordinate descent procedure for the lasso, and is useful for estimating sparse undirected graphs. The Meinshausen *et al.* (2006) approximation is also implemented. The latest package version includes the block diagonal screening rule (Witten *et al.*, 2011) to speed up computations;
- the *glassoFast* package (Sustik *et al.*, 2018), also based on the work of Friedman *et al.* (2008), was further developed for faster computations;
- the *huge* package (Jiang *et al.*, 2020), also based on Friedman *et al.* (2008) and Liu *et al.* (2010), was developed with extra features and a minor convergence fix;
- the *hglasso* package (Ming Tan, 2019) offers the hub lasso approach for sparse precision matrices, implementing an ADMM algorithm, estimation as described in Ming Tan *et al.* (2014);
- the *sparsenetgls* package (Zeng *et al.*, 2021) in the Bioconductor library also applies the *glasso* solution and provides some extra features.

Of note, the *GraphicalLasso* function in the Python Scikit-Learn package also offers solutions to sparse precision matrices estimation (Pedregosa *et al.*, 2011).

3.6 Groups of Gaussian graphs

In the previous sections, the estimation of one single Gaussian graphical model was addressed. However, data can be collected from multiple groups of individuals which share some common traits and differ on others (Qiu *et al.*, 2016). The fundamental hypothesis of these estimation methods is that each trait follows the same distribution across the observations. However, this assumption might be invalid.
In an example reviewed by Friston *et al.* (2011), brain connectivity networks of subjects are expected to be similar if these share covariate features (*e.g.* demographic,...) but may vary due to non-common features (*e.g.* health status,...) between groups of subjects.

Under these settings, estimating one single graphical model based on pooled data might lead to estimates that do not reflect the true state of the network in either group. On the other hand, estimating the graphical models separately for each group of subjects might lead to dissimilarities between the adjacent graphical models where, actually, these should be identical.

Essentially, one might then be tempted to jointly estimate groups of graphical models to preserve similarities and dissimilarities between groups. However, this task grows in complexity with the number of graphs involved.

Several approaches were developed to concomitantly estimate multiple Gaussian graphical models. Among those, important research papers for this master thesis are described in the next sections where the term "groups" will be interchangeably referred as classes. Each group is identified in mathematical expressions by a (k) superscript, *e.g.* $\boldsymbol{\Theta}^{(k)}$ with $k = 1, \cdots, K$; the set of groups represented with braces, *e.g.* $\{\boldsymbol{\Theta}\} \equiv \{\boldsymbol{\Theta}^{(1)}, \cdots, \boldsymbol{\Theta}^{(K)}\}$.

3.6.1 Joint GGM estimation

The method developed by Guo *et al.* (2011) was aimed at preserving the common structure while allowing for differences between groups. In their approach the joint estimation is achieved through a hierarchical penalty. The proposed joint optimization function takes the following form:

$$\underset{\boldsymbol{\Omega},\{\boldsymbol{\Gamma}\}}{\text{minimize}} \left\{ -\sum_{k=1}^{K} \left[\log(\det \boldsymbol{\Theta}^{(k)}) - \operatorname{tr}\left(\mathbf{S}^{(k)}\boldsymbol{\Theta}^{(k)}\right)\right] + \eta_1 \sum_{a\neq b} \omega_{a,b} + \eta_2 \sum_{a\neq b}\sum_{k=1}^{K} |\gamma_{a,b}^{(k)}| \right\}$$

where $\{\boldsymbol{\Gamma}\}$ is the set of $(\gamma_{a,b}^{(k)})_{p\times p}$ matrices with $k = 1, 2, \ldots, K$; $\boldsymbol{\Theta}^{(k)} \equiv (\boldsymbol{\Sigma}^{(k)})^{-1} = (\theta_{a,b}^{(k)})_{p\times p}$ and the latter was re-parametrized as $\theta_{a,b}^{(k)} = \omega_{a,b}\gamma_{a,b}^{(k)}$.

The tuning parameter η_1 controls the common sparsity (*i.e.* $\omega_{a,b} = 0$) which preserves the common structure across the K groups, while the parametrization parameter η_2 allows zeroing of elements in some groups and not others; allowing for structure differences between groups. The so-called *hierarchical group* penalty proposed by Guo was reformulated, for computational purposes, as follows:

$$h_{Guo}(\{\boldsymbol{\Theta}\}) = \lambda \sum_{a\neq b} \sqrt{\sum_{k=1}^{K} |\theta_{a,b}^{(k)}|}. \tag{3.6.1}$$

The penalty (3.6.1) is not convex, therefore local minima could be reached instead of the global one.

The $\ell_{1,s}$-norm penalizing function proposed in Honorio *et al.* (2015) was motivated from the multi-task learning literature (Crawshaw, 2020). The family of convex penalty functions takes the form:

$$h_{Honorio}(\{\boldsymbol{\Theta}\}) = \lambda \sum_{a\neq b} \left(\sum_{k=1}^{K} |(\theta_{a,b}^{(k)})^s| \right)^{1/s} \tag{3.6.2}$$

with $s > 1$. Their research was limited to $s = 2$ and $s = \infty$. The solution to their non-smooth convex optimization problem is searched using the block coordinate descent

method.

In Danaher *et al.* (2014), two sensibly different approaches were proposed: the Fused Graphical Lasso (FGL) and the Grouped Graphical Lasso (GGL).
Their common convex penalty function takes the form:

$$h_{Danaher}(\{\boldsymbol{\Theta}\}) = \underbrace{\lambda_1 \sum_{k=1}^{K} \sum_{a \neq b} |\theta_{a,b}^{(k)}|}_{\text{I}} + \underbrace{\tilde{h}(\{\boldsymbol{\Theta}\})}_{\text{II}} .$$

The first term of the penalty promotes **sparsity** in the precision matrices and is common to both proposals.
The second term of the penalty, $\tilde{h}(\{\boldsymbol{\Theta}\})$, defined in the FGL proposal as

$$\tilde{h}_{FGL}(\{\boldsymbol{\Theta}^{(k)}, \boldsymbol{\Theta}^{(k')}\}) = \lambda_2 \sum_{a \neq b} \sum_{k=1}^{K} |\theta_{a,b}^{(k)} - \theta_{a,b}^{(k')}| \text{ with } k \neq k', \tag{3.6.3}$$

promotes **similarity between pairs**.

The GGL proposal encourages **similarity across all groups** with the following penalty

$$\tilde{h}_{GGL}(\{\boldsymbol{\Theta}\}) = \lambda_2 \sum_{a \neq b} \left(\sum_{k=1}^{K} {\theta_{a,b}^{(k)}}^2 \right)^{1/2} \tag{3.6.4}$$

which is similar to expression (3.6.2) for $s = 2$.

3.6.2 Differential GGM estimation

Alternatively to jointly estimating the precision matrices, one can instead be interested by network differentials, *i.e.* edges that differ across groups.

The differential estimation method developed in Lee *et al.* (2015) first considers the common and unique precision matrices then their sparse constrained optimization. These were defined, respectively, as follows

$$\boldsymbol{\Theta}_{comm} := \frac{1}{K} \sum_{k=1}^{K} \boldsymbol{\Theta}^{(k)} \quad \text{and} \quad \boldsymbol{\Theta}_{diff}^{(k)} := \boldsymbol{\Theta}^{(k)} - \boldsymbol{\Theta}_{comm}$$

with, by definition, $\sum_{k=1}^{K} \boldsymbol{\Theta}_{diff}^{(k)} = \mathbf{0}$.
The optimization problem of estimating $\{\boldsymbol{\Theta}_{comm}, \boldsymbol{\Theta}_{diff}^{(1)}, \ldots, \boldsymbol{\Theta}_{diff}^{(K)}\}$ is then formulated as follows

$$\min\{||\boldsymbol{\Theta}_{comm}||_1 + \lambda \sum_{k=1}^{K} ||\boldsymbol{\Theta}_{diff}^{(k)}||_1\}$$

subject to

$$|\frac{1}{K}\sum_{k=1}^{K}\Big\{\mathbf{S}^{(k)}(\mathbf{\Theta}_{comm}+\mathbf{\Theta}_{diff}^{(k)})-\mathbf{I}\Big\}|_\infty \leq \eta_1 \quad \text{and} \quad |\Big\{\mathbf{S}^{(k)}(\mathbf{\Theta}_{comm}+\mathbf{\Theta}_{diff}^{(k)})-\mathbf{I}\Big\}|_\infty \leq \eta_2$$

where the operator $|\cdot|_\infty$ is the elementwise ℓ_∞ norm. The first and second constraints enforce the closeness of the estimates to the inverse sample covariance matrices, $(\mathbf{S}^{(k)})^{-1}$, globally and locally, respectively. As the second constraint is the stringiest, only pairs (η_1, η_2) satisfying $\eta_1 \leq \eta_2$ were considered.

A more direct estimation of differential networks proposed by Zhao *et al.* (2014) . They defined those as the difference between two precision matrices, denoted

$$\mathbf{\Theta}_{diff}^{(kk')} = \mathbf{\Theta}^{(k)} - \mathbf{\Theta}^{(k')} \text{ with } k \neq k'.$$

The ℓ_1-constrained optimization problem, to estimate differential precision matrices, was formulated as follows

$$\arg\min |\mathbf{\Theta}_{diff}^{(kk')}|_1 \quad \text{s.t}$$
$$|\Big(\hat{\mathbf{\Sigma}}^{(k)}\otimes\hat{\mathbf{\Sigma}}^{(k')}\Big)\,\mathrm{vec}\Big(\mathbf{\Theta}_{diff}^{(kk')}\Big)-\mathrm{vec}\Big(\hat{\mathbf{\Sigma}}^{(k)}-\hat{\mathbf{\Sigma}}^{(k')}\Big)|_\infty \leq \lambda$$

where $\otimes$ is the Kronecker product and $\mathrm{vec}(\cdot)$ is the vectorization operator.

In cases where $\mathbf{\Theta}_{diff}^{(kk')}$ is symmetric, the optimization problem can be simplified into

$$\arg\min |\beta|_1 \quad \text{s.t} \quad \begin{cases} |\mathbf{L}^T\hat{\mathbf{\Sigma}}_{Kr}\mathbf{L}\beta-\mathbf{L}\hat{b}|_\infty \leq \lambda, \text{on off-diagonal elements } \mathbf{\Theta}_{diff,a\neq b}^{(kk')} \\ |\mathbf{L}^T\hat{\mathbf{\Sigma}}_{Kr}\mathbf{L}\beta-\mathbf{L}\hat{b}|_\infty \leq \frac{\lambda}{2}, \text{on the diagonal elements } \mathbf{\Theta}_{diff,a=b}^{(kk')} \end{cases}$$

where $\hat{\mathbf{\Sigma}}_{Kr} = \hat{\mathbf{\Sigma}}^{(k)}\otimes\hat{\mathbf{\Sigma}}^{(k')}$, $\hat{b} = \mathrm{vec}\Big(\hat{\mathbf{\Sigma}}^{(k)}-\hat{\mathbf{\Sigma}}^{(k')}\Big)$,

$$\beta = \mathrm{vec}\Big(upper.triangle(\mathbf{\Theta}_{diff}^{(kk')})\Big),$$

a vector of dimension $p(p+1)/2\times 1$; $\mathbf{L}$ is of dimension $p^2\times p(p+1)/2$ with $L_{ab,ab} = L_{ab,ba} = 1$ and otherwise equal to zero.
This simplified optimization problem, however, requires that $\mathbf{\Theta}^{(k)}$ and $\mathbf{\Theta}^{(k')}$ have comparable sparsity, and that the parameters of the precision matrices are not too large.

3.6.3 Methods comparison of GGM estimation

It is generally accepted that the joint estimation of groups of precision matrices outperforms separate estimation. In the simulated data example from Guo *et al.* (2011), shown in Figure

3.6.1, the average true positive rate against the average false positive rate[4] indicated that joint estimations lead to better results as long as the proportion of common traits is large, *i.e.* ρ is small.

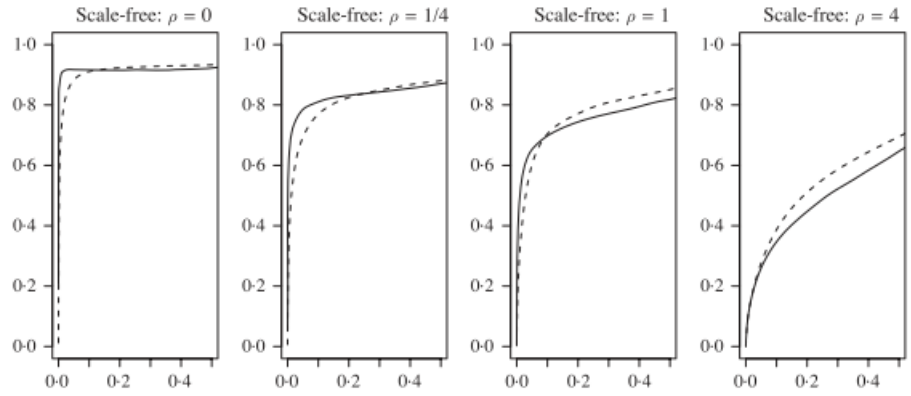

Figure 3.6.1: **Receiver operating characteristic curves comparison of the joint and separate estimations**. Parameters ρ indicate the proportion of common traits across the groups. A small ρ denotes that the precision matrices share many common traits. The x- and y-axis are the true and false positive rates, respectively. The solid and the dashed lines indicate the joint and separate estimations, respectively. The curves were generated using a range of the tuning parameter λ values. *Figure from Guo et al. (2011).*

The two penalties proposed by Danaher *et al.* (2014), *i.e.* the FGL and the GGL, are compared, on simulated data, to the one introduced by Guo *et al.* (2011) and the disjoint estimation, *i.e.* the Graphical Lasso, in Figure 3.6.2. The number of edges (TP and FP) increases with smaller sparsity tuning parameter λ_1 values. The results of the GGL method were comparable to the method proposed by Guo *et al.* (2011) for some λ_1 and λ_2 combination values. For larger similarity tuning parameter λ_2 values, the FGL penalty dominated over the hierarchical and the GGL.
In both GGL and FGL, λ_1 disjointly drives edges to zero.
In FGL, λ_2 enforces similarity across pairs of classes while it drives edges to zero simultaneously across all classes in GGL. To uncouple these effects in GGL, the authors re-formulated the tuning parameters as $\omega_1 = \lambda_1 + \frac{1}{\sqrt{2}}\lambda_2$ and $\omega_2 = \frac{1}{\sqrt{2}}\lambda_2/\omega_1$ which reflect the sparsity and similarity, respectively.
The single tuning parameter in Guo *et al.* (2011) drives sparsity which leads to similarity in sparse matrices. The proposal of Danaher *et al.* (2014) has the advantage of separately, at least to some extend for the GGL, controlling the sparsity and similarity.

[4]These notions are clarified in Section 5.1.

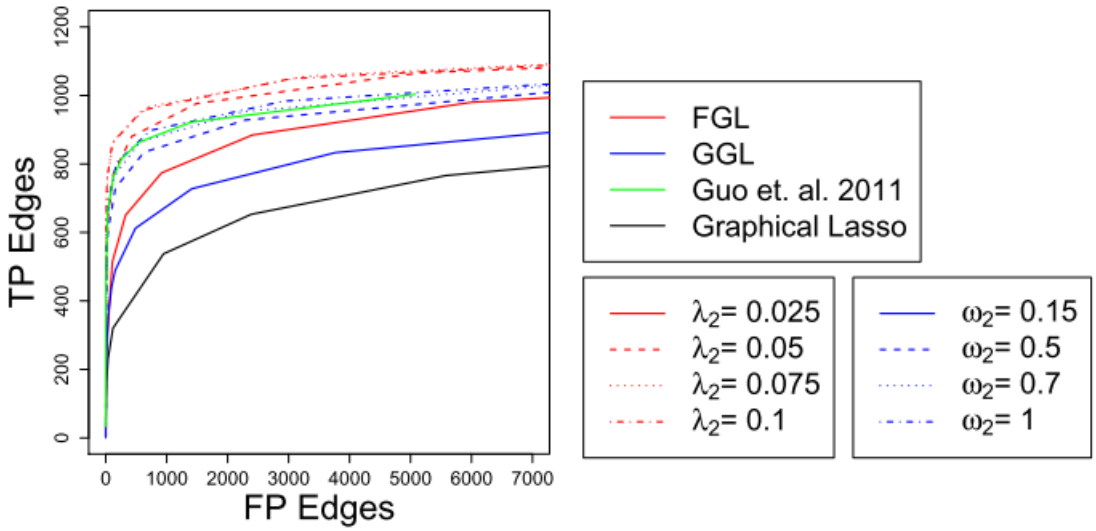

Figure 3.6.2: **Receiver operating characteristic curves comparison of the joint and disjoint estimations**. The true positive and true negative edges were computed for a range of the tuning parameters λ_1 and λ_2. The colors indicate the estimation methods. These results were generated using a synthetic data matrix of size $p = 500$ and $n = 150$. *Figure from Danaher et al. (2014).*

A key strength of the direct estimation is that it does not require precision matrices to be sparse, and therefore tolerates the underlining networks to contain hub nodes. However, direct estimation still requires that $\mathbf{\Theta}^{(k)}$ and $\mathbf{\Theta}^{(k')}$ have comparable sparsity, which is a reasonable assumption given the unlikelihood that networks differ overall.
In addition, the number of elements to be estimated are obviously greatly reduced as one differential matrix needs to be considered for each pair of precision matrix. Furthermore, the method in Zhao *et al.* (2014) can determine, with high confidence, the signs of sufficiently large non-zero coefficients. On the down side, differential networks estimation only allows for comparison of pair of networks which limits the opportunity of collecting large amount of information to estimate the common traits.

A comparison of the direct estimation and the FGL one, on simulated data, indicated similar performances. As shown in Figure 3.6.3, the true positive/true negative rates were comparable while the accuracy of the estimated parameters were slightly better. Of note, the FGL penalty (3.6.3) in Danaher *et al.* (2014) takes the differential to encourage similarity across precision matrices but does not compute an estimation of it.

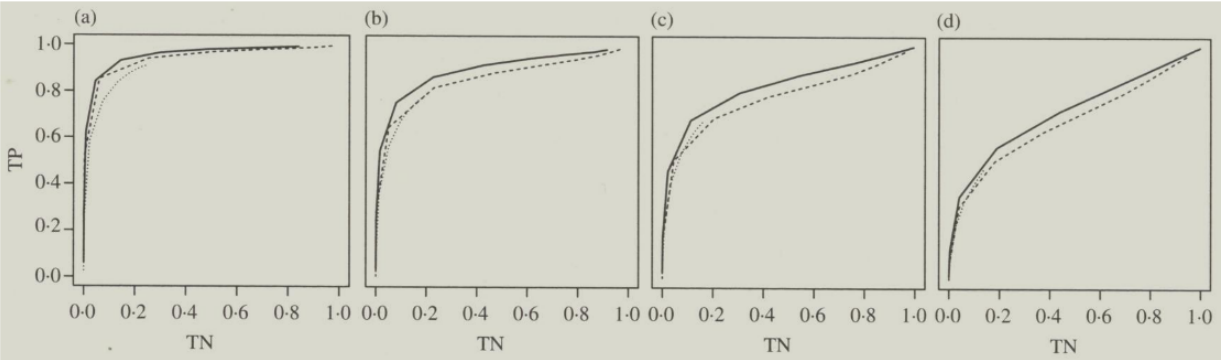

Figure 3.6.3: **Receiver operating characteristic curves comparison of the direct and the FGL estimations**. The differential precision matrices were obtained from simulated data with $n = 100$ and $p = 40$ (a), $p = 60$ (b), $p = 90$ (c) or $p = 120$ (d). The true positive and true negative rates were computed for various values of the tuning parameter λ and λ_2 for the direct and FGL estimation, respectively. The solid lines indicate the direct estimation and the dashed lines represent the FGL estimations for $\lambda_1 = 0$ and dotted lines $\lambda_1 = 0.1$, respectively. *Figure from Zhao et al. (2014).*

Although the computation speed of these methods is of importance, this was out of our master thesis scope and is therefore not discussed.

Chapter 4

Proposals

4.1 The joint optimization problem and its solution

Since we were interested in jointly estimating K distinct but related Gaussian graph models in a high-dimensional framework, different alternatives were considered:

- estimate each precision matrix separately;
- simultaneously estimate all precision matrices and **foster similarities**;
- simultaneously estimate all precision matrices and **capturing dissimilarities** across classes while maintaining their similarities.

The first option bears little interest to us since classes are related. The second approach was addressed in Danaher *et al.* (2014) by the following two penalties[1]: the fused and the grouped *graphical lasso* (FGL and GGL, respectively). The third alternative is the main focus of this master thesis.

Based on the work of Danaher *et al.* (2014), the multi-classes joint optimization problem was expressed as follows:

$$\underset{\{\boldsymbol{\Theta}\}}{\text{minimize}} \left\{ \underbrace{-\sum_{k=1}^{K} n_k \left[\log\left(\det \boldsymbol{\Theta}^{(k)}\right) - \operatorname{tr}\left(\mathbf{S}^{(k)}\boldsymbol{\Theta}^{(k)}\right) \right]}_{\mathbf{I}} + \underbrace{\lambda_1 \sum_{k=1}^{K} \sum_{a \neq b} |\theta_{a,b}^{(k)}|}_{\mathbf{IIa}} + \underbrace{\tilde{h}(\{\boldsymbol{\Theta}\})}_{\mathbf{IIb}} \right\} \quad (4.1.1)$$

with term (**I**) being the multi-class log-likelihood function, term (**IIa**) the convex sparsity penalty function and term (**IIb**) an additional convex penalty function aimed at fostering similarity in the above cited paper. The latter term is the main focus of this chapter.

[1] See Section 3.6.1.

The multi-class optimization problem can be re-formulated into the scaled augmented Lagrangian[2] (Boyd *et al.*, 2010) as

$$\begin{aligned} L_\rho(\{\boldsymbol{\Theta}\},\{\mathbf{Z}\},\{\mathbf{U}\}) = & - \sum_{k=1}^{K} n_k \left(\log\det \boldsymbol{\Theta}^{(k)} - \operatorname{tr}\left(\mathbf{S}^{(k)}\boldsymbol{\Theta}^{(k)}\right)\right) \\ & + \lambda_1 \sum_{k=1}^{K}\sum_{a\neq b} |Z_{a,b}^{(k)}| + \tilde{h}(\{\mathbf{Z}\}) \\ & + \frac{\rho}{2}\sum_{k=1}^{K} ||\boldsymbol{\Theta}^{(k)} - \mathbf{Z}^{(k)} + \mathbf{U}^{(k)}||_F^2, \end{aligned} \tag{4.1.2}$$

where $\rho > 0$ is the step size, n_k is the sample size, $\mathbf{Z}^{(k)}$ is the matrix of auxiliary variables, $\mathbf{U}^{(k)}$ is the matrix of the dual variables and $\tilde{h}(\{\mathbf{Z}\})$ corresponds to term (**IIb**) in expression (4.1.1). The solution to the optimization problem (4.1.2) can be obtained using a three steps ADMM algorithm[3]. The analytical solution to each step is developed in the next paragraphs.

The solution to the minimization with respect to $\{\boldsymbol{\Theta}\}$, at step one of the algorithm, was developed in Witten & Tibshirani (2009) as

$$\hat{\boldsymbol{\Theta}}^{(k)} = \mathbf{V}^{(k)}\tilde{\mathbf{D}}^{(k)}(\mathbf{V}^{(k)})^T$$

with $\mathbf{V}^{(k)}\mathbf{D}^{(k)}(\mathbf{V}^{(k)})^T$ resulting from the eigendecomposition of the expression

$$\mathbf{S}^{(k)} - \rho\mathbf{Z}^{(k)} + \rho\mathbf{U}^{(k)}; \tag{4.1.3}$$

$\mathbf{V}^{(k)}$ being the matrix of eigenvectors and $\mathbf{D}^{(k)}$ the matrix of eigenvalues. The diagonal matrix $\tilde{\mathbf{D}}^{(k)}$ is composed of elements $\tilde{D}_{b,b}^{(k)}$ given by

$$\frac{n_k}{2\rho}\left(-D_{b,b}^{(k)} + \sqrt{(D_{b,b}^{(k)})^2 + \frac{4\rho}{n_k}}\right)$$

with $D_{b,b}^{(k)}$ the b^{th} diagonal element of $\mathbf{D}^{(k)}$.

The solution to the minimization with respect to $\{\mathbf{Z}\}$ depends on the form of the (additional) penalty. The optimization problem can be re-formulated as follows

$$\underset{\{\mathbf{Z}\}}{\text{minimize}} \left\{ \underbrace{\frac{\rho}{2}\sum_{k=1}^{K} ||\mathbf{Z}^{(k)} - \mathbf{A}^{(k)}||_F^2}_{\textbf{III}} + \underbrace{\lambda_1 \sum_{k=1}^{K}\sum_{a\neq b} |Z_{a,b}^{(k)}|}_{\textbf{IIa}} + \underbrace{\tilde{h}(\{\mathbf{Z}\})}_{\textbf{IIb}} \right\}. \tag{4.1.4}$$

[2]For simplification, the Lagrangian $\mathcal{L}$ is divided by ρ, *i.e.* L_ρ.

[3]See Section 3.4.

where $\mathbf{A}^{(k)} = \mathbf{\Theta}^{(k)} + \mathbf{U}^{(k)}$. Minimization of expression (4.1.4) is achieved through the nullification of the partial derivatives of terms (**III**), (**IIa**) and (**IIb**).

The Frobenius norm of term (**III**) is defined as

$$(\mathbf{III}) = \mathrm{tr}\big((\mathbf{Z}^{(k)} - \mathbf{A}^{(k)})^T(\mathbf{Z}^{(k)} - \mathbf{A}^{(k)})\big) = \mathrm{tr}\Big(\mathbf{Z}^{(k)^T}\mathbf{Z}^{(k)} - \mathbf{Z}^{(k)^T}\mathbf{A}^{(k)} - \mathbf{A}^{(k)^T}\mathbf{A}^{(k)}\Big)$$

and its derivative with respect to $Z_{a,b}^{(k)}$ is given as

$$\frac{\partial(\mathbf{III})}{\partial Z_{a,b}^{(k)}} = 2(Z_{a,b}^{(k)} - A_{a,b}^{(k)}).$$

The second term (**IIa**), which encourages **sparsity**, has derivative

$$\frac{\partial(\mathbf{IIa})}{\partial Z_{a,b}^{(k)}} = \lambda_1 \frac{\partial|Z_{a,b}^{(k)}|}{\partial Z_{a,b}^{(k)}} = \lambda_1 \,\mathrm{Sgn}\Big(Z_{a,b}^{(k)}\Big),$$

with function $\mathrm{Sgn}(\cdot)$ outputting the sign of its input, under the condition that $Z_{a,b}^{(k)} \neq 0$.

The solution to expression (4.1.4) considering the GGL penalty in Danaher *et al.* (2014), *i.e.* $\tilde{h}_{GGL}(\{\mathbf{Z}\}) = \lambda_2 \sum_{a\neq b} \left(\sum_{k=1}^{K} \left(Z_{a,b}^{(k)}\right)^2\right)^{1/2}$, was given as

$$\mathrm{Soft}\left(A_{a,b}^{(k)}, \frac{\lambda_1}{\rho}\right)\left(1 - \frac{\lambda_2}{\rho\sqrt{\sum_k^K \mathrm{Soft}\left(A_{a,b}^{(k)}, \lambda_1/\rho\right)^2}}\right)_+ .$$

As mentioned in Section 3.6.1, the GGL penalty encourages similarity across all groups but does so also by contributing to the sparsity of the matrices. As stated in Chapter 1, the main objective of this work was to devise an estimation method that best mirrors the probabilistic dependence structures of each group, *i.e.* the graphical models should reflect the sparsity pattern in terms of similarities and differences between groups.

We based our work on Danaher *et al.* (2014), borrowing the optimization problem in expression (4.1.4) and considering alternative penalties $\tilde{h}(\{\mathbf{Z}\})$. Similar to the approach taken in Honorio *et al.* (2015), the following family of penalty functions

$$\tilde{h}_{q,r}(\{\mathbf{Z}\}) = \lambda_2 \sum_{a\neq b} \left(\sum_{k=1}^{K} \left(Z_{a,b}^{(k)}\right)^2\right)^{q/r}, \tag{4.1.5}$$

with parameters $q \in \mathbb{N}$ and $r \in \mathbb{N}_+$, was considered. Of note, among the possible functions, the GGL penalty has parameters $q = 1$ and $r = 2$. The proposal functions $\tilde{h}_{q,r}(\{\mathbf{Z}\})$ and their q/r quotient are referred to as, respectively, penalty and exponent in the text.

The intuition was that among this family of convex penalties, which include the GGL, there might be at least one other member displaying better performances.

Our goal was to select at least one exponent q/r mounting to an estimator that would outperform[4] the GGL approach. Given the dimension and complexity of the optimization problem, it was not possible to analytically select the optimal exponent. Besides, it was not obvious how to derive the solution of the generic penalty function $\tilde{h}_{q,r}(\{\mathbf{Z}\})$ that would allow for the exponent selection through screening (in a simulation set-up). This led us to having to "manually" pick few penalties for testing. The choice was mostly driven by

- the amplitude of the exponent, *i.e.* ideally at least a lower and a higher quotient than the GGL exponent, *i.e.* $q/r = 1/2$.
- the solution to the optimization problem should be analytically derivable.

Considering these aspects, three proposals were put forward for study:

$$\tilde{h}_{(1)}(\{\mathbf{Z}\}) = \lambda_2 \sum_{a \neq b} \left(\sum_k {Z_{a,b}^{(k)}}^2 \right)^{3/2} \tag{4.1.6}$$

$$\tilde{h}_{(2)}(\{\mathbf{Z}\}) = \lambda_2 \sum_{a \neq b} \left(\sum_k {Z_{a,b}^{(k)}}^2 \right)^{-1/2} \tag{4.1.7}$$

$$\tilde{h}_{(3)}(\{\mathbf{Z}\}) = \lambda_2 \sum_{a \neq b} \left(\sum_k {Z_{a,b}^{(k)}}^2 \right)^{2} \tag{4.1.8}$$

Proposals $\tilde{h}_{(1)}$ and $\tilde{h}_{(3)}$ are the candidates with a higher exponent than the GGL penalty. Only penalty $\tilde{h}_{(2)}$ with a lower exponent than the GGL penalty was found to meet our requirements. The solution to each of these proposals is developed in the next sections. For each, the first step was to differentiate term $\tilde{h}(\{\mathbf{Z}\})$ with regards to $\{\mathbf{Z}\}$. The results is then assembled with the derivatives of terms (**IIa**) and (**III**), provided in the previous paragraphs, to minimize expression (4.1.4).

[4]The performance criteria are discussed in Section 5.1.

4.2 Proposal 1

Considering penalty $\tilde{h}_{(1)}$ in (4.1.6), *i.e.* $\lambda_2 \sum_{a\neq b} \left(\sum_k {Z_{a,b}^{(k)}}^2\right)^{3/2}$, its derivative with respect to $\{\mathbf{Z}\}$ is given as

$$\frac{\partial \tilde{h}_{(1)}/\rho}{\partial Z_{a,b}^{(k)}} = \frac{3\lambda_2}{2\rho}\sqrt{\sum_k {Z_{a,b}^{(k)}}^2}\ \left(2\, Z_{a,b}^{(k)}\right) = \frac{3\lambda_2}{\rho} Z_{a,b}^{k}\sqrt{\sum_k {Z_{a,b}^{(k)}}^2}.$$

The optimization of expression (4.1.4) with respect to $\{\mathbf{Z}\}$ is developed as follows

$$\frac{\partial \mathcal{L}/\rho}{\partial Z_{a,b}^{(k)}} = (Z_{a,b}^{(k)} - A_{a,b}^{(k)}) + \frac{\lambda_1}{\rho}\,\mathrm{Sgn}\Big(Z_{a,b}^{(k)}\Big) + \frac{3\lambda_2}{\rho} Z_{a,b}^{(k)}\sqrt{\sum_k {Z_{a,b}^{(k)}}^2} = 0$$

$$\Leftrightarrow \qquad Z_{a,b}^{(k)}\left(1 + \frac{3\lambda_2}{\rho}\sqrt{\sum_k {Z_{a,b}^{(k)}}^2}\right) = A_{a,b}^{(k)} - \frac{\lambda_1}{\rho}\,\mathrm{Sgn}\Big(Z_{a,b}^{(k)}\Big)$$

$$\Leftrightarrow \qquad Z_{a,b}^{(k)}\left(1 + \frac{3\lambda_2}{\rho}\sqrt{\sum_k {Z_{a,b}^{(k)}}^2}\right) = \mathrm{Soft}\left(A_{a,b}^{(k)}, \frac{\lambda_1}{\rho}\right) \tag{4.2.1}$$

$$\Leftrightarrow \qquad {Z_{a,b}^{(k)}}^2\left(1 + \frac{3\lambda_2}{\rho}\sqrt{\sum_k {Z_{a,b}^{(k)}}^2}\right)^2 = \mathrm{Soft}^2\left(A_{a,b}^{(k)}, \frac{\lambda_1}{\rho}\right)$$

$$\Leftrightarrow \qquad \sum_{k=1}^{K} {Z_{a,b}^{(k)}}^2\left(1 + \frac{3\lambda_2}{\rho}\sqrt{\sum_k {Z_{a,b}^{(k)}}^2}\right)^2 = \sum_{k=1}^{K}\mathrm{Soft}^2\left(A_{a,b}^{(k)}, \frac{\lambda_1}{\rho}\right)$$

$$\Leftrightarrow \qquad \sqrt{\sum_{k=1}^{K} {Z_{a,b}^{(k)}}^2}\left(1 + \frac{3\lambda_2}{\rho}\sqrt{\sum_k {Z_{a,b}^{(k)}}^2}\right) = \sqrt{\sum_{k=1}^{K}\mathrm{Soft}^2\left(A_{a,b}^{(k)}, \frac{\lambda_1}{\rho}\right)}. \tag{4.2.2}$$

Let,

$$\sqrt{\sum_{k=1}^{K} {Z_{a,b}^{(k)}}^2} = x \qquad \textit{and} \qquad \sqrt{\sum_{k=1}^{K}\mathrm{Soft}^2\left(A_{a,b}^{(k)}, \frac{\lambda_1}{\rho}\right)} = c.$$

where $x \geq 0$ (condition 1) and $c \geq 0$. Then equation (4.2.2) becomes,

$$x\left(1 + \frac{3\lambda_2}{\rho}x\right) - c = \frac{3\lambda_2}{\rho}x^2 + x - c = 0.$$

The root of x was obtained as follow:

$$x = \rho\frac{-1 + \sqrt{\Delta}}{2 \cdot 3\lambda_2},$$

with $\Delta = 1 + \frac{12\lambda_2 c}{\rho}$ and under the condition that $\Delta > 0$

$$\Leftrightarrow c = \sqrt{\sum_{k=1}^{K} \text{Soft}^2\left(A_{a,b}^{(k)}, \frac{\lambda_1}{\rho}\right)} > -\frac{\rho}{12\lambda_2}$$

which always holds since $c \geq 0$ and both $\lambda_2 > 0$ and $\rho > 0$. It should be noted that condition $x \geq 0$ is verified since $\sqrt{\Delta} > 1$ and therefore the other root, *i.e.* $\rho\frac{-1-\sqrt{\Delta}}{6\lambda_2}$, does not meet the requirement.

Replacing $\sqrt{\sum_k {Z_{a,b}^{(k)}}^2}$ with x in equation (4.2.1) leads to

$$\begin{aligned} Z_{a,b}^{(k)}\left(1 + \frac{3\lambda_2}{\rho}x\right) &= \text{Soft}\left(A_{a,b}^{(k)}, \frac{\lambda_1}{\rho}\right) \\ \Leftrightarrow \qquad Z_{a,b}^{(k)} &= \text{Soft}\left(A_{a,b}^{(k)}, \frac{\lambda_1}{\rho}\right)\left(1 + \frac{3\lambda_2}{\rho}x\right)^{-1} \end{aligned}$$

under the condition that

$$x \neq -\frac{\rho}{3\lambda_2}, \text{ which always holds since } x = \sqrt{\sum_{k=1}^{K} {Z_{a,b}^{(k)}}^2} \geq 0.$$

4.3 Proposal 2

Considering penalty $\tilde{h}_{(2)}$ in (4.1.7), *i.e.* $\lambda_2 \sum_{a \neq b}\left(\sum_k {Z_{a,b}^{(k)}}^2\right)^{-1/2}$, its derivative with regards to $\{\mathbf{Z}\}$ is given as

$$\frac{\partial \tilde{h}_{(2)}/\rho}{\partial Z_{a,b}^{(k)}} = \frac{-\lambda_2}{2\rho}\left(\sum_k {Z_{a,b}^{(k)}}^2\right)^{-3/2}\left(2\, Z_{a,b}^{(k)}\right) = \frac{-\lambda_2}{\rho} Z_{a,b}^{(k)}\left(\sum_k {Z_{a,b}^{(k)}}^2\right)^{-3/2}.$$

The optimization of expression (4.1.4) with respect to $\{\mathbf{Z}\}$ is developed as follows

$$\frac{\partial \mathcal{L}/\rho}{\partial Z_{a,b}^{(k)}} = (Z_{a,b}^{(k)} - A_{a,b}^{(k)}) + \frac{\lambda_1}{\rho} Sgn(Z_{a,b}^{(k)}) - \frac{\lambda_2}{\rho} Z_{a,b}^{(k)}\left(\sum_k {Z_{a,b}^{(k)}}^2\right)^{-3/2} = 0$$

$$\Leftrightarrow \quad Z_{a,b}^{(k)}\left[1-\frac{\lambda_2}{\rho}\left(\sum_k Z_{a,b}^{(k)^2}\right)^{-3/2}\right]=A_{a,b}^{(k)}-\frac{\lambda_1}{\rho}\,\mathrm{Sgn}\left(Z_{a,b}^{(k)}\right)$$

$$\Leftrightarrow \quad Z_{a,b}^{(k)}\left[1-\frac{\lambda_2}{\rho}\left(\sum_k Z_{a,b}^{(k)^2}\right)^{-3/2}\right]=\mathrm{Soft}\left(A_{a,b}^{(k)},\frac{\lambda_1}{\rho}\right) \tag{4.3.1}$$

$$\Leftrightarrow \quad Z_{a,b}^{(k)^2}\left[1-\frac{\lambda_2}{\rho}\left(\sum_k Z_{a,b}^{(k)^2}\right)^{-3/2}\right]^2=\mathrm{Soft}^2\left(A_{a,b}^{(k)},\frac{\lambda_1}{\rho}\right)$$

$$\Leftrightarrow \quad \sum_{k=1}^{K} Z_{a,b}^{(k)^2}\left[1-\frac{\lambda_2}{\rho}\left(\sum_k Z_{a,b}^{(k)^2}\right)^{-3/2}\right]^2=\sum_{k=1}^{K}\mathrm{Soft}^2\left(A_{a,b}^{(k)},\frac{\lambda_1}{\rho}\right) \tag{4.3.2}$$

$$\Leftrightarrow \quad \sqrt{\sum_{k=1}^{K} Z_{a,b}^{(k)^2}}\left[1-\frac{\lambda_2}{\rho}\left(\sum_k Z_{a,b}^{(k)^2}\right)^{-3/2}\right]=\sqrt{\sum_{k=1}^{K}\mathrm{Soft}^2\left(A_{a,b}^{(k)},\frac{\lambda_1}{\rho}\right)} \tag{4.3.3}$$

Let,

$$\sqrt{\sum_{k=1}^{K} Z_{a,b}^{(k)^2}}=x \qquad \text{and} \qquad \sqrt{\sum_{k=1}^{K}\mathrm{Soft}^2\left(A_{a,b}^{(k)},\frac{\lambda_1}{\rho}\right)}=c$$

where $x \geq 0$ (*condition 1*) and $c \geq 0$. Of note, the development from step (4.3.2) to (4.3.3) assumes that $\left[1-\frac{\lambda_2}{\rho}\left(\sum_k Z_{a,b}^{(k)^2}\right)^{-3/2}\right]>0$, *i.e.* $\sqrt{\sum_k Z_{a,b}^{(k)^2}}>\sqrt[3]{\frac{\lambda_2}{\rho}}$ (*condition 2*).

Then equation (4.3.3) becomes,

$$x\left(1-\frac{\lambda_2}{\rho x^3}\right)-c=x^3-cx^2-\frac{\lambda_2}{\rho}=0$$

The root of x was obtained using the Tartaglia-Cardano method, with

$$A=1,\ B=-c,\ C=0 \text{ and } D=-\left(\frac{\lambda_2}{\rho}\right),$$

$$x=\sqrt[3]{-\frac{Q}{2}-\sqrt{\frac{Q^2}{4}+\frac{P^3}{27}}}+\sqrt[3]{-\frac{Q}{2}+\sqrt{\frac{Q^2}{4}+\frac{P^3}{27}}}-\frac{B}{3A} \tag{4.3.4}$$

subject to $4P^3 + 27Q^2 > 0$ (*condition 3*) in which case x has only one real solution and with

$$P = \frac{3AC - B^2}{3A^2} = -\frac{c}{3} \qquad \text{and}$$

$$Q = \frac{2B^3 - 9ABC - 27A^2D}{27A^3} = \left(\frac{\lambda_2}{\rho}\right) - \frac{2c^3}{27}.$$

No obvious analytical expression to *condition 1* and *condition 3*, *i.e.* respectively $x \geq 0$ and $4P^3 + 27Q^2 > 0$, were found. These were tested numerically.

Replacing $\sqrt{\sum_k Z_{a,b}^{(k)^2}}$ with x in equation (4.3.1) leads to

$$\begin{aligned} Z_{a,b}^{(k)}\left(1 - \frac{\lambda_2}{\rho x^3}\right) &= \text{Soft}\left(A_{a,b}^{(k)}, \frac{\lambda_1}{\rho}\right) \\ \Leftrightarrow \qquad Z_{a,b}^{(k)} &= \text{Soft}\left(A_{a,b}^{(k)}, \frac{\lambda_1}{\rho}\right)\left(1 - \frac{\lambda_2}{\rho x^3}\right)_+^{-1}, \end{aligned}$$

i.e. under the condition that

$$x > \sqrt[3]{\frac{\lambda_2}{\rho}}$$

which is the necessary *condition 2* to obtain expression (4.2.2).

4.4 Proposal 3

Considering penalty $\tilde{h}_{(3)}$ in (4.1.8), *i.e.* $\lambda_2 \sum_{a \neq b} \left(\sum_k Z_{a,b}^{(k)^2}\right)^2$, its derivative with respect to $\{\mathbf{Z}\}$ is given as

$$\frac{\partial \tilde{h}_{(2)}/\rho}{\partial Z_{a,b}^{(k)}} = \frac{2\lambda_2}{\rho}\left(\sum_k Z_{a,b}^{(k)^2}\right)\left(2\, Z_{a,b}^{(k)}\right) = \frac{4\lambda_2}{\rho} Z_{a,b}^{k}\left(\sum_k Z_{a,b}^{(k)^2}\right)$$

The optimization of expression (4.1.4) with respect to $\{\mathbf{Z}\}$ is developed as follow

$$\frac{\partial \mathcal{L}/\rho}{\partial Z_{a,b}^{(k)}} = (Z_{a,b}^{(k)} - A_{a,b}^{(k)}) + \frac{\lambda_1}{\rho}\,\text{Sgn}\left(Z_{a,b}^{(k)}\right) + \frac{4\lambda_2}{\rho} Z_{a,b}^{(k)}\left(\sum_k Z_{a,b}^{(k)^2}\right) = 0$$

$$\Leftrightarrow \quad Z_{a,b}^{(k)}\left[1+\frac{4\lambda_2}{\rho}\left(\sum_k Z_{a,b}^{(k)^2}\right)\right] = A_{a,b}^{(k)} - \frac{\lambda_1}{\rho}\,\mathrm{Sgn}\left(Z_{a,b}^{(k)}\right)$$

$$\Leftrightarrow \quad Z_{a,b}^{(k)}\left[1+\frac{4\lambda_2}{\rho}\left(\sum_k Z_{a,b}^{(k)^2}\right)\right] = \mathrm{Soft}\left(A_{a,b}^{(k)}, \frac{\lambda_1}{\rho}\right) \tag{4.4.1}$$

$$\Leftrightarrow \quad Z_{a,b}^{(k)^2}\left[1+\frac{4\lambda_2}{\rho}\left(\sum_k Z_{a,b}^{(k)^2}\right)\right]^2 = \mathrm{Soft}^2\left(A_{a,b}^{(k)}, \frac{\lambda_1}{\rho}\right)$$

$$\Leftrightarrow \quad \sqrt{\sum_k Z_{a,b}^{(k)^2}}\left[1+\frac{4\lambda_2}{\rho}\left(\sum_k Z_{a,b}^{(k)^2}\right)\right] = \sqrt{\sum_{k=1}^{K}\mathrm{Soft}^2\left(A_{a,b}^{(k)}, \frac{\lambda_1}{\rho}\right)} \tag{4.4.2}$$

Let,

$$\sqrt{\sum_k Z_{a,b}^{(k)^2}} = x \qquad \textit{and} \qquad \sqrt{\sum_{k=1}^{K}\mathrm{Soft}^2\left(A_{a,b}^{(k)}, \frac{\lambda_1}{\rho}\right)} = c.$$

where $x \geq 0$ (condition 1) and $c \geq 0$.

Then equation (4.4.2) becomes,

$$x\left(1+\frac{4\lambda_2}{\rho}x^2\right) - c = \frac{4\lambda_2}{\rho}x^3 + x - c = 0$$

The root of x was obtained using the Tartaglia-Cardano method, with

$$A = \frac{4\lambda_2}{\rho},\ B = 0,\ C = 1 \text{ and } D = -c,$$

as in (4.3.4) with

$$P = \frac{3AC - B^2}{3A^2} = \frac{\rho}{4\lambda_2} \qquad \text{and}$$

$$Q = \frac{2B^3 - 9ABC - 27A^2D}{27A^3} = -c\frac{\rho}{4\lambda_2}.$$

No obvious analytical expression to condition 1, *i.e.* $x \geq 0$, was found. This was tested numerically. Condition 2 was verified since $4P^3 + 27Q^2 > 0$ for $\lambda_2 > 0$ and $\rho > 0$.

Replacing $\left(\sum_k Z_{a,b}^{(k)^2}\right)$ with x in equation (4.4.1) leads to

$$Z_{a,b}^{(k)}\left(1+\frac{4\lambda_2}{\rho}x^2\right) = \mathrm{Soft}\left(A_{a,b}^{(k)}, \frac{\lambda_1}{\rho}\right)$$

$$\Leftrightarrow \quad Z_{a,b}^{(k)} = \mathrm{Soft}\left(A_{a,b}^{(k)}, \frac{\lambda_1}{\rho}\right)\left(1+\frac{4\lambda_2}{\rho}x^2\right)^{-1}.$$

4.5 Regularization profile of the proposals

Using a small toy dataset, the effects of each regularization method on the individual parameters in the precision matrices were investigated as illustrated in Figure 4.5.1. Of note, Proposal 2 was omitted from this figure as its curves were compressing the scales. Six parameters are illustrated in panels (a) to (f). In panel a1, larger ω_2 values increased the parameter value, in class 1, with the GGL penalty while no much effect was observed with the proposals. The same parameter, in class 2 (panel a2), was reduced with the GGL penalty while it was increased with the proposals, at larger ω_2 values. Depending on the parameters, the effects were different. However, in no scenario did we observe that larger ω_2 values flattened the parameters value to zero with the proposals contrary to the GGL penalty (*e.g.* panels e1 and f1). This confirms that the latter might encourage sparsity in some scenarios while our proposal do not. The figure also shows that ω_2 had a slightly stronger impact with Proposal 3 (green curves) compared to Proposal 1 as the change rate in the parameters value was steeper.
The same figure containing all the proposals and the GGL is provided in Figure 7.5.5 of the Appendix Section 7.5. Although erratic, the effects of Proposal 2 were much more pronounced compared to the other penalties and in some instance moved the parameters value on the opposite direction compared to Proposal 1 and 3 (*e.g.* panel d2).

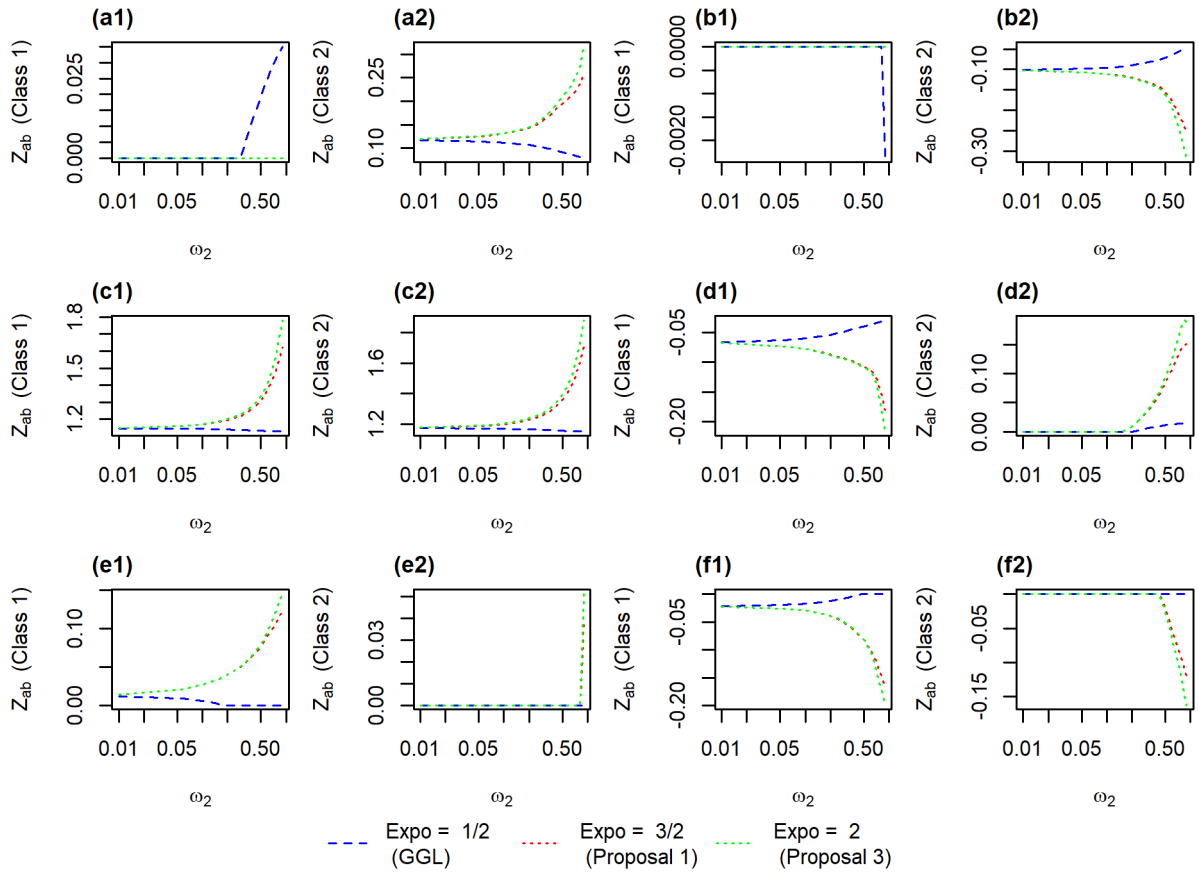

Figure 4.5.1: **Effect of ω_2 in the different penalties on the precision matrix parameters**. The precision matrices were of dimension (50×50) and sample size$= 50$. The proposals and the GGL penalty are plotted in different colors. Each letter in the legend indicate a parameter while the number indicate the graph class (also mentioned on the y-axis).

Chapter 5

Results

In this chapter, we apply and compare our proposals, on simulated data as well as real-world data sets, to the GGL procedure of Danaher *et al.* (2014) and the direct estimation of differential network in Zhao *et al.* (2014).

5.1 Performance of the regularization methods

The performance of regularization methods need to be objectively quantified with metrics. In this work, the notion of performance is mostly based on recovery of the graphical structures with some aspects of accuracy.

Zero-entries in precision matrices designate conditional independences in the graphical structures while non-zero elements indicate the existence of a direct relation or edge, *i.e.* the parameter $\hat{\theta}_{a,b} \neq 0$. Parameters of the precision matrices might vary across the classes K, *i.e.* differential with $\hat{\theta}^{(k)}_{a,b} \neq \hat{\theta}^{(k')}_{a,b}$. What constitutes a true/false edge, and a true/false differential edge needs to be clearly defined. In a simulation framework, these notions are fairly straightforward. An edge in a class is considered as follows

Edge		**Definition**
TP	True Positive	$\theta^{(k)}_{a,b} \neq 0$ and $\hat{\theta}^{(k)}_{a,b} \neq 0$
FP	False Positive	$\theta^{(k)}_{a,b} = 0$ and $\hat{\theta}^{(k)}_{a,b} \neq 0$
TN	True Negative	$\theta^{(k)}_{a,b} = 0$ and $\hat{\theta}^{(k)}_{a,b} = 0$
FN	False Negative	$\theta^{(k)}_{a,b} \neq 0$ and $\hat{\theta}^{(k)}_{a,b} = 0$

An edge is truly differential (TPDE) if $\{\theta^{(k)}_{a,b} \neq \theta^{(k')}_{a,b}$ and $\hat{\theta}^{(k)}_{a,b} \neq \hat{\theta}^{(k')}_{a,b}\}$ and falsely differential (FPDE) if $\{\theta^{(k)}_{a,b} = \theta^{(k')}_{a,b}$ and $\hat{\theta}^{(k)}_{a,b} \neq \hat{\theta}^{(k')}_{a,b}\}$. However, most methods do not yield the exact same parameters estimates across the classes even when they should. For those methods,

the definition of true differential edge is relaxed to accommodate for minor differences across matrices:

$$|\hat{\theta}_{a,b}^{(k)} - \hat{\theta}_{a,b}^{(k')}| > 10^{-2}.$$

The sensitivity and specificity, also referred to as respectively the true positive rate (TPR) and the true negative rate (TNR), are broadly used to estimate the recovery of the underlying graphical structures. These are defined as follows:

$$\text{sensitivity} = \frac{\text{TP}}{\text{TP} + \text{FN}} \quad \text{and}$$
$$\text{specificity} = \frac{\text{TN}}{\text{TN} + \text{FP}}$$

where FN and TN are the number of false negatives and true negatives, respectively. The sensitivity and specificity apply to the edges and differential edge recovery. Additionally, the false positive rate (FPR) was defined as:

$$\text{FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}} = 1 - \text{specificity}$$

also provides valuable information on the recovery. Another extensively used measure, as in Danaher *et al.* (2014), is the false discovery rate (FDR) defined as:

$$\text{FDR} = \frac{\text{FP}}{\text{FP} + \text{TP}}$$

which applies to the edges and differential edge recovery.

The TPE and FPE measures provide counts within a class as follows[1]

$$\text{TPE}^{(k)} = \sum_{1 \leq a < b \leq p} \mathbb{1}(\theta_{ab}^{(k)} \neq 0, \hat{\theta}_{ab}^{(k)} \neq 0),$$
$$\text{FPE}^{(k)} = \sum_{1 \leq a < b \leq p} \mathbb{1}(\theta_{ab}^{(k)} = 0, \hat{\theta}_{ab}^{(k)} \neq 0)$$

where $\mathbb{1}(\cdot)$ is the indicator function.
In the joint estimation methods, one needs to consider the selected edges and differential edge, whether true or false, across all classes. The simplest method is to sum-up these measures across the K classes: *i.e.* $\text{TPE}^{(\{\boldsymbol{\Theta}\})} = \sum_{k=1}^{K} \text{TPE}^{(k)}$. The average false positive and negative rate (TPR and FNR, respectively) defined in Guo *et al.* (2011) as follows:

$$\text{FPR} = \frac{1}{K} \sum_{k=1}^{K} \frac{\text{FP}^{(k)}}{\text{FP}^{(k)} + \text{TN}^{(k)}},$$
$$\text{FNR} = \frac{1}{K} \sum_{k=1}^{K} \frac{\text{FN}^{(k)}}{\text{FN}^{(k)} + \text{TP}^{(k)}}$$

[1] The same logic can be extended to the other measures, *i.e.* TNE and FNE.

provide an alternative metric[2] to evaluate the performance of methods. These rates are also applied to edges and DE recovery. Of note, these metrics correspond to the FPR described above and the FDR.

The accuracy of the parameters estimation was quantified, as described in Bühlmann and van de Geer (2011), by the Kullback-Leibler divergence (dKL) distance:

$$\text{dKL} = \sum_{k=1}^{K} -\log(\det \hat{\boldsymbol{\Theta}}^{(k)}\boldsymbol{\Sigma}^{(k)}) + \text{tr}\Big(\hat{\boldsymbol{\Theta}}^{(k)}\boldsymbol{\Sigma}^{(k)}\Big) - p$$

and the error of the estimation in terms of the Frobenius norm:

$$\sum_{k=1}^{K} ||\boldsymbol{\Theta}^{(k)} - \hat{\boldsymbol{\Theta}}^{(k)}||_F^2 = \sum_{k=1}^{K} \sum_{1\leq a<b\leq p} \Big(\hat{\theta}_{a,b}^{(k)} - \theta_{a,b}^{(k)}\Big)^2$$

which is referred to as the sum of squared errors (SSE) in Danaher *et al.* (2014). The L_1-norm of $\{\boldsymbol{\Theta}\}$ was used to measure the overall size of the off-diagonal parameters:

$$\sum_{k=1}^{K} \sum_{1\leq a<b\leq p} |\hat{\theta}_{a,b}^{(k)}|.$$

The selection of the optimal tuning parameters (λ_1, λ_2) combination can be achieved using the Akaike and/or the Bayesian information criteria (AIC and BIC, respectively), to compromise between the goodness-of-fit of the model and its complexity. These were approximated as follows:

$$AIC(\lambda_1, \lambda_2) = \sum_{k=1}^{K} \Big[n_k \,\text{tr}\big(\mathbf{S}^{(k)}\boldsymbol{\Theta}^{(k)}\big) - n_k \log(\det \hat{\boldsymbol{\Theta}}_{\lambda_1,\lambda_2}^{(k)}) + 2E_k\Big]$$

$$BIC(\lambda_1, \lambda_2) = \sum_{k=1}^{K} \Big[n_k \,\text{tr}\big(\mathbf{S}^{(k)}\boldsymbol{\Theta}^{(k)}\big) - n_k \log(\det \hat{\boldsymbol{\Theta}}_{\lambda_1,\lambda_2)}^{(k)} + \log(n_k)E_k\Big]$$

where $\hat{\boldsymbol{\Theta}}_{\lambda_1,\lambda_2}^{(k)}$ is the k^{th} precision matrix estimated considering the tuning parameters (λ_1, λ_2), and E_k is the number of selected edges in the lower (or upper) triangle of $\hat{\boldsymbol{\Theta}}^{(k)}$ excluding the diagonal elements.

Alternatively, the selection of the tuning parameters can be performed through cross-validation if the goal is to maximize the model predictive power, however this more computationally involve route was not explored here.

[2]For each of the metrics, *i.e.* FP, TN, FN and TN, only the lower (or upper) triangle of the matrices are considered.

5.2 Algorithmic implementation

The analytical solutions to all three proposals were implemented, along Danaher's GGL method, within the *mtJGL* R package[3] starting from the *JGL* R package developed by Danaher *et al.* (2018). The convex problems optimization was addressed using the ADMM algorithm described in Section 3.4. The pseudo-code is summarized in Algorithm 3.

The *mtJGL* package contains multiple functions among which

- the call function *mtJGL* which takes as parameters the list of data, the exponent q/r of the penalty $\tilde{h}_{q,r} = \lambda_2 \sum_{a \neq b} \left(\sum_k Z_{a,b}^{(k)^2} \right)^{q/r}$, the maximum number of iterations, the tuning parameters λ_1 and λ_2, and the step size ρ;
- the *admm.iters* function in which the ADMM algorithm is applied;
- the *dsgl* function in which the optimization of the scaled augmented Lagrangian (4.1.2) with regards to $\{\mathbf{Z}\}$ is performed.

The following critical adaptations had to be integrated into the codes to fulfil the conditions on the analytical solutions described in sections 4.2 to 4.4:

- an indicator matrix for Proposal 2, *i.e.* $\tilde{h}_{(2)} = \lambda_2 \sum_{a \neq b} \left(\sum_k Z_{a,b}^{(k)^2} \right)^{-1/2}$, was created to record adherence to the Tartaglia-Cardano condition, *i.e.* $4P^3 + 27Q^2 > 0$ and condition $\sqrt{\sum_k Z_{a,b}^{(k)^2}} > \sqrt[3]{\frac{\lambda_2}{\rho}}$ (see Section 4.3);
- for Proposal 3, another indicator matrix was created to record adherence to condition $\sqrt{\sum_k Z_{a,b}^{(k)^2}} \geq 0$.

In cases where the above conditions were not fulfilled, the parameters were not updated and kept unchanged until the next iteration.

Additionally, the matrices $\{\mathbf{S}\}$, $\{\mathbf{\Theta}\}$, $\{\mathbf{Z}\}$, $\{\mathbf{A}\}$ and $\{\mathbf{W}\}$ were rounded to the 9^{th} decimal at any step involving a matrix operation. This modification was motivated by (unexpected) rounding in R which, iteration after iteration, disbanded the symmetry of the matrices.

The output of the *mtJGL* R package was a list of optimal $\{\hat{\mathbf{\Theta}}\}$ matrices, along with their convergence status, from which the performance criteria were computed as described in Section 5.1.

[3] Available on the *DIAL* repository.

Algorithm 3: ADMM algorithm applied in the *mtJGL* R package

Initialize $\hat{\mathbf{\Theta}}^{(k)}_{(m=0)} = \text{diag}\big(\mathbf{S}^{(k)}\big), \hat{\mathbf{U}}^{(k)}_{(m=0)} = \mathbf{0}, \hat{\mathbf{Z}}^{(k)}_{(m=0)} = \mathbf{0}$ for $k = 1, \ldots, K$
Set $\rho = 1$, $\lambda_1 > 0$ and $\lambda_2 > 0$
while $m < 1000$ OR *diff* [a] $> 10^{-5}$ **do**
  for $k = 1, 2, \ldots, K$ **do**
    Update $\hat{\mathbf{\Theta}}^{(k)}_{(m)}$ as $\mathbf{V}^{(k)}_{(m-1)}\tilde{\mathbf{D}}^{(k)}_{(m-1)}(\mathbf{V}^{(k)}_{(m-1)})^T$;
  end for
  for $k = 1, 2, \ldots, K$ **do**
    Update $\mathbf{A}^{(k)}_{(m)}$ as $\mathbf{\Theta}^{(k)}_{(m)} + \mathbf{U}^{(k)}_{(m-1)}$;
  end for
  for $k = 1, 2, \ldots, K$ **do**
    Update $\mathbf{Z}^{(k)}_{(m)}$ as
    $\underset{\mathbf{Z}^{(k)}}{\arg\min}\ \frac{\rho}{2}\sum_{k=1}^{K}||\mathbf{Z}^{(k)}_{(m-1)} - \mathbf{A}^{(k)}_{(m-1)}||^2_F + \lambda_1||\mathbf{Z}_{(m-1)}||_1 + \tilde{h}_{q,r}(\{\mathbf{Z}_{(m-1)}\})$;
  end for
  for $k = 1, 2, \ldots, K$ **do**
    Update $\mathbf{U}^{(k)}_{(m)}$ as $\mathbf{U}^{(k)}_{(m-1)} + \mathbf{\Theta}^{(k)}_{(m)} - \mathbf{Z}^{(k)}_{(m)}$;
  end for
  Update convergence conditions
end while

[a] *diff* is the quantity $\sum_k \left(\frac{\hat{\mathbf{\Theta}}^{(k)}_{(m)} - \hat{\mathbf{\Theta}}^{(k)}_{(m-1)}}{\hat{\mathbf{\Theta}}^{(k)}_{(m-1)}} \right)$

5.3 Simulation study

We begin by comparing the ability of the regularization methods to recover the true graphical structures from synthetic data.
In section 5.3.1, we describe how the simulated data were build and provide a description of the resulting true graphical networks. The criteria measuring the performance of the regularization methods are defined in Section 5.1. The results of the comparison between the proposal, the GGL and direct estimation are presented in Section 5.3.2.

5.3.1 Simulation set-up

In the current work the number of groups was limited to two classes, *i.e.* $K = 2$, of equal size. Three sets of networks were build with $p \in \{50, 100, 500\}$ nodes forming, respectively, 2 ($p = 50$), 5 ($p = 100$) or 10 ($p = 500$) unconnected sub-networks of equal size. The degree of the nodes followed a power-law distribution (Newman, 2003), *i.e.* the probability of any node to connect p vertices is proportional to $1/p^{\alpha}$. The maximum degrees were 10 for $p \in \{50, 100\}$ or 20 for $p = 500$ and the parameter α was set to 2.3. These scale-free networks should emulate real-life biological networks constituted of hub nodes, *i.e.* vertices having a large degree (Chen *et al.*, 2004).
The density of edges in the graph was at around 4% ($p = 50$), 2% ($p = 100$) and 0.4% ($p = 500$).

The first $p \times p$ precision matrix, $\mathbf{\Theta}^{(1)}$, was generated as follows:

1. the positions of zero and non-zero parameters were based on the network structure described above;
2. the diagonal elements were set to value 1;
3. non-zero elements were sampled from a uniform distribution with support on $[-.4, -.1] \cup [.1, .4]$;
4. positive definiteness was ensured by dividing each row (excluding the diagonal element) with 1.5 times the sum of the absolute value of its off-diagonal elements ;
5. the matrix was averaged with its transpose to enforce symmetry.

The second $p \times p$ precision matrix, $\mathbf{\Theta}^{(2)}$, was globally equal to $\mathbf{\Theta}^{(1)}$ with modifications. The proportion of altered elements was set at 10% with a higher probability of reverting existing edges, *i.e* non-zero elements to zeros. Steps 3 to 5 were applied to obtain the second symmetric positive definite precision matrix $\mathbf{\Theta}^{(2)}$.

The 100 nodes power-law networks structure are illustrated in Figure 5.3.1. Panel (a) depicts the initial construct with its 5 unconnected sub-networks and hub nodes. Panel (b) shows the second class networks which bears 4 major sub-networks, 4 singletons and a

pair of connected nodes. Panel (c) overlays both structures and highlights the edges that are specific to each class.

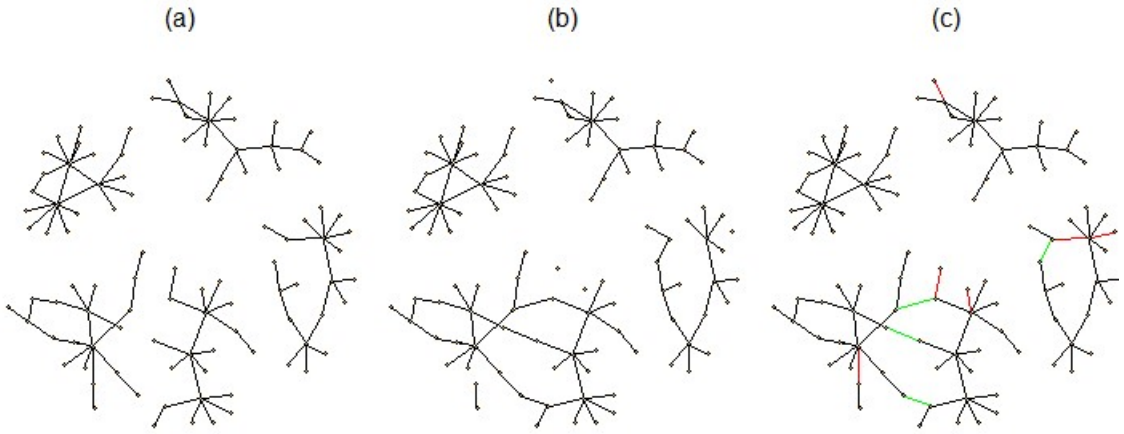

Figure 5.3.1: **Topology of the simulated networks**. Power-law networks built from 100 vertices. Each shape and segment indicate a node and an edge, respectively. The black segments indicate the common edges, the red and green ones illustrate those that are respectively unique to class 1 and class 2. (a) Class 1 is formed by 5 disjoint sub-graphs. (b) Class 2 is made of 4 large disjoint sub-graphs, one two-nodes sub-network and 4 unconnected nodes. (c) Overlay of class 1 and class 2.

The precision matrices encoding the 100-nodes networks are illustrated in Figure 5.3.2. Rows and columns are organised as to regroup the sub-structures. Panels depict $\mathbf{\Theta}^{(1)}$ (a) and $\mathbf{\Theta}^{(2)}$ (b) which contains 99 and 97 edges in total, respectively. Panel (c) shows the position of 10 altered parameters.

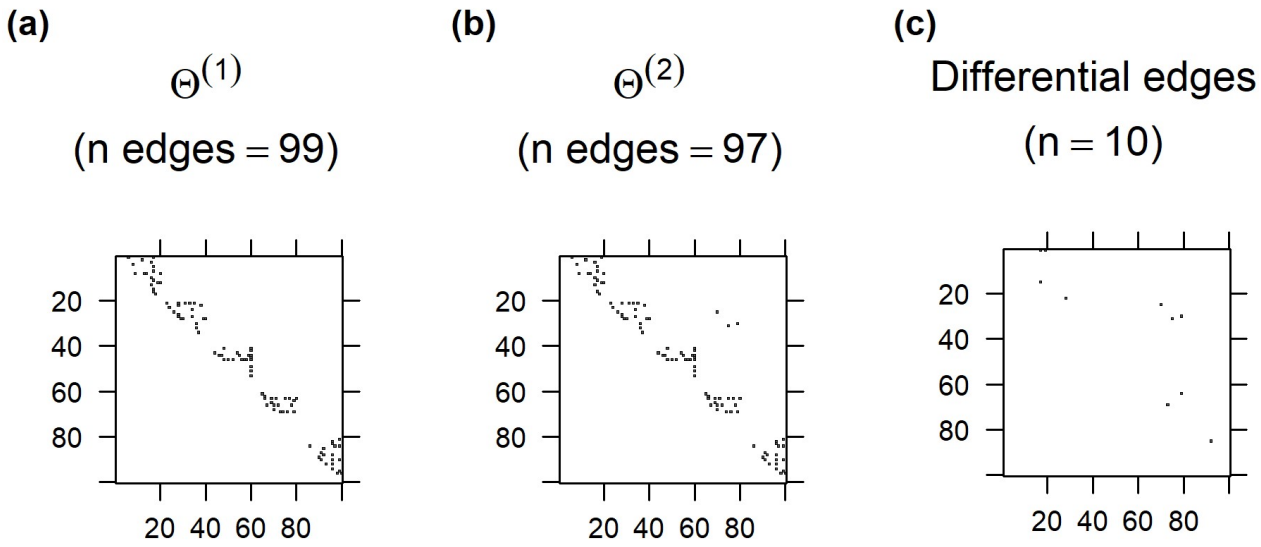

Figure 5.3.2: **Precision matrices encoding the synthetic networks**. Each matrix of dimension (100×100) is restricted to the upper off-diagonal elements. Each black square in the matrices indicates a non-zero parameter. The number of non-zeros, in each class, and their differential are provided above the panels. (a) Class 1. (b) Class 2. (c) Differential edges between class 1 and class 2.

The illustration of the 50- and the 500-nodes networks and their respective precision matrices are available in the Appendix (see figures 7.5.1, 7.5.2, 7.5.3 and 7.5.4 in Section 7.5).

The data matrices were sampled from a multivariate normal distribution with mean vector 0 and using the Cholesky factorised precision matrices.

The elements of the true $\mathbf{\Sigma}^{(k)}$ covariance matrices were build as follows:

$$\Sigma_{a,b}^{(k)} = \frac{\theta_{a,b}^{(k)}}{\sqrt{(\theta^{(k)})_{a,a}^{-1} (\theta^{(k)})_{b,b}^{-1}}}$$

where $\left(\theta^{(k)}\right)_{a,a}^{-1}$ and $\left(\theta^{(k)}\right)_{b,b}^{-1}$ are the diagonal elements of the inverted precision matrices.

5.3.2 Simulation results

We first sought to compare the performances of the proposals to the GGL penalty in Danaher *et al.* (2014). Varying tuning parameters were tested: 3 values of $\omega_2 \in \{0.01, 0.1, 0.3\}$ and 20 values of $\omega_1 \in [10^{-4}, 0.25]$. For each combination of the tuning parameters, $p \in \{50, 100, 500\}$ and $n \in \{50, 100, 500\}$ with $n \leq p$, 100 graphs were estimated using 100 different simulated datasets (replicates) generated as described in Section 5.3.1.

Figure 5.3.3 shows the median results over the 100 simulated datasets. The number of features and sample size were $p = 500$ and $n = 50$, respectively. The panels on the same row indicate the results obtained with a single value of the tuning parameter ω_2, as indicated on the top left corner. Of note, no missing results were responsible for the seemingly disrupted curves.

Panels (a1 - a3) of Figure 5.3.3 illustrate the sum of the dKL's as a function of the sparsity tuning parameter ω_1. In all cases the accuracy tended to improve (*i.e.* low dKL values indicate better performance) with larger values of the "sparsity" tuning parameter ω_1. At all values of ω_1 and ω_2, Proposal 1 and 3 were either worse, or comparable to the GGL penalty in terms of accuracy. The decrease of dKL values with Proposal 2 was slower than with the other penalties.
Overall, the accuracy results obtained for the other combinations of p and n were fairly similar, as shown in Figure 7.5.6 to 7.5.10 of the Appendix Section 7.5. A notable exception was with Proposal 2 (*i.e.* exponent "-1/2") at combination $\{p = 500, n = 500, \omega_2 = 0.3\}$. Figure 7.5.7, showed an initial dKL increase with larger values of ω_1 followed by a sharp decrease. This might be explained by an initial relatively accurate estimation of the parameters, due to the larger sample size. We have seen in the previous paragraph that, with Proposal 2, the profile of parameters values changes in response to ω_2 variations were somewhat erratic. One could expect it to hold when varying the ω_1 values until some threshold value where the sparsity drastically increases, resulting in a significant improvement in accuracy.
The tuning parameter ω_2 seemed to decrease the accuracy for the all three proposals with a stronger impact on Proposal 2.

Panels (b1 - b3) of Figure 5.3.3 illustrate the number of differential edges detected as a function of the total numbers of edges detected across both classes. Larger values of ω_1 decreased the number of edges and differential edges (DE) detected with all penalties. At lower ω_2 values, Proposal 1 (*i.e.* expo="3/2") and Proposal 3 (*i.e.* expo="2") were hardly distinguishable from the GGL penalty. At larger ω_2, *e.g.* 0.3, for a comparable number of edges detected, the number of differential edges was larger for all three proposals and in particular for Proposal 2 (*i.e.* expo="-1/2"). Interestingly, the initial decrease in the number of edges detected was accompanied by a slower or no reduction in the number of differential edges for all three proposals, in particular at large ω_2 values.
The (differential) edges detection results obtained for the other combinations of p and n were similar, as shown in Figures 7.5.6 to 7.5.10 of the Appendix Section 7.5.

Panels (c1-c3) of Figure 5.3.3 illustrate the RoC curves of edges detection. Larger values of ω_1 decreased the sensitivity and the false positive rate (FPR = 1−specificity). At lower ω_2 values, the proposals were fairly comparable, with Proposal 1 and Proposal 3 generating results that were hardly distinguishable from the GGL penalty. The performance of Proposal 2 compared to all other penalties was worse, especially at larger ω_2 values, but still better than a random flip. This observation might be explained by the larger number of edges detected which, however, were not true positives.

A sudden drop in sensitivity (with larger ω_1 values) was observed with Proposal 2 which seemed to coincide with the start of the differential edges reduction phase.
The RoC curves of edges detection obtained for the other combinations of p and n were similar, as shown in Figures 7.5.6 to 7.5.10 of the Appendix Section 7.5.

Panels (d1-d3) of Figure 5.3.3 illustrate the RoC curves of differential edges detection. Larger values of ω_1 decreased the sensitivity and the FPR. At lower ω_2 values, the proposals were hardly distinguishable from the GGL penalty. At larger ω_2 values, Proposal 1 and Proposal 3 were still comparable to the GGL penalty while Proposal 2 performed worse. This suggests that the larger number of differential edges detected with Proposal 2 were not accurate.
The RoC curves of differential edges detection obtained for the other combinations of p and n were similar, as shown in Figures 7.5.6 to 7.5.10 of the Appendix Section 7.5. The comparison of the proposals and GGL penalty to the direct estimation method of Zhao *et al.* (2014) is available for the datasets containing less than 500 features. The Kronecker product in the Zhao *et al.* (2014) algorithm required so much computer memory that it was not possible for us to obtain the estimates for large datasets. The results on the smaller datasets ($p \in \{100, 50\}$) indicate that Proposal 1, Proposal 2, Proposal 3 and the GGL penalty perform better than the direct estimator for any ω_2 and ω_1 values (Figures 7.5.8 to 7.5.10 of the Appendix Section 7.5). This comes as a surprise given that many more parameters need to be estimated with the joint estimate methods. The joint estimation borrows information from all classes and therefore lead to more accurate results, especially since the vast majority of parameters are expected (in this case, known) to be identical across classes.

A different perspective on these results is provided in Figures 7.5.11 to 7.5.16 of the Appendix Section 7.5. These illustrate the effect of ω_2 focusing on each regularisation method separately. Contrary to the GGL penalty, larger ω_2 values decreased the estimations accuracy (column panels a1-d1). The results indicate a major non-convergence issue (red rug lines) with Proposal 2 which was also observed with the other methods at the lower ω_1 values. This observation likely contributes to all the unexpected and erratic behaviour of Proposal 2. These convergence issues might be related to the analytical constrains on the optimisation problem. More specifically, the algorithmic bypass around the unmet conditions might have either slowed down or even caused the failure of the optimisation algorithm.
Unlike our proposals, larger ω_2 values decreased the proportion of differential edges detected with the GGL penalty (column panels a1-d1). The impact, if any, of ω_2 on the sensitivity and FPR of (differential) edges detection was mild, except with Proposal 2.

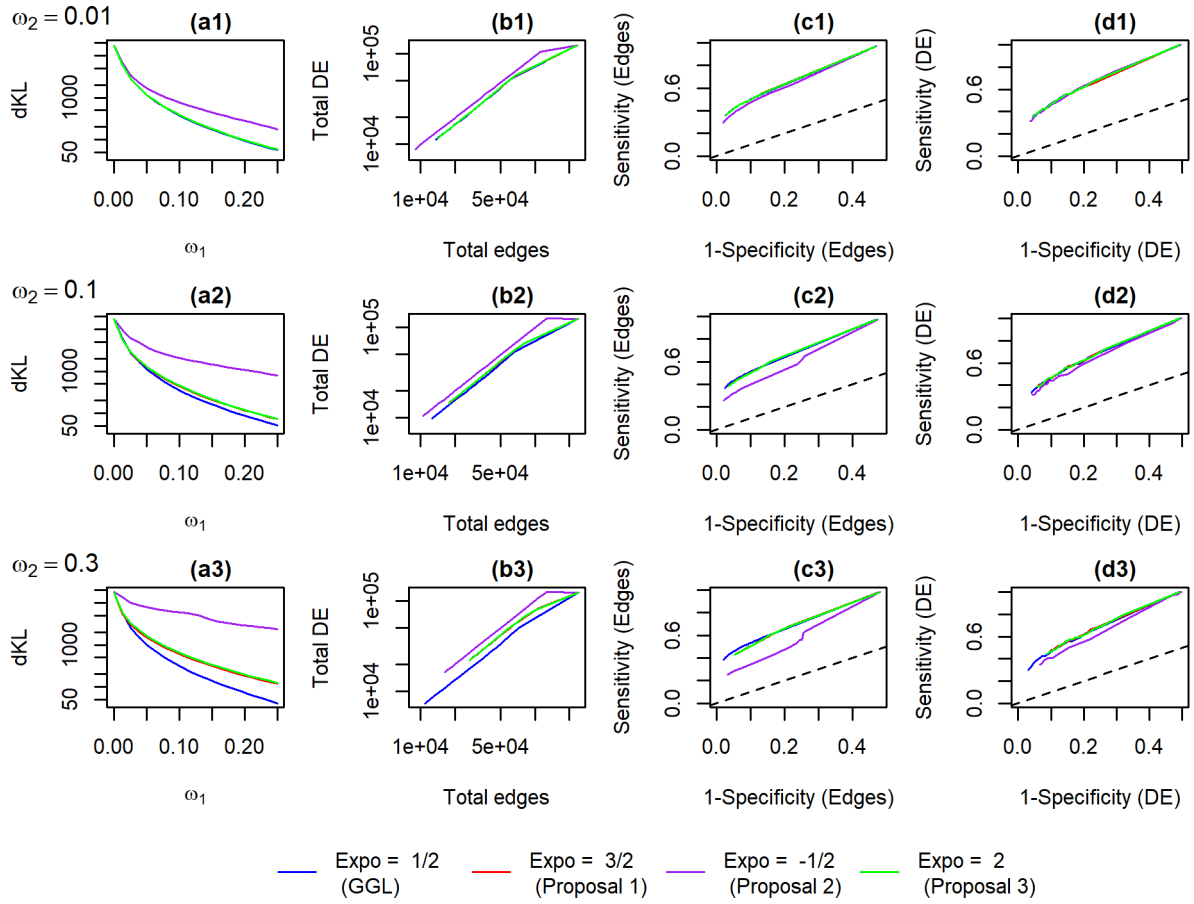

Figure 5.3.3: **Performance of the proposals for precision matrices of dimension** (500×500) **and sample size**$= 50$. The curves are medians over 100 replicates. Results generated with each value of ω_2 are displayed on the rows. In all panels, the curves were generated with varying tuning parameter ω_1 values. The proposals and the GGL penalty are plotted in different colors. The Kullback-Leibler divergence, of the estimated networks from the true ones, are plotted against ω_1 values in panels (a1-a3). The total number of edges and differential edges (DE) detection are plotted in panels (b1-b3). The RoC curves of the edges and differential edges (DE) are plotted in panels (c1-c3) and (d1-d3) respectively. The dashed lines represent the perfect chance.

The impact of the number of features and sample size on the proposals performance was evaluated using the same results discussed in the above paragraphs. For each proposal and within each run/replicate, the optimal couple $\{\omega_1, \omega_2\}$ was identified on the basis of the AIC or the BIC criterion. The median, and the median absolute deviation of the 100 BIC- or AIC-optimal replicates were then computed and listed, respectively, in Table 5.3.1 and Table 7.6.1 of the Appendix Section 7.6. Both tables indicated the same trends.

For all 3 proposals and the GGL penalty, the accuracy was reduced with larger numbers of features and smaller sample sizes. As the dKL value also reflects the matrices dimension,

the effect of the number of features should be interpreted with caution. The impact of the number of variables and sample size on the sensitivity to detect edges and differential edges was not obvious as it moved up and down, however this was consistent across Proposal 1, Proposal 2 and Proposal 3 and the GGL penalty. The FPR (edges and differential edges detection) decreased with larger sample sizes, for all penalties, and surprisingly seemed to increase with less features in the matrices.

Overall, the estimation of graph structures, using synthetic datasets, with the different regularization methods showed that our proposals did not perform better than the GGL penalty. In some conditions our proposals lead to higher proportion of differential edges, which might have suggested an improvement in preserving dissimilarities between classes, however this did not improve the sensitivity and false positive rate. Also, Proposal 1 (exponent "3/2") and Proposal 3 (exponent "2") displayed similar behaviour, if not identical. This suggests that the geometric constraints they exert on the log-likelihood are very similar.
A second learning piece concern the issues of non-convergence in particular with Proposal 2 which generated reliable estimates in some cases cases only. It might be that the convergence issue could be fixed or at least lessened by increasing the number of iterations in the ADMM algorithm. In any case, this shows the impact of imposing more constrains on the search for solutions.
At last, our work emphasized the limitation of the direct estimator from the computing point of view as the memory resources required to estimate large matrices are truly egregious.

				Edges		DE	
Penalty Exponent	p	n	dKL (MAD)	Sensitivity (MAD)	FPR (MAD)	Sensitivity (MAD)	FPR (MAD)
"1/2" (GGL)	50	50	2 (0)	.340 (.044)	.024 (.003)	.333 (.247)	.039 (.005)
	100	50	4 (0)	.378 (.023)	.023 (.001)	.400 (.148)	.035 (.002)
		100	3 (0)	.352 (.023)	.002 (.000)	.300 (.148)	.006 (.001)
	500	50	41 (1)	.384 (.013)	.021 (.000)	.296 (.055)	.030 (.000)
		100	20 (0)	.377 (.010)	.002 (.000)	.278 (.055)	.004 (.000)
		500	8 (0)	.593 (.010)	.001 (.003)	.556 (.027)	.002 (.000)
"3/2" (Proposal 1)	50	50	2 (0)	.330 (.044)	.033 (.003)	.333 (.247)	.063 (.005)
	100	50	5 (0)	.367 (.030)	.033 (.001)	.400 (.148)	.058 (.002)
		100	3 (0)	.347 (.023)	.006 (.001)	.352 (.148)	.012 (.001)
	500	50	58 (1)	.359 (.012)	.028 (.000)	.352 (.055)	.047 (.000)
		100	21 (0)	.370 (.012)	.006 (.000)	.352 (.055)	.009 (.000)
		500	9 (0)	.554 (.009)	.001 (.000)	.556 (.027)	.002 (.000)
"-1/2" (Proposal 2)	50	50	15 (3)	.330 (.044)	.043 (.009)	.333 (.247)	.095 (.016)
	100	50	62 (1)	.347 (.023)	.046 (.001)	.500 (.148)	.095 (.002)
		100	13 (2)	.413 (.023)	.016 (.004)	.500 (.148)	.036 (.008)
	500	50	2405 (9)	.251 (.010)	.032 (.000)	.352 (.055)	.065 (.000)
		100	713 (4)	.378 (.009)	.027 (.000)	.444 (.055)	.054 (.000)
		500	37 (0)	.634 (.007)	.009 (.000)	.667 (.055)	.020 (.000)
"2" (Proposal 3)	50	50	2 (0)	.330 (.044)	.033 (.003)	.333 (.247)	.063 (.005)
	100	50	5 (0)	.367 (.030)	.033 (.001)	.400 (.148)	.046 (.002)
		100	3 (0)	.347 (.023)	.006 (.001)	.400 (.148)	.012 (.001)
	500	50	58 (1)	.359 (.012)	.028 (.000)	.352 (.055)	.046 (.000)
		100	21 (0)	.370 (.012)	.006 (.000)	.352 (.055)	.009 (.000)
		500	9 (0)	.554 (.009)	.001 (.000)	.556 (.027)	.002 (.000)

Table 5.3.1: **Performance of the proposals as a function of the number of features and sample size**. The optimal combination of tuning parameters, based on the BIC, were selected from each run/replicate. The results are medians over 100 replicates. The Kullback-Leibler divergence (dKL), the sensitivity and the false positive rate (FPR=1-specificity) of edges and differential edges detection are provided.

5.4 Real Data

The proposed penalties were tested on a real dataset publicly available at the Gene Expression Omnibus[4] repository with the accession number GDS2771. GEO DataSets[5] are curated sets of biologically and statistically comparable samples, *i.e.* measurements are comparable in terms of processing and normalization across samples.

The gene expression was measured by DNA Microarray, a technology in which slides are spotted with thousands of specific DNA sequences (also called probes) at defined positions. mRNA extracted from samples are retro-transcribed into complementary DNA (cDNA), labelled with a fluorescent dye and then hybridized onto the slide probes. The fluorescent signal measured from the probes is considered proportional to the expression level of their mapped gene.

The GDS2771 dataset contains 22,283 Affymetrix HG-U133A microarray probes detecting mRNA in large airway epithelial lung cells collected from 192 smokers (Spira *et al.*, 2007): 90 controls (class 1), 97 with lung cancer (class 2) and a smaller panel of patients with suspected cancer. The latter was omitted from the analyses.
The original study aimed at finding lung cancer predictive biomarkers using a gene expression pattern strategy. The authors identified an 80-gene set that showed 80% sensitivity and 84% specificity in lung cancer prediction on smokers with suspected cancer. However their findings did not identify the affected cellular pathways. The panel of biomarkers can be exploited to investigate cellular networks but this would be labour intensive and costly. For example, the above dataset was the starting point for a later study that revealed the role of the phosphatidylinositol 3-kinase (PI3K) pathway in the onset of lung cancer development (Gustafson *et al.*, 2010).
The joint graphical model approach is therefore particularly useful to generate working hypotheses to unravel the underlying mechanisms responsible for healthy cells transition into malignant cells.

Given the large number of variables, the limited number of observations, *i.e.* $p \ggg n$, and the limited time and resources at our disposal to estimate the entire precision matrices, the number of features included in the analyses had to be limited. The selection of genes is not trivial and might impact the quality of the estimated network. Indeed, an important assumption in structure learning is that there must be no unobserved variables. Another important assumption in structure learning is that there must be a one-to-one correspondence between the nodes of the graph and the random variables included in the model. An inherent challenge with the microarray data is the presence of probes that might detect the same gene. And to add complexity to it, although some probes map to the same gene they might actually detect alternatively spliced mRNA which are translated

[4] `http://www.ncbi.nlm.nih.gov/geo`
[5] GDSxxx where letters "xxx" stands for the dataset id.

into different proteins. Therefore results generated from a fraction of the network nodes and possibly replicated vertices must be interpreted with caution.

The filtering was therefore restricted to excluding:

- control and house keeping genes, mostly used for signal normalization;
- probes with negligible data dispersion. The median absolute deviation (MAD) was considered as it is a robust estimation of the variability. The microarray-derived values were $\log_2$-transformed prior to computing the MAD of the variables. These were then divided by their respective probe median values as to avoid biasing the selection toward the highest expressed transcripts. The top 1000 probes with the highest variation were kept.

Some of the 180 differential genes mentioned in Danaher *et al.* (2014) were manually added, into our dataset, as these were excluded at the filtering steps. This additional step led to the selection of 1290 features, since some of the 180 added genes are detected by multiple probes. Of note, in Danaher *et al.* (2014) 17720 probes were included in their dataset.

The microarray-derived signals were $\log_2$-transformed and normalized, *i.e.* within each class the variables mean was set to zero and their standard deviation equal to one. As in Danaher *et al.* (2014), the classes were weighted equally to prevent the larger class (*i.e.* Class 2 - cancer patients) from prevailing in the networks estimation. The convergence criteria were set to the maximal accepted number of 1000 iterations and a relative difference between successive estimates $\leq 10^{-5}$. A high granularity screen was performed to identify the optimal combination of tuning parameters λ_1 and λ_2 : respectively 20 values within $[0.1; 1]$ and 10 values within $[0.005; 0.9]$. For each combination (λ_1, λ_2), the AIC and BIC were recorded to select the optimal graphical model.

The resulting AIC and BIC of all tuning parameters combinations are shown in Figure 7.5.17 of the Appendix Section 7.5. Except for exponent "-1/2", the AIC profiles were roughly S-shaped while the BIC profiles were roughly bell-shaped. Large λ_2 values lead to higher AIC and BIC across the λ_1's with the GGL penalty and, to a lesser extend, with Proposal 1 and 3. The AIC and BIC profiles of exponent "-1/2" were similar and very low at lower λ_2 values. Of note, in some instances the AIC and BIC curves seemed flat however the scale was so compressed that even large variations cannot be spotted on the figure, *e.g.* the AIC of penalty "1/2" at large λ_2.
The red rugs on the figures indicate that the ADMM algorithm did not converge. This was particularly striking for exponent "-1/2" where most tested combinations of tuning parameters lead to no optimal solution. The GGL penalty was less impacted by the convergence issues at larger λ_2 values where the AIC and BIC criteria were the highest. It is expected that the precision matrices, estimated using the GGL regularization method, become sparser throughout the ADMM iterations and therefore the difference between successive estimates would necessarily be dropping. Exponents "3/2" and "2" displayed

similar convergence profiles with more success at λ_1 values between 0.1 and 0.3 where the AIC and BIC values were the lowest.

The results of the tuning parameters selection are summarized in Table 5.4.1 which lists the tuning parameters combination leading to the lowest AIC and BIC, and the convergence status of the ADMM algorithm. The GGL penalty, Proposal 1 and 3 reached their lowest AIC and BIC at the lowest tested λ_2 value of 0.005. The optimal λ_1 values were in the lower range of the tested values for all penalties. Slightly larger λ_1 values were picked up with the BIC criterion for the GGL penalty, Proposal 1 and 3. Of note, our λ_1 selection greatly contrasts from Danaher *et al.* (2014) not only because the dataset[6] differs but also the authors considered the interpretability, *i.e* a large λ_1 value was used to detect a manageable number of edges, of the results which is not the focus of this master thesis. Since the selection based on the BIC criterion led to the same or to sparser matrices, it was considered for the final choice of the tuning parameters pair.
Table 5.4.1 also provides the number of discovered edges and altered edges across the classes. Altered edges are defined as those that are detected in one class but not in the others plus the edges encoded by parameters of opposite sign in the different classes' precision matrix. The results indicated that the vast majority of discovered edges were classified as altered edges. This behaviour was observed with many (λ_1, λ_2) pairs as illustrated in Figure 7.5.18 of the Appendix Section 7.5. The proportion of altered edges was comparable between Proposal 1 and 3, *i.e.* respectively 77.2% and 77.4%, slightly lower with the GGL (*i.e.* 74.4%) and very high with Proposal 2. Considering the BIC criterion the GGL penalty led to the largest number discovered edges, *i.e* with 55177 non-zero parameters while Proposal 1, Proposal 2 and 3 resulted in the comparable number of discovered edges. Of note the number of detected edges and altered edges can not be directly compared between penalties unless the tuning parameters are identical which is the case for the GGL, Proposal 1 and 3. Our results suggest that these proposals generated sparser matrices and proportionally more dissimilarities.

The Venn diagram illustrated in Figure 5.4.1 shows the number of altered edges that are common or unique to the regularization methods. A large proportion of altered edges were common to the GGL, Proposal 1 and Proposal 3. Proposal 1 and 3 shared the vast majority (36534) of their altered edges. More than 60% of the altered edges in Proposal 2 were unique to it.
Of note, the total number of altered edges printed in the Venn diagram, for all three penalties, did not amount to the numbers reported in Table 5.4.1. The GGL penalty lacked 44 edges, Proposal 2 was 25 edges short while Proposal 1 and Proposal 3 missed 7 each. This apparent discrepancy was due to some edges encoded with opposite sign in the precision matrices of the classes.

[6]The precision matrices contained 17720 features and were estimated using the FGL regularization method. The selected λ_1 value was 0.95.

Exponent (Proposal)	Criterion	λ_1	λ_2	Conv	Edges	Altered edges
"1/2" (GGL)	AIC	0.100	0.005	TRUE	71875	55436
	BIC	0.147			55177	41083
"3/2" (Proposal 1)	AIC	0.100	0.005	TRUE	75365	62210
	BIC	0.195			47591	36768
"-1/2" (Proposal 2)	AIC	0.100	0.05	TRUE	44537	43894
	BIC					
"2" (Proposal 3)	AIC	0.100	0.005	TRUE	74835	61869
	BIC	0.195			47338	36635

Table 5.4.1: **Selection of the tuning parameters λ_1 and λ_2.** The selected tuning parameters pairs were based on the lowest AIC and BIC criteria. The numbers of discovered edges and altered edges are provided. The final selection of λ_1 and λ_2 are highlighted in gray. Conv denotes whether the ADMM algorithm converged or not.

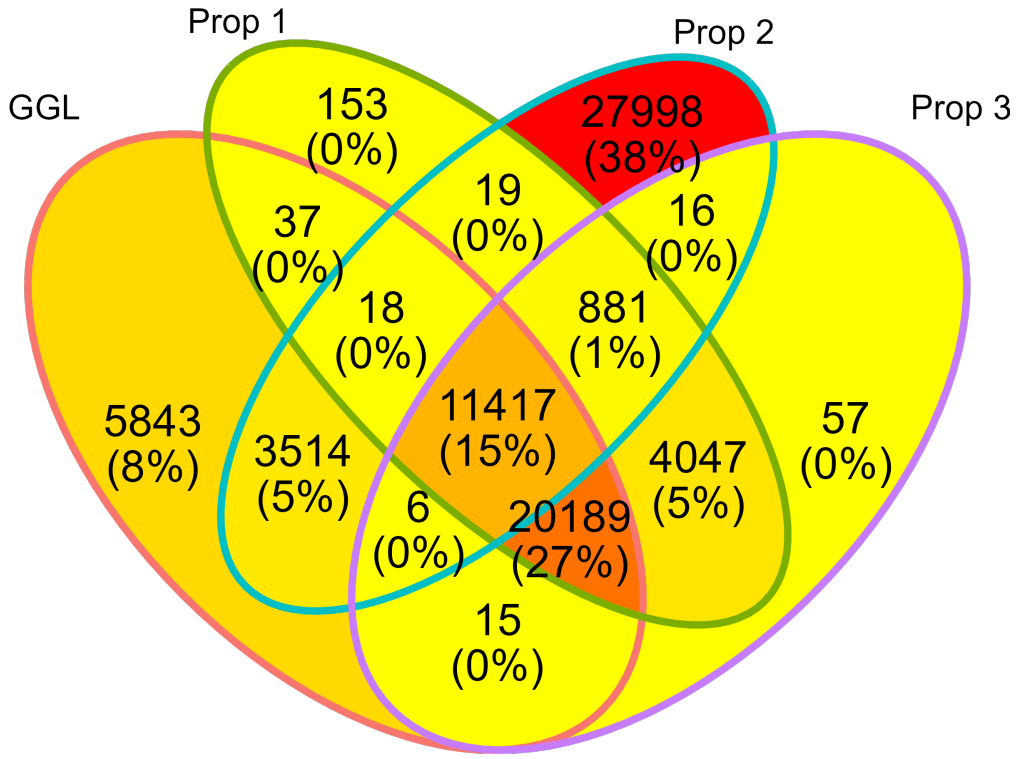

Figure 5.4.1: **Common and unique altered edges in graphs estimated with the GGL, Proposals 1, 2 and 3.** The number of shared or unique altered edges are printed in their respective zone. The reported percentages were computed against the grand total of altered edges across all penalties. The colors are indicative of the percentage levels. Prop 1: Proposal 1, Prop 2: Proposal 2, Prop 3: Proposal 3.

The sign inversion of the parameters across classes is not captured in the drawn networks but might still indicate an important change. The relevance of this observation has to be considered case by case as one can imagine that low parameters (borderline to the zeroing threshold) could randomly fluctuate around the value zero. More interestingly, these changes could reflect the inversion of relations in the graphs. The list of the sign-altered genes is provided in Table 7.6.2 of the Appendix Section 7.6. Some of the genes in the list (*e.g.* CLK, MPI,..) are involved in several sign-altered edges. The same sign altered edges were detected with Proposal 1 and 3 and six out seven of those were also identified with the GGL method. All, except for one, of the 25 sign altered edges detected with Proposal 2 were unique to it.

The density of the graphs is reported in Table 5.4.2. The edges density was much higher in the GGL penalty graphs while the proposals were more comparable. The density in Class 2 (*i.e.* cancer group) appeared higher, compared to class 1, for all penalties. The increased density appeared more important for Proposal 2 and the GGL regularization methods. Of note, a change of 1% roughly represents an alteration of over 8000 edges. This observation might suggest that, compared to the healthy cells, either some new cellular networks are activated or that some networks cross-reach in cancer cells (Csermely *et al.*, 2016).
The maximum degree was much higher in the Proposal 2 graphs; Proposal 1 and 3 were comparable and slightly lower than the GGL. The maximum degree of the cancer group (Class 2) graphs tended to be higher with all regularization methods. This observation might indicate that new gene relations are established rather new networks being activated.

Exponent (Proposal)	**Class**	**Edges**	**Density (%)**	**Degree**
"1/2" (GGL)	1	26898	3.2	81
	2	28279	3.4	99
"3/2" (Proposal 1)	1	23405	2.8	69
	2	24186	2.9	83
"-1/2" (Proposal 2)	1	21120	2.5	134
	2	23417	2.8	136
"2" (Proposal 3)	1	23279	2.8	68
	2	24059	2.9	82

Table 5.4.2: **Features of the estimated graphs**. The number of detected edges, the edges density and the maximal node degree in the graph are provided for each class and estimation method.

Given the large number of discovered and altered edges, it was not possible to illustrate the networks in their entirety[7]. Some sets of nodes were selected to illustrate few examples of sub-networks. For example, selecting the adjacent nodes to one gene, *e.g* STAT6 an

[7]The list of discovered edges in each class using all regularization methods can be provided on demand.

important effector of tumorigenesis (Delgado-Ramirez *et al.*, 2020), which is detected with one probe in our data set, led to a GGL estimated network of 41 vertices and 128 edges. The STAT6 adjacent nodes networks are illustrated in Figure 7.5.19 of the Appendix Section 7.5. The same nodes were selected in the proposals graphs and their respective position was identical in all networks (panels a-d) to facilitate the reading. Even with such limited networks, it was difficult to visually spot differences although the graph of Proposal 2 clearly contained less edges, *i.e.* 70.
The size of the networks in the next examples was thus limited as to facilitate visualization. Three examples are shown in Figure 5.4.2.

The sub-networks of the probes detecting Tubulin or Tubulin-related genes (*i.e.* TUBA1B, TUBB3, TUBB4B, TUBB, TUBA1A, TUBA1C, TUBA3D-TUBA3C and LOC101927673-OTUB1) are illustrated in panels a, d, g and j. These 8 genes were represented by 16 probes in our dataset. Tubulins form the cell cytoskeleton, play a variety of functions and are thought to be involved in cancer chemotherapy resistance. Since these genes are likely involved in the same process, it was not surprising to detect 62, 64, 56 and 64 edges in the sub-networks estimated with respectively the GGL (a), Proposal 1 (d), Proposal 2 (g) and Proposal 3 (j) regularisation methods. Most relations appeared unchanged when comparing the GGL to Proposal 1 and 3. The profile of identified altered edges was similar across these three sub-networks with Proposal 1 seemingly being identical to Proposal 3. Of note, the node gene LOC101927673-OTUB1 (uncharacterised at the time of the study) was not connected to any other probe.

A sub-network of 20 nodes was generated by randomly selecting 10 edges from the GGL graph of class 1. The nodes were then extracted, from the graphs, along with their edges. A total of 27, 20, 17 and 20 edges were detected in the sub-networks estimated with respectively the GGL (b), Proposal 1 (e), Proposal 2 (h) and Proposal 3 (k) regularisation methods. One could have expected, given how the sub-networks were built, some level of disconnection but it might be that the density of the graphs wouldn't allow it. As in the first example, Proposal 1 and 3 were identical while Proposal 2 and the GGL differed by several edges.

A second sub-network of 20 nodes was generated by randomly selecting 20 vertices from the GGL graph of class 1. A total of 8, 7, 9 and 7 edges were detected in these sub-networks estimated with respectively the GGL (c), Proposal 1 (f), Proposal 2 (i) and Proposal 3 (l) regularisation methods. As one would have expected, randomly selecting nodes in a graph leads to a sparse sub-network. As in the first and second examples, Proposal 1 and 3 were identical while Proposal 2 and the GGL differed by several edges.

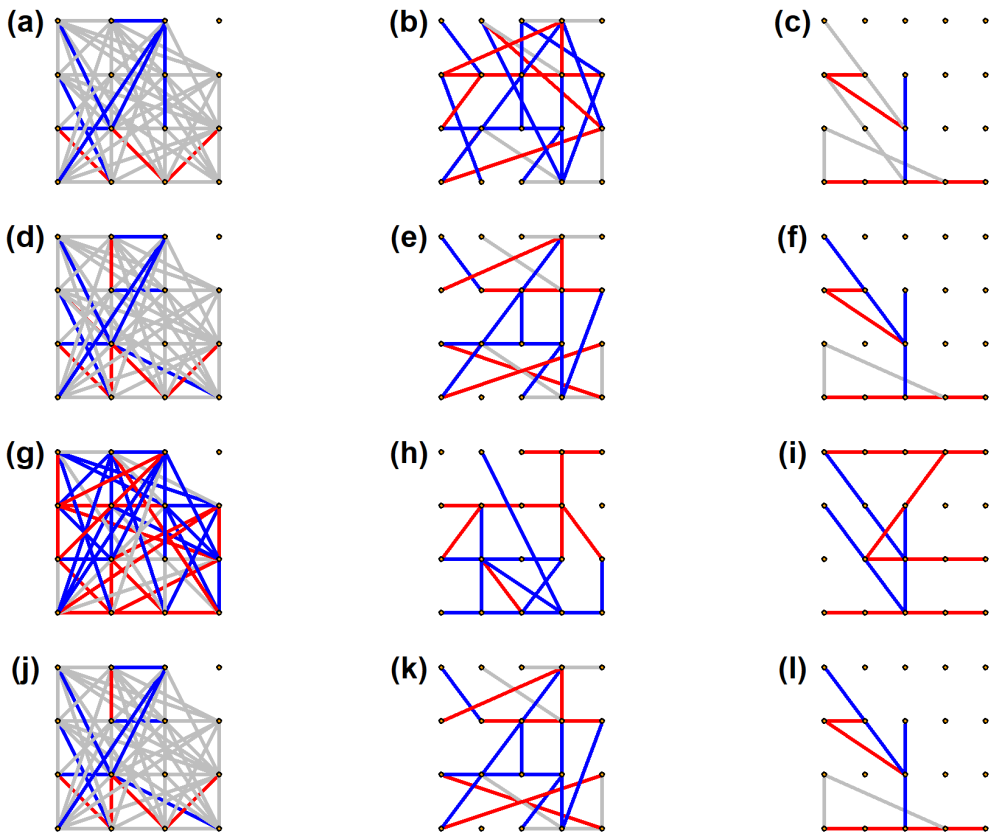

Figure 5.4.2: **Subnetworks extracted from the graphs**. Panels (a, d, g and j), (b, e, h and k) and (c, f, i and l) are the sub-networks of respectively the Tubulin probes, the randomly selected edges and the randomly selected nodes. Panels (a, b and c), (d, e and f), (g, h and i) and (j, k and l) are the sub-networks from graphs estimated with the GGL, Proposal 1, Proposal 2 and Proposal 3 regularisation methods. All nodes are identical and positioned identically across each sub-network (column panels). The blue and red lines illustrate the edges that are found only in class 1 and 2, respectively, while the gray ones are those common to both classes.

Overall, the estimation of the graph structure, using a real-life dataset, with different regularization methods taught us that the computation power needed requires resources and skills that are not readily available. We therefore could not estimate the entire graphs. A second learning piece concerned the selection of optimal tuning parameters. Depending on the objectives, one might have to weight the pro's and con's of selecting a more or less large sparsity parameter to obtain either interpretable or complete networks. The advantage of the former is obvious while the latter might be of interest when the scientific question needs to be addressed in a more holistic way.
A major issue of convergence was observed across all four methods and in particular with Proposal 2. It might be that the convergence issue could be fixed or at least lessened by

increasing the number of iterations in the ADMM algorithm. The analytical solutions to our proposed regularisation methods were conditioned to satisfy extra requirements. It is most likely that these conditions bear some weight on the convergence speed.
The results showed that our proposed regularisation methods led to sparser estimates, at the optimal tuning parameter values, than the GGL approach. Nevertheless, the vast majority of altered edges were commonly detected by the GGL, Proposal 1 and Proposal 3 methods. The proposed methods appeared to proportionally detect more altered edges which was our targeted goal, if these were true. Finally, all the tested regularization methods resulted in very low similarity between classes as most detected edges were deemed altered edges. This observation is to be put in perspective with the poor sensitivity and specificity of these methods.

Chapter 6

Discussion & Conclusion

In this master thesis we set out to jointly estimate groups of sparse Gaussian graphical models with the main goal of preserving both the similarities and the differences between classes. Our tasks were to come up with new regularization methods, implement them algorithmically and evaluate their performance on synthetic data. Our proposals were then tested on a real-data set to illustrate their advantages or drawbacks.

Besides their utility in the networks structure estimation, it is worth noting that the multi-class graphical estimators are useful in generating covariance matrix estimates used in quadratic discriminant analysis and linear discriminant analysis (Simon *et al.*, 2011).

The grouped graphical *lasso* method developed in Guo *et al.* (2011) also aimed at preserving the common structure while allowing for differences between groups. However their penalty, and therefore their optimization problem, was not convex. Since the optimization of non-convex problems can lead to sub-optimal solutions, other regularization methods needed to be investigated.
The proposals of this work are reminiscent of the multi-class $\ell_{1,s}$-norm penalties family introduced in Honorario *et al.* (2015). Their regularization method employs a single penalty function to control both the sparsity and the conditional independence pattern in the precision matrices. The interesting multi-class regularization methods in Danaher *et al.* (2015) introduced a second penalty to control the pattern in the matrices besides the ℓ_1 penalty to tune the sparsity. Their methods were aimed at encouraging similarity across classes. In particular, the GGL penalty suffered from a major drawback as it also contributed to the sparsity in the matrices. Hence, similarity would be achieved through forcing on sparsity. Therefore additional methods to preserve the conditional independence pattern in the precision matrices are needed.

The choice of penalty was restricted by the complexity necessary to develop the analytical solution to the optimization problem. Proposal 2 and Proposal 3 entailed solving third degree polynomial equations which was feasible with the Tartaglia-Cardano method but at the cost of additional conditions on the solutions. These, if not fulfilled, might either slow

down the optimization process or worse lead to sub-optimal solutions. Neither of these possibilities were formally investigated. Other penalties could be proposed, however, the complexity and the extra conditions might unbalance their gains, if any.

The theoretical properties of our proposed regularization methods, including consistency of the estimators, were not examined. At this stage of the project and this work, it would not be have been of much interest to gain that knowledge. However, in future works it might become of interest to compare some estimators displaying similar performances in term of network recovery.

The computational cost aspect of the proposals was overlooked at this stage of the project. However, our experience with large data sets has highlighted the importance of working with methods having fast convergence speed and with efficient algorithms. For instance, in Danaher *et al.* (2014) it is shown that block diagonal optimization problems can be solved way faster if the tuning parameters are large enough. We did not implement that aspect in our algorithm as we had no knowledge whether it holds true for our proposals.

Using the synthetic data, we showed that Proposal 1 and Proposal 3 were mostly undistinguishable and both are comparable or slightly worse than the GGL penalty in terms of performances (*i.e.* accuracy, RoC curves of the edges and differential edges detection). It is worth noting that the detection of (differential) edges relies on some criteria, *i.e* the parameters and their between-class differences were thresholded. These cut-off values were based on the literature, however it was not clear how different values would affect the results of the analyses and the conclusions. The unlikely scenario that would lead to an alternative outcome would be if one class was globally more affected by the thresholding.

We focused our simulation studies on the family of scale-free networks. These are thought to mimic the structure of biological networks (Chen *et al.*, 2004) and are generally harder to estimate than simpler uniform structures (Peng *et al.*, 2009). The performance of our proposals has not been compared on other type of structured networks (*e.g. Nearest-neighbour network*). The performance of the regularization method in Guo *et al.* (2011) was shown to vary between underlying network structures. Whether the performance of our methods would be sensitive to the underlying network is unclear and might be of interest to generalize the conclusions.

The similarity between the groups was shown to affect performances of the multi-class regularization methods (Guo *et al.*, 2011). The true underlying networks in our simulations studies were designed with roughly 10% of the total edges differing between classes which reported to the entire networks means that the largest networks (*i.e.* $p = 500$) displayed more between-classes similarity than the smallest one (*i.e.* $p = 50$). Therefore our results on the impact of the number of features on the performances should be interpreted with caution.

The proposed regularization methods were aimed at jointly estimating multiple classes although we provided only 2-class datasets examples. One could expect better performance with more than 2 classes, particularly if these share a high degree of similarity. This is because the multi-class estimators borrow information from all classes. However this need to be verified experimentally.

Our results on the real-world cancer data set showed that Proposal 1 and Proposal 3 were similar in terms of the numbers of edges and altered edges detected. Their entire networks could not be pictured, due to their sizes, but the visualized sub-networks were identical and the vast majority of their altered edges were common. Although many differences with the GGL-estimated networks were observed, the large majority of altered edges were common to all three networks. This result was somewhat reassuring as these regularisation methods are related, and was in line with the simulations results.
A careful interpretation of our results, from the biological point of view, needs to be considered as we only included a fraction of the actual nodes in the estimated networks. Obviously, the selection of the nodes can greatly influence the outcome.

The normal distribution of the data is a strong assumption in Gaussian graphical models. Violation of this premise can lead to poor performance of the regularization methods (Dokuzoglu *et al.*, 2017). The data in our real dataset do not strictly adhere to the normality assumption. In fact, the distribution of microarray data is a matter of debate (Wit *et al.*, 2004). More robust methods could instead be applied to estimate Gaussian graphical models (Castelo *et al.*, 2006).

The case of cancer data sets is most likely very complex. Indeed cancerous cells can undergo many changes at the gene expression level but also chromosomal alteration can lead to large differences compared to healthy cells. Our results indicated that the degree of the networks was increased in the cancer group. The literature also mentioned that the genes involved in cancer were more connected (Jonsson *et al.*, 2006). Working on "cancer networks" might therefore be particularly challenging given that the level of similarity between the classed might impact the networks estimates.

In this master thesis, we have introduced a new family of penalties to estimate Gaussian graphical models. We have studied the performance of 4 members; one of them being the GGL penalty published in Danaher *et al.* (2014) and another 3 proposals. We have shown that Proposal 1 and Proposal 3 displayed comparable performances and both were relatively similar to the GGL penalty. Proposal 2 was less reliable in terms of algorithmic convergence and showed poor performance. Interestingly, Proposal 1 and Proposal 3 did not contribute to sparsity unlike the GGL penalty.
During the course of this work, while testing the regularization methods on a large real dataset, we came across some computational challenges despite our access to the high performance computing network *CECI*. This limitation could be a major hurdle when one wants to generate working hypotheses based on the estimated networks.

Chapter 7

Appendix

7.1 Log-likelihood

Considering the centred data matrix $\mathbf{X} \in \mathbb{R}^{p \times n}$ following a multivariate Gaussian distribution, the likelihood is as follows

$$L(\mathbf{X}|\mu, \mathbf{\Sigma}) = \prod_{i=1}^{n} (2\pi)^{-\frac{p}{2}} (\det \mathbf{\Sigma})^{-\frac{1}{2}} \exp\left\{-\frac{1}{2}\mathbf{X}^T \mathbf{\Sigma}^{-1} \mathbf{X}\right\}.$$

Taking the logarithm of the likelihood, we obtain

$$\ell_n(\mathbf{X}|\mu, \mathbf{\Sigma}) = -\frac{n}{2}\log(\det \mathbf{\Sigma}) - \frac{1}{2}\sum^{n} \mathbf{X}^T \mathbf{\Sigma}^{-1} \mathbf{X} + \textit{constant}$$

. Using the "trace trick", the expression rewrites to

$$\begin{aligned}
\ell_n(\mathbf{X}|\mu, \mathbf{\Sigma}) &= -\frac{n}{2}\log(\det \mathbf{\Sigma}) - \frac{1}{2}\sum^{n} \operatorname{tr}\big(\mathbf{X}^T \mathbf{\Sigma}^{-1} \mathbf{X}\big) + \textit{constant} \\
&\propto -\frac{n}{2}\log(\det \mathbf{\Sigma}) - \frac{1}{2}\sum^{n} \operatorname{tr}\big(\mathbf{X}\mathbf{X}^T \mathbf{\Sigma}^{-1}\big) \\
&= -\frac{n}{2}\log(\det \mathbf{\Sigma}) - \frac{1}{2}\operatorname{tr}\left(\sum^{n} \mathbf{X}\mathbf{X}^T \mathbf{\Sigma}^{-1}\right) \\
&= -\frac{n}{2}\log(\det \mathbf{\Sigma}) - \frac{n}{2}\operatorname{tr}\big(\mathbf{S}\mathbf{\Sigma}^{-1}\big) \qquad = \frac{n}{2}\log\big(\det \mathbf{\Sigma}^{-1}\big) - \frac{n}{2}\operatorname{tr}\big(\mathbf{S}\mathbf{\Sigma}^{-1}\big)
\end{aligned}$$

7.2 Schur complement

Let matrix $\mathbf{\Sigma} = \begin{pmatrix} \mathbf{\Sigma}_{AA} & \mathbf{\Sigma}_{AB} \\ \mathbf{\Sigma}_{BA} & \mathbf{\Sigma}_{BB} \end{pmatrix}$ on which a block Gaussian elimination is performed to obtain

$$\begin{pmatrix} \mathbf{\Sigma}_{AA} & \mathbf{\Sigma}_{AB} \\ \mathbf{\Sigma}_{BA} & \mathbf{\Sigma}_{BB} \end{pmatrix} \begin{pmatrix} \mathbf{I}_q & -\mathbf{\Sigma}_{AA}^{-1}\mathbf{\Sigma}_{AB} \\ \mathbf{0}_{q+r} & \mathbf{I}_r \end{pmatrix} = \begin{pmatrix} \mathbf{\Sigma}_{AA} & \mathbf{0}_{q+r} \\ \mathbf{\Sigma}_{BA} & \mathbf{\Sigma}_{BB} - \mathbf{\Sigma}_{BA}\mathbf{\Sigma}_{AA}^{-1}\mathbf{\Sigma}_{AB} \end{pmatrix}$$

where the lower right term is the Schur complement, with dimension $q \times q$, of the block $\mathbf{\Sigma}_{AA}$. The transformation assumes that the block $\mathbf{\Sigma}_{AA}$ is invertible.

7.3 Cholesky factorization

Positive definite matrices can uniquely written as

$$\mathbf{\Sigma} = \mathbf{L}\mathbf{D}\mathbf{L}^T$$

where $\mathbf{L}$ is unit lower triangular and $\mathbf{D}$ is diagonal (Higham, 2009). The Cholesky factor, $R = \mathbf{D}^{1/2}\mathbf{L}^T$, can be obtained by solving the following equations

$$\begin{cases} \Sigma_{a,a} = \sum_{k=1}^{a} R_{k,a}^2 & \text{when } a = b \\ \\ \Sigma_{a,b} = \sum_{k=1}^{a} R_{k,a} R_{k,b} & \text{when } b > a. \end{cases}$$

7.4 Inverse variance Lemma

$$\begin{aligned} \mathbf{\Sigma}^{-1} &= \left[\begin{pmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{\Sigma}_{BA}\mathbf{\Sigma}_{AA}^{-1} & \mathbf{I} \end{pmatrix} \begin{pmatrix} \mathbf{\Sigma}_{AA} & \mathbf{0} \\ \mathbf{0} & \mathbf{\Sigma}_{A|B} \end{pmatrix} \begin{pmatrix} \mathbf{I} & \mathbf{\Sigma}_{AA}^{-1}\mathbf{\Sigma}_{AB} \\ \mathbf{0} & \mathbf{I} \end{pmatrix} \right]^{-1} \\ &= \begin{pmatrix} \mathbf{I} & \mathbf{\Sigma}_{AA}^{-1}\mathbf{\Sigma}_{AB} \\ \mathbf{0} & \mathbf{I} \end{pmatrix}^{-1} \begin{pmatrix} \mathbf{\Sigma}_{AA} & \mathbf{0} \\ \mathbf{0} & \mathbf{\Sigma}_{A|B} \end{pmatrix}^{-1} \begin{pmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{\Sigma}_{BA}\mathbf{\Sigma}_{AA}^{-1} & \mathbf{I} \end{pmatrix}^{-1} \\ &= \begin{pmatrix} \mathbf{I} & -\mathbf{\Sigma}_{AB}^{-1}\mathbf{\Sigma}_{AA} \\ \mathbf{0} & \mathbf{I} \end{pmatrix} \begin{pmatrix} \mathbf{\Sigma}_{AA}^{-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{\Sigma}_{A|B}^{-1} \end{pmatrix} \begin{pmatrix} \mathbf{I} & \mathbf{0} \\ -\mathbf{\Sigma}_{AA}\mathbf{\Sigma}_{BA}^{-1} & \mathbf{I} \end{pmatrix} \\ &= \begin{pmatrix} \mathbf{\Sigma}_{AA}^{-1} + \mathbf{B}^{\mathbf{T}}\mathbf{\Sigma}_{A|B}^{-1}\mathbf{B} & -\mathbf{B}^{\mathbf{T}}\mathbf{\Sigma}_{A|B}^{-1} \\ \mathbf{\Sigma}_{A|B}^{-1}\mathbf{B} & \mathbf{\Sigma}_{A|B}^{-1} \end{pmatrix} \end{aligned}$$

where $\mathbf{B} = \mathbf{\Sigma}_{BA}\mathbf{\Sigma}_{AA}^{-1}$. Demonstration adapted from Whittaker, 1990.

7.5 Supplementary figures

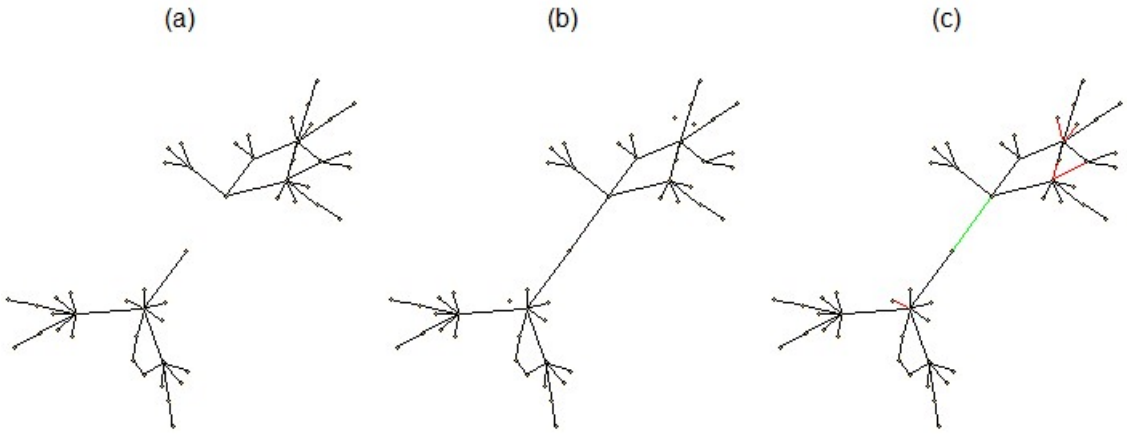

Figure 7.5.1: **Topology of the simulated 50-nodes networks**. Power-law networks built from 50 vertices. Each shape and segment indicate a node and an edge, respectively. The black segments indicate the common edges, the red and green ones illustrate those that are respectively unique to class 1 and class 2. (a) Class 1 formed by 2 disjoint sub-graphs. (b) Class 2 . (c) Overlay of class 1 and class 2.

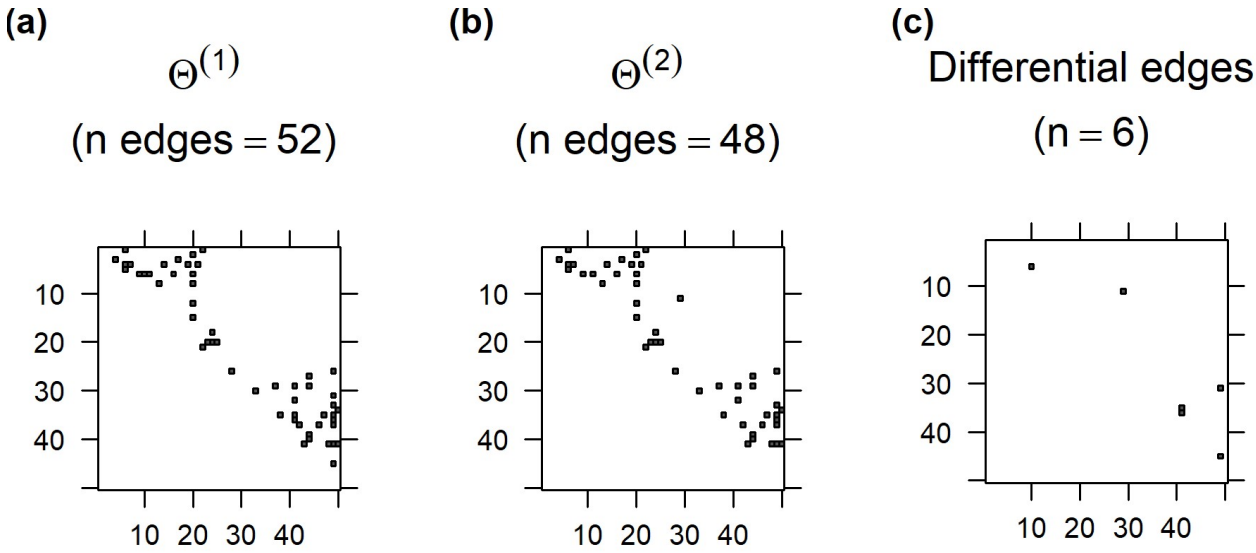

Figure 7.5.2: **Precision matrices encoding the synthetic 50-nodes networks**. Each matrix of dimension (50×50) is restricted to the upper off-diagonal elements. Each black square in the matrices indicate a non-zero parameter. The number of non-zeros, in each class, and their differential are provided above the panels. (a) Class 1. (b) Class 2. (c) Differential between class 1 and class 2.

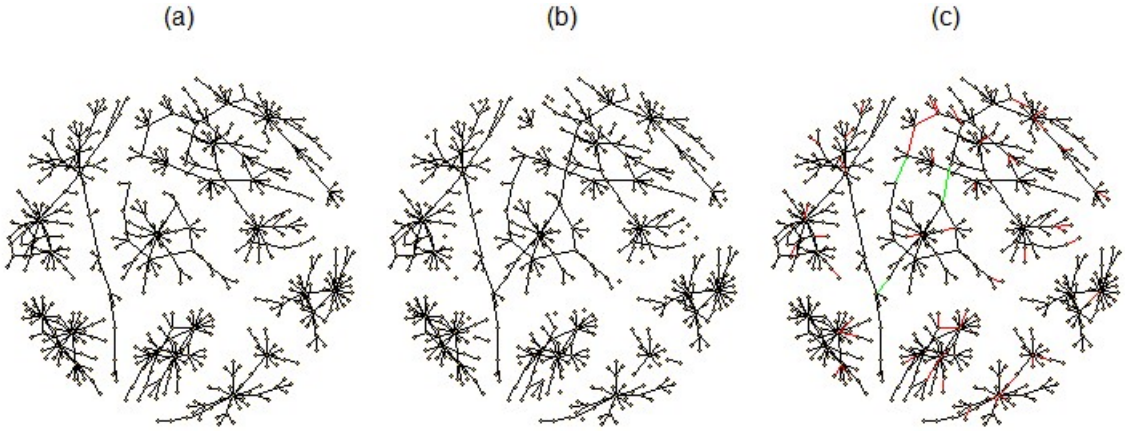

Figure 7.5.3: **Topology of the simulated 500-nodes networks**. Power-law networks built from 500 vertices. Each shape and segment indicate a node and an edge, respectively. The black segments indicate the common edges, the red and green ones illustrate those that are respectively unique to class 1 and class 2. (a) Class 1 formed by 10 disjoint sub-graphs. (b) Class 2 . (c) Overlay of class 1 and class 2.

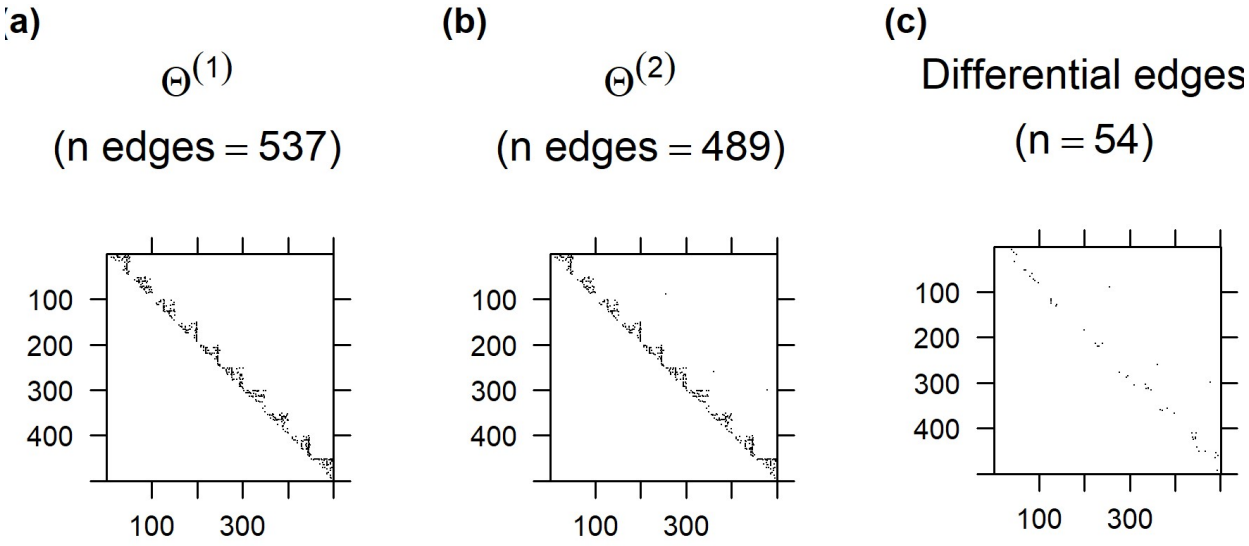

Figure 7.5.4: **Precision matrices encoding the synthetic 500-nodes networks**. Each matrix of dimension (500×500) is restricted to the upper off-diagonal elements. Each black square in the matrices indicate a non-zero parameter. The number of non-zeros, in each class, and their differential are provided above the panels. (a) Class 1. (b) Class 2. (c) Differential between class 1 and class 2.

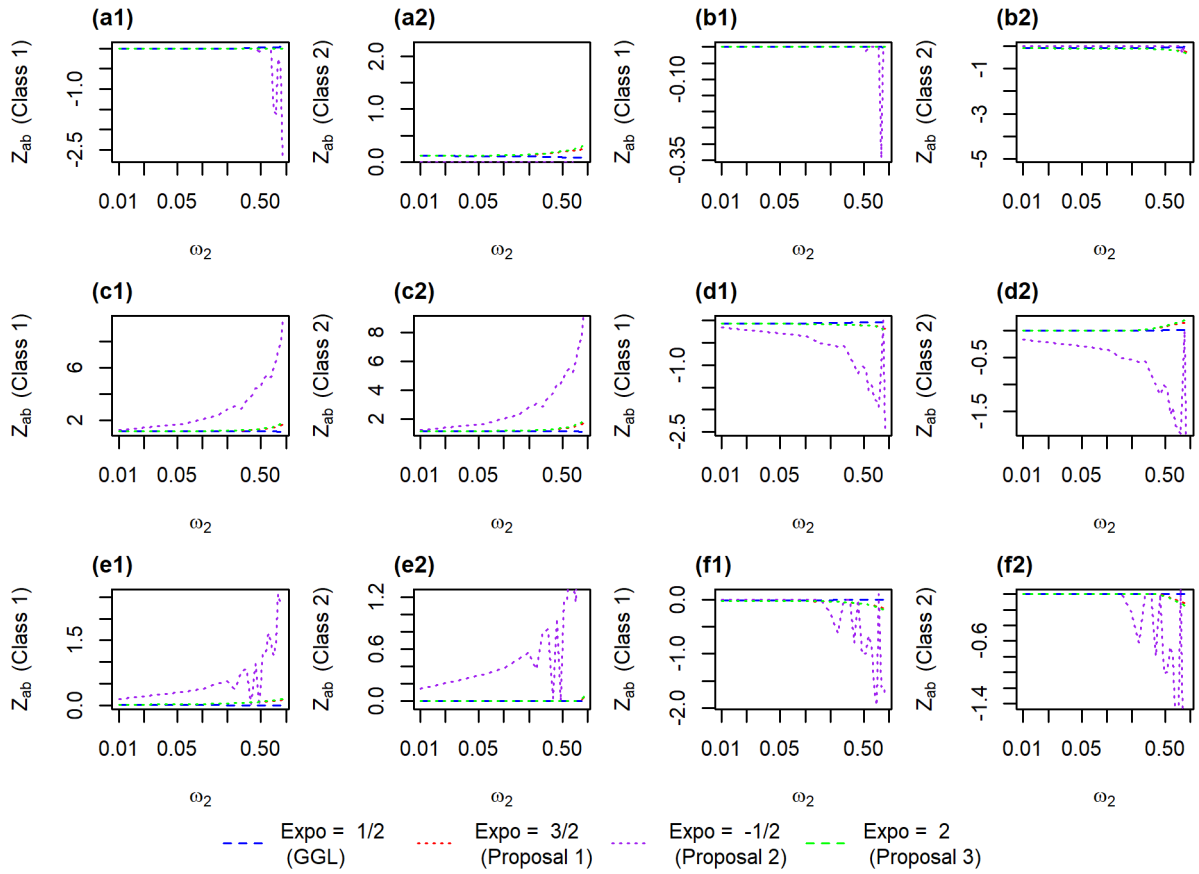

Figure 7.5.5: **Effect of ω_2 in the different penalties on the precision matrix parameters**. The precision matrices were of dimension (50×50) and sample size$= 50$. The proposals and the GGL penalty are plotted in different colors. Each letter in the legend indicate a scenario while the number indicate the class (also mentioned on the y-axis).

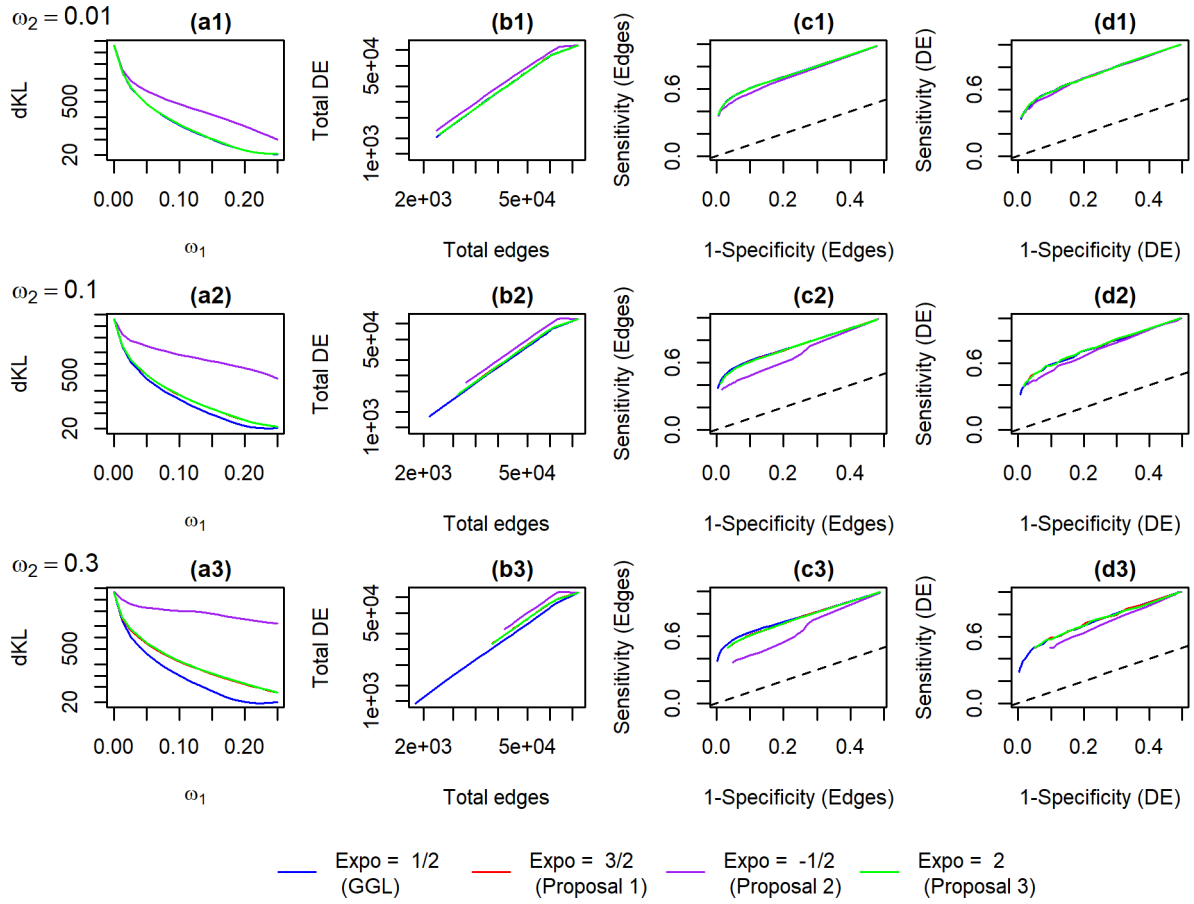

Figure 7.5.6: **Performance of the proposals for precision matrices of dimension** (500×500) **and sample size**$= 100$. Results generated with each value of ω_2 are displayed by rows. In all panels, the curves are generated with a varying tuning parameter ω_1. The proposals and the GGL in Danaher *et al.* (2014) are plotted in different colors. The Kullback-Leibler divergence, of the estimated networks from the true ones, are plotted against ω_1 values in panels (a1-a3). The total number of edges and differential edges (DE) detection are plotted in panels (b1-b3). The RoC curves of the edges and differential edges (DE) are plotted in panels (c1-c3) and (d1-d3) respectively. The dashed lines represents the perfect chance.

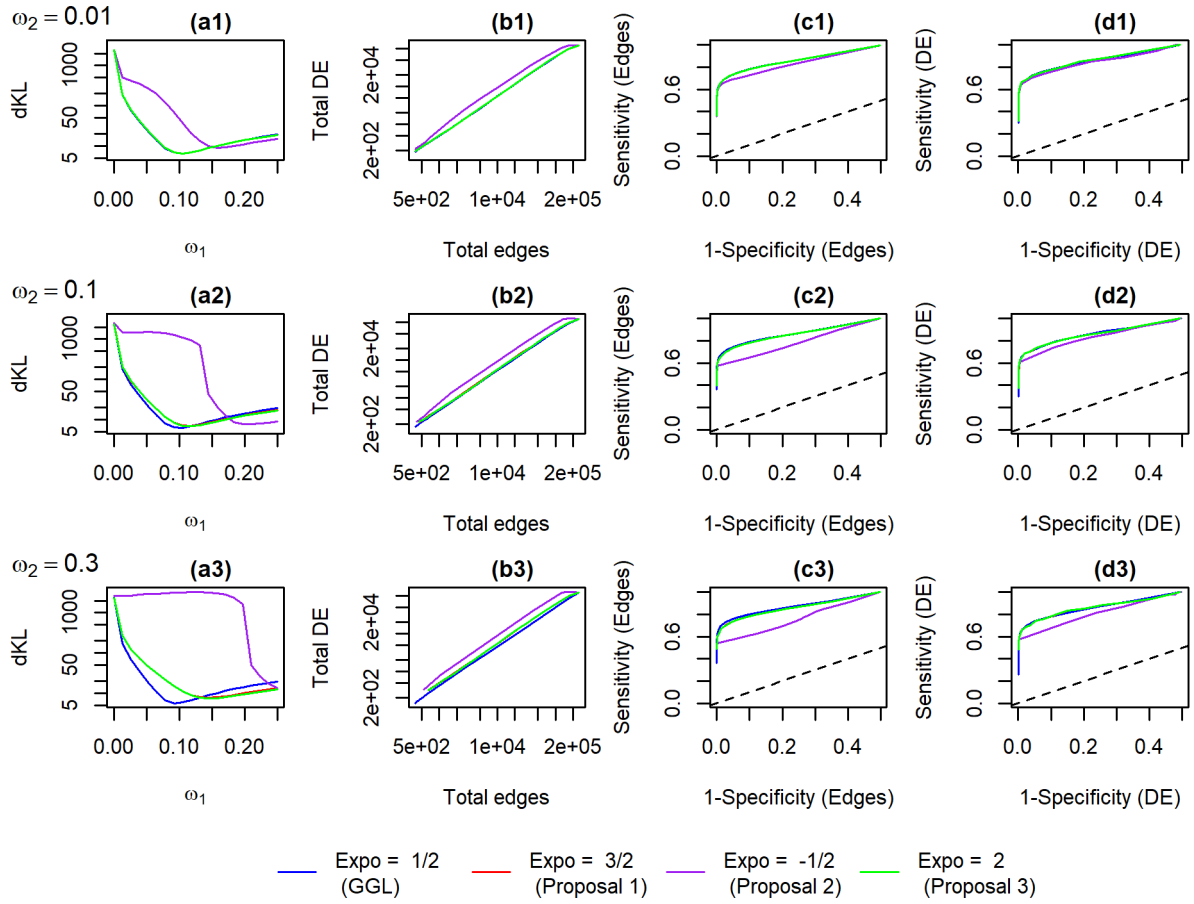

Figure 7.5.7: **Performance of the proposals for precision matrices of dimension** (500×500) **and sample size**$= 500$. Results generated with each value of ω_2 are displayed by rows. In all panels, the curves are generated with a varying tuning parameter ω_1. The proposals and the GGL in Danaher *et al.* (2014) are plotted in different colors. The Kullback-Leibler divergence, of the estimated networks from the true ones, are plotted against ω_1 values in panels (a1-a3). The total number of edges and differential edges (DE) detection are plotted in panels (b1-b3). The RoC curves of the edges and differential edges (DE) are plotted in panels (c1-c3) and (d1-d3) respectively. The dashed lines represents the perfect chance.

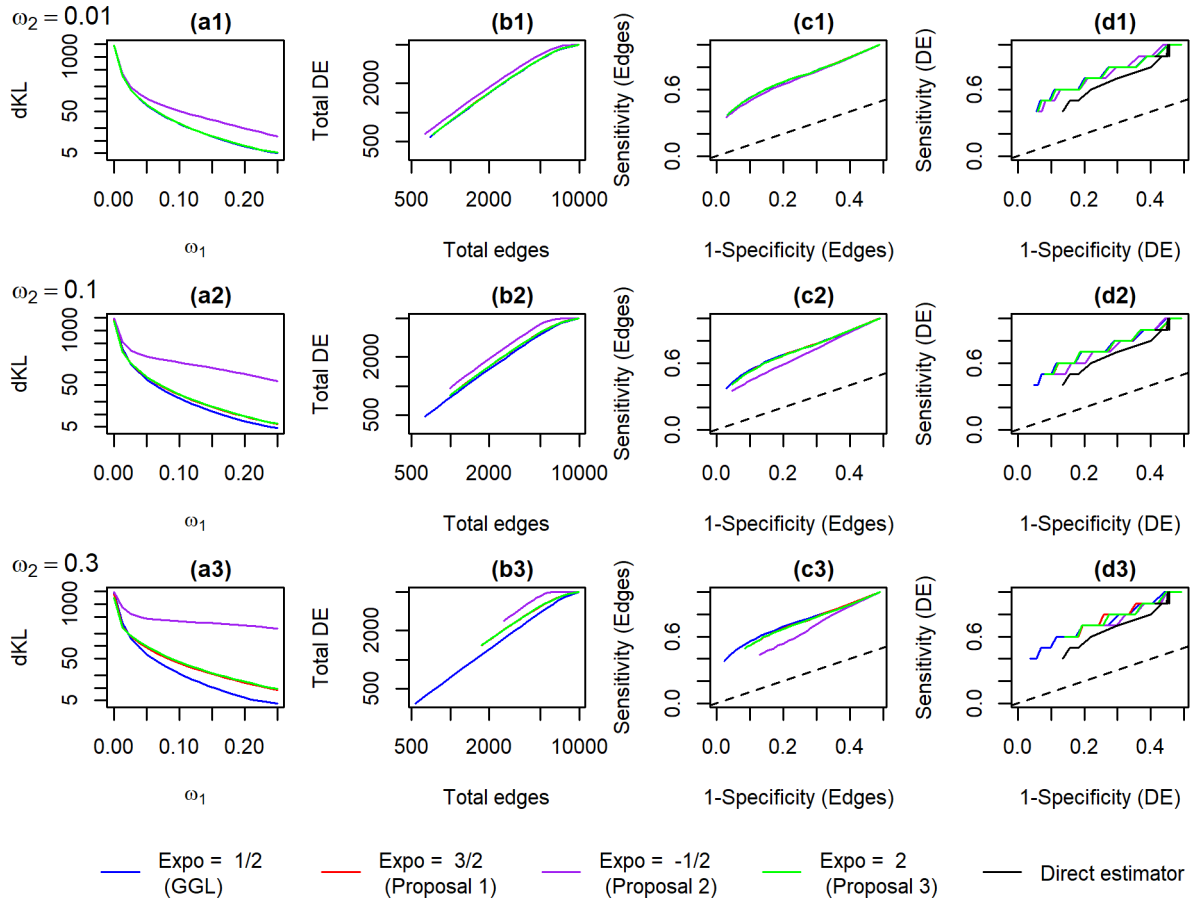

Figure 7.5.8: **Performance of the proposals for precision matrices of dimension** (100×100) **and sample size**$= 50$. Results generated with each value of ω_2 are displayed by rows. In all panels, the curves are generated with a varying tuning parameter ω_1. The proposals and the GGL in Danaher *et al.* (2014) are plotted in different colors. The Kullback-Leibler divergence, of the estimated networks from the true ones, are plotted against ω_1 values in panels (a1-a3). The total number of edges and differential edges (DE) detection are plotted in panels (b1-b3). The RoC curves of the edges and differential edges (DE) are plotted in panels (c1-c3) and (d1-d3) respectively. The dashed lines represents the perfect chance.

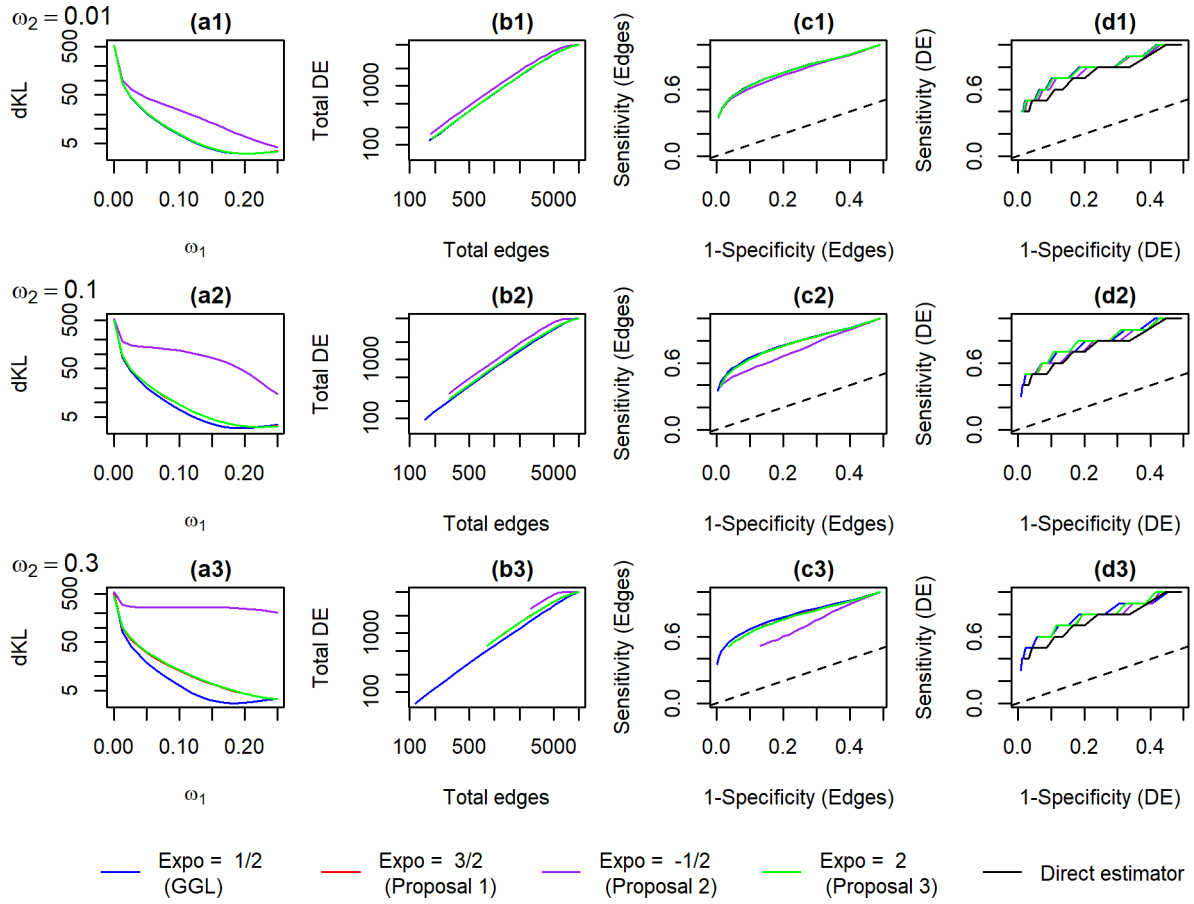

Figure 7.5.9: **Performance of the proposals for precision matrices of dimension** (100×100) **and sample size**$= 100$. Results generated with each value of ω_2 are displayed by rows. In all panels, the curves are generated with a varying tuning parameter ω_1. The proposals and the GGL in Danaher *et al.* (2014) are plotted in different colors. The Kullback-Leibler divergence, of the estimated networks from the true ones, are plotted against ω_1 values in panels (a1-a3). The total number of edges and differential edges (DE) detection are plotted in panels (b1-b3). The RoC curves of the edges and differential edges (DE) are plotted in panels (c1-c3) and (d1-d3) respectively. The dashed lines represents the perfect chance.

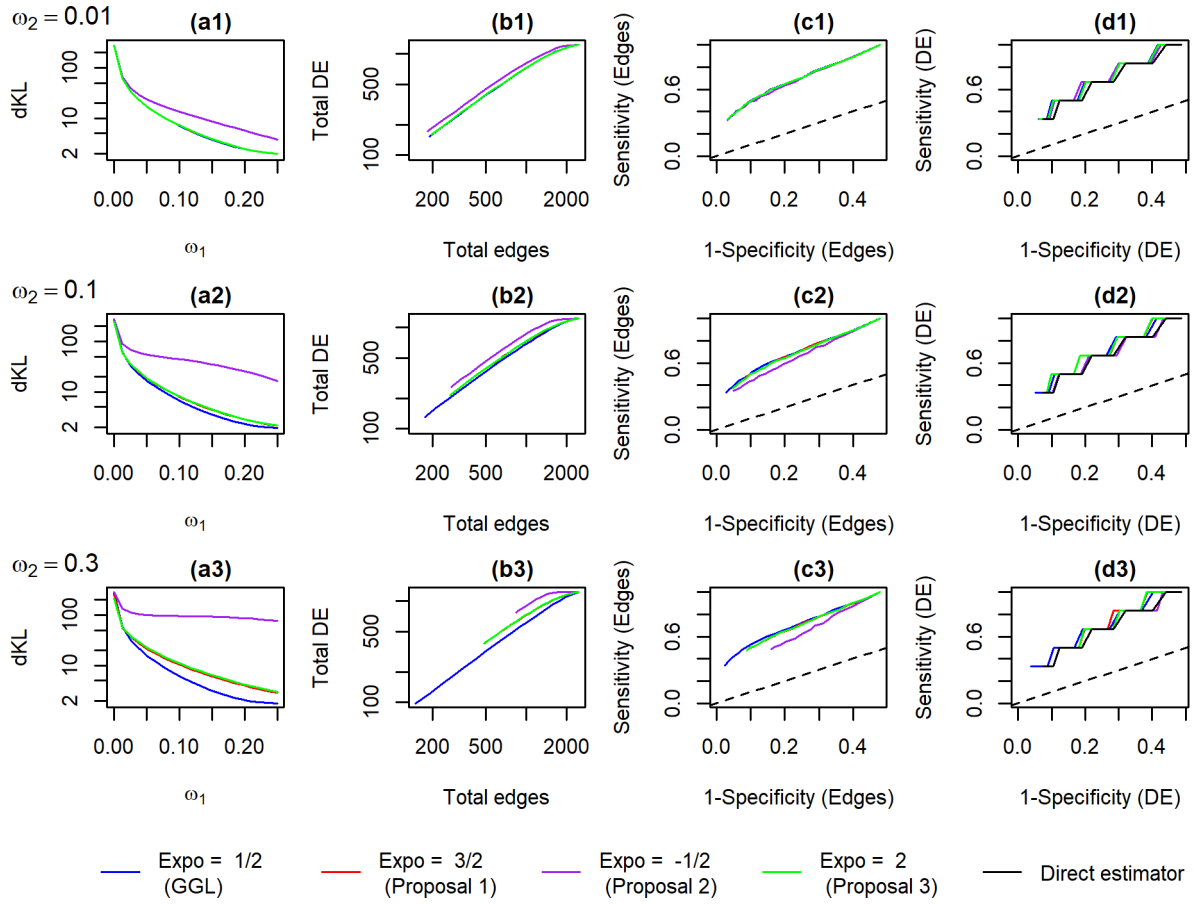

Figure 7.5.10: **Performance of the proposals for precision matrices of dimension** (50×50) **and sample size**$= 50$. Results generated with each value of ω_2 are displayed by rows. In all panels, the curves are generated with a varying tuning parameter ω_1. The proposals and the GGL in Danaher *et al.* (2014) are plotted in different colors. The Kullback-Leibler divergence, of the estimated networks from the true ones, are plotted against ω_1 values in panels (a1-a3). The total number of edges and differential edges (DE) detection are plotted in panels (b1-b3). The RoC curves of the edges and differential edges (DE) are plotted in panels (c1-c3) and (d1-d3) respectively. The dashed lines represents the perfect chance.

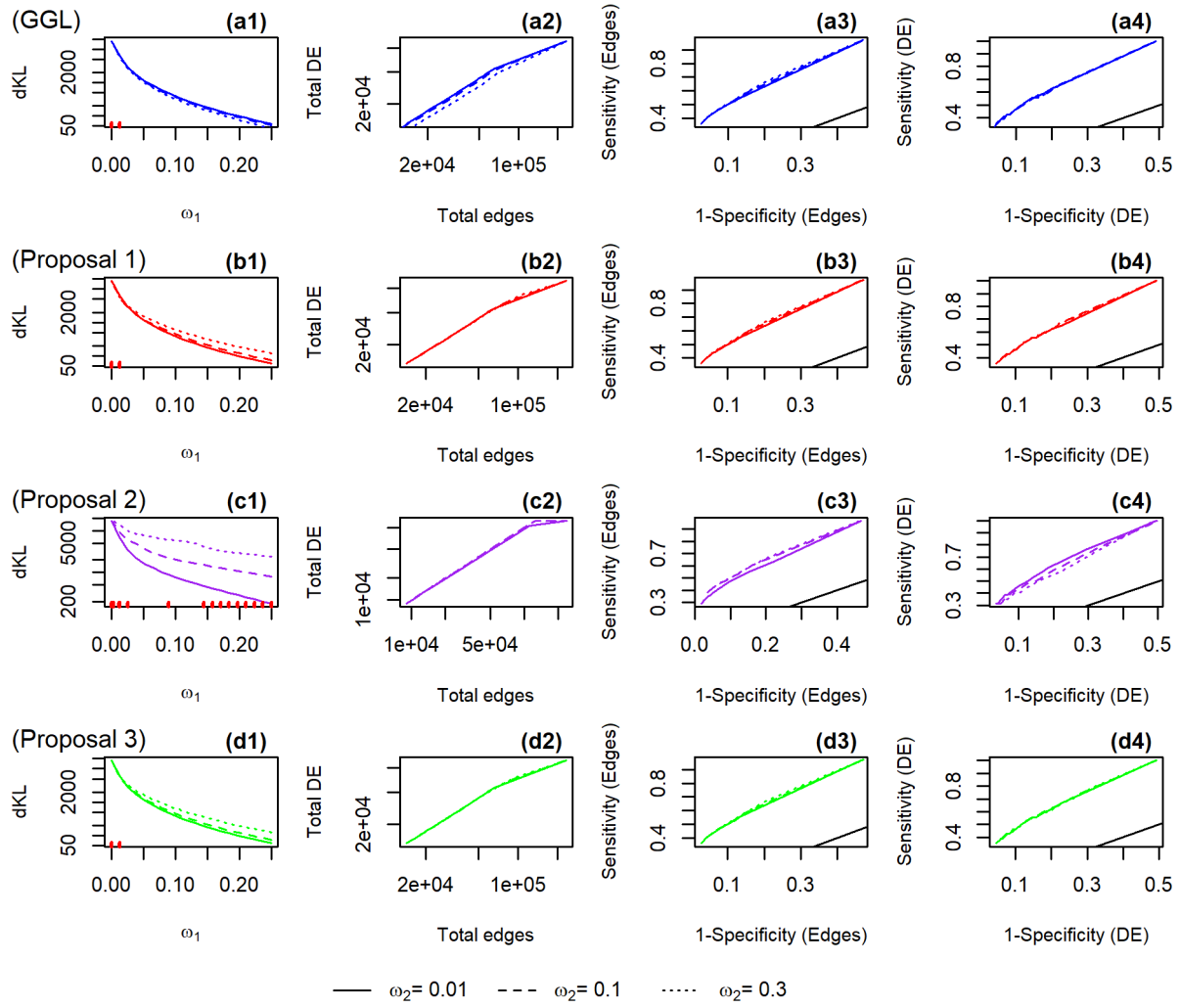

Figure 7.5.11: **Performance of the proposals for precision matrices of dimension** (500×500) **and sample size**= 50. The curves are medians over 100 replicates. Results generated with different values of ω_2 are displayed within each panel (plain, dashed and dotted lines). In all panels, the curves were generated with a varying tuning parameter ω_1 values. The proposal and the GGL penalty are plotted in different colors and arranged in rows. The Kullback-Leibler divergence, of the estimated networks from the true ones, are plotted against ω_1 values in column 1 (panels a1-d1). The red rug lines on the x-axis indicate the non-convergence of the ADMM algorithm. The total number of edges and differential edges (DE) detection are plotted in column 2 (panels a2-d2). The RoC curves of the edges and differential edges (DE) are plotted in column 3 (panels (a3-d3) and 4 (a4-d4) respectively. The black diagonal lines in panels (a3-d3 and a4-d4) represent the perfect chance.

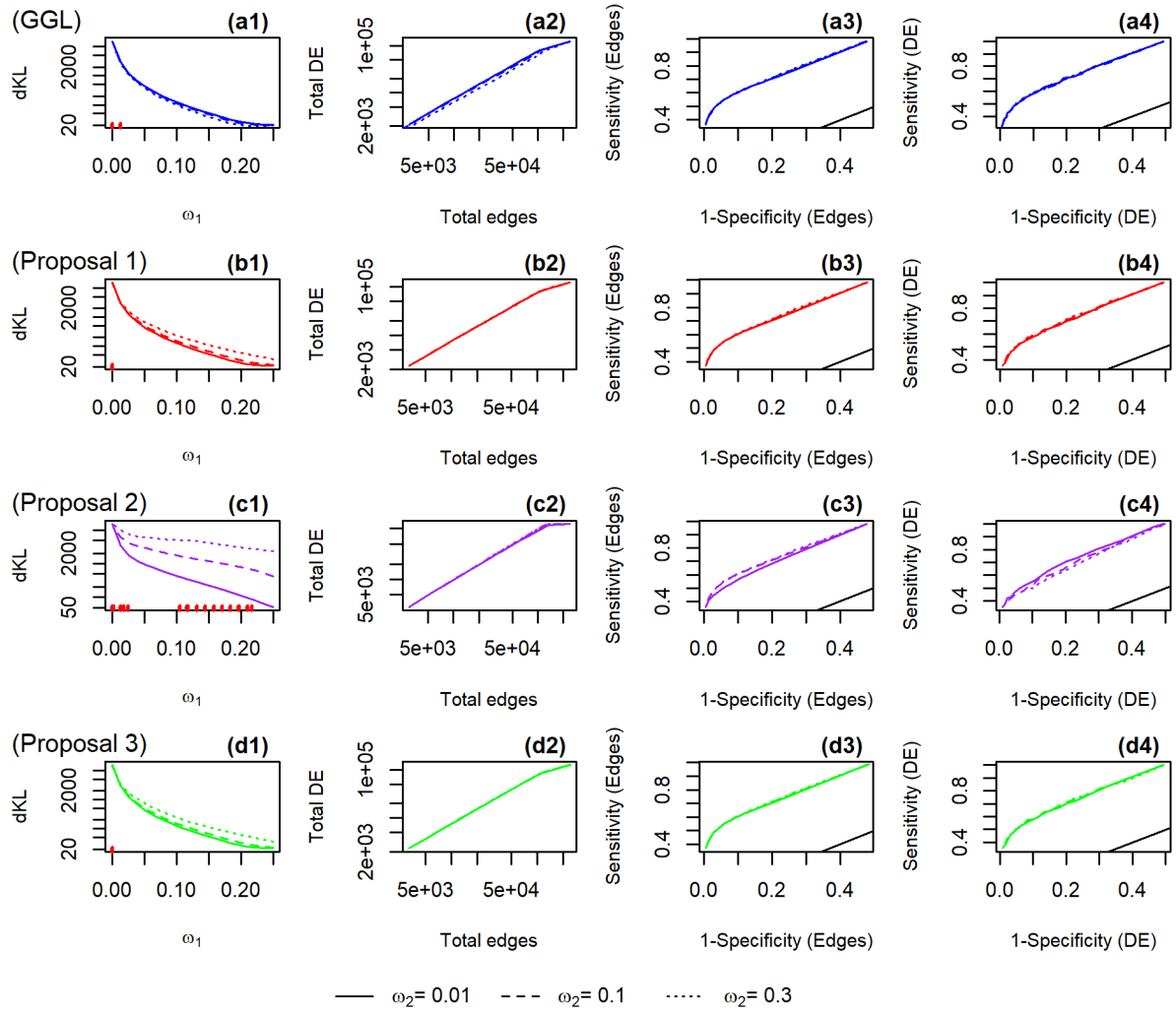

Figure 7.5.12: **Performance of the proposals for precision matrices of dimension** (500×500) **and sample size**$= 100$. The curves are medians over 100 replicates. Results generated with different values of ω_2 are displayed within each panel (plain, dashed and dotted lines). In all panels, the curves were generated with a varying tuning parameter ω_1 values. The proposal and the GGL penalty are plotted in different colors and arranged in rows. The Kullback-Leibler divergence, of the estimated networks from the true ones, are plotted against ω_1 values in column 1 (panels a1-d1). The red rug lines on the x-axis indicate the non-convergence of the ADMM algorithm. The total number of edges and differential edges (DE) detection are plotted in column 2 (panels a2-d2). The RoC curves of the edges and differential edges (DE) are plotted in column 3 (panels (a3-d3) and 4 (a4-d4) respectively. The black diagonal lines in panels (a3-d3 and a4-d4) represent the perfect chance.

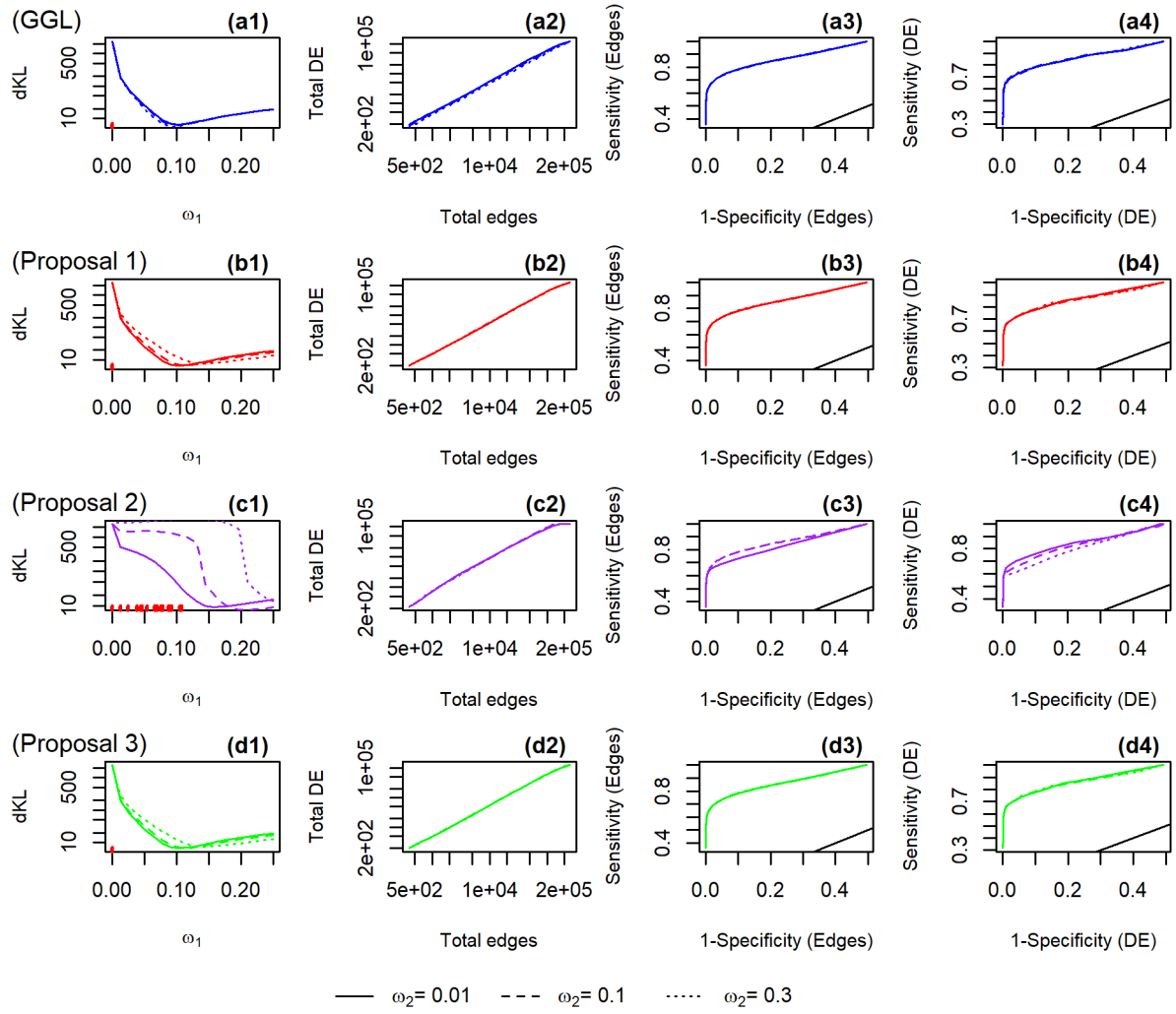

Figure 7.5.13: **Performance of the proposals for precision matrices of dimension** (500×500) **and sample size**$= 500$. The curves are medians over 100 replicates. Results generated with different values of ω_2 are displayed within each panel (plain, dashed and dotted lines). In all panels, the curves were generated with a varying tuning parameter ω_1 values. The proposal and the GGL penalty are plotted in different colors and arranged in rows. The Kullback-Leibler divergence, of the estimated networks from the true ones, are plotted against ω_1 values in column 1 (panels a1-d1). The red rug lines on the x-axis indicate the non-convergence of the ADMM algorithm. The total number of edges and differential edges (DE) detection are plotted in column 2 (panels a2-d2). The RoC curves of the edges and differential edges (DE) are plotted in column 3 (panels (a3-d3) and 4 (a4-d4) respectively. The black diagonal lines in panels (a3-d3 and a4-d4) represent the perfect chance.

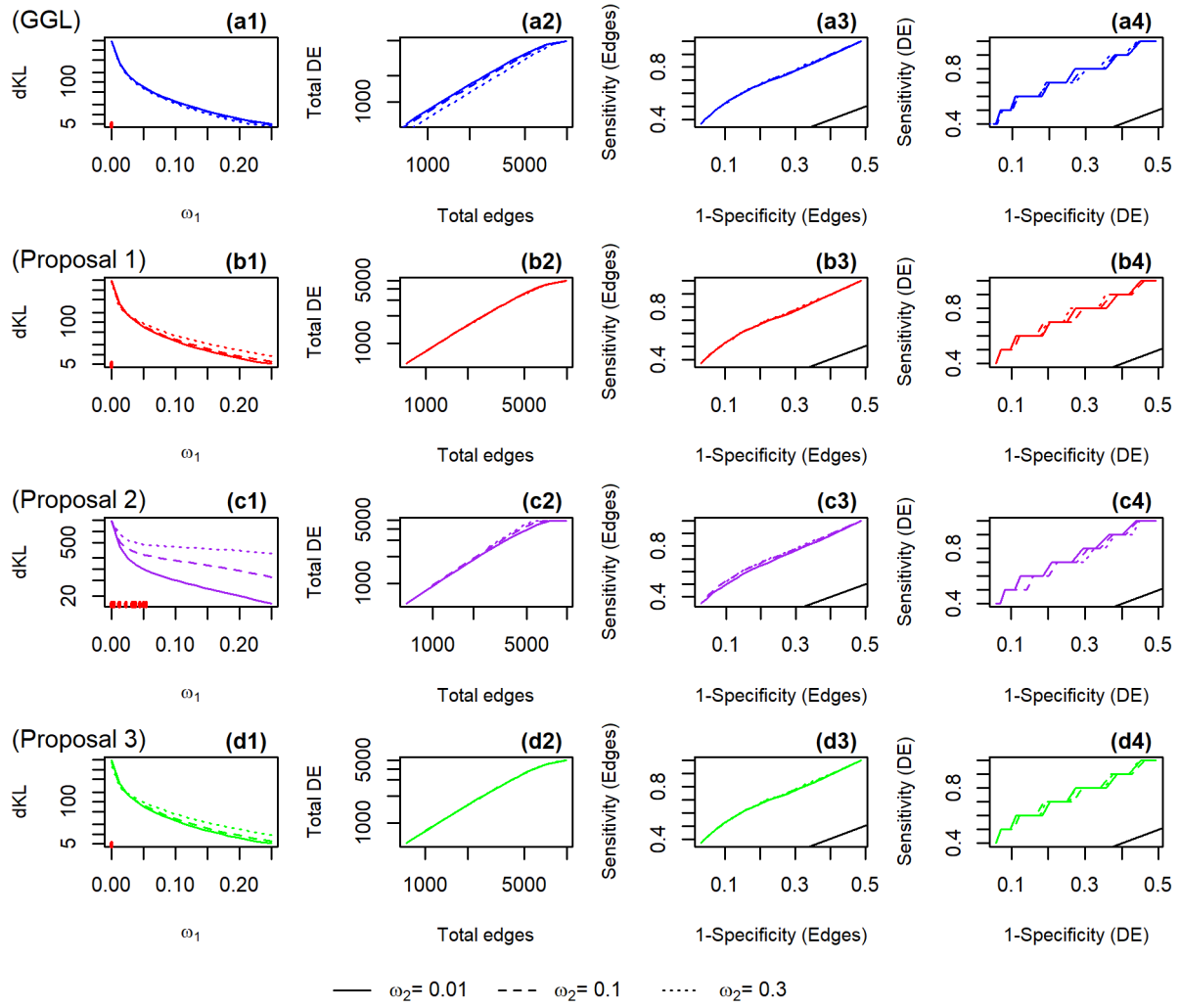

Figure 7.5.14: **Performance of the proposals for precision matrices of dimension** (100×100) **and sample size**= 50. The curves are medians over 100 replicates. Results generated with different values of ω_2 are displayed within each panel (plain, dashed and dotted lines). In all panels, the curves were generated with a varying tuning parameter ω_1 values. The proposal and the GGL penalty are plotted in different colors and arranged in rows. The Kullback-Leibler divergence, of the estimated networks from the true ones, are plotted against ω_1 values in column 1 (panels a1-d1). The red rug lines on the x-axis indicate the non-convergence of the ADMM algorithm. The total number of edges and differential edges (DE) detection are plotted in column 2 (panels a2-d2). The RoC curves of the edges and differential edges (DE) are plotted in column 3 (panels (a3-d3) and 4 (a4-d4) respectively. The black diagonal lines in panels (a3-d3 and a4-d4) represent the perfect chance.

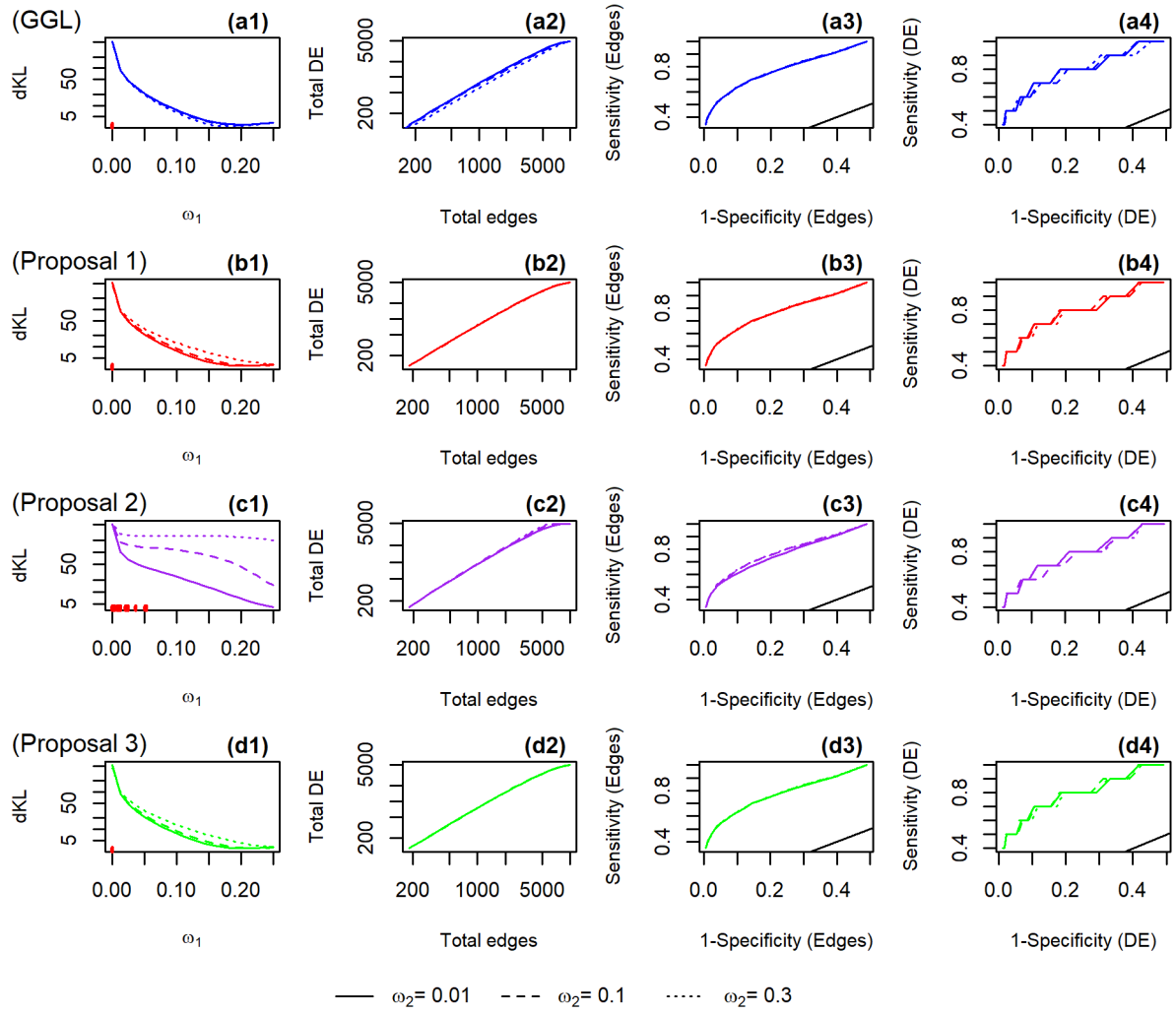

Figure 7.5.15: **Performance of the proposals for precision matrices of dimension** (100×100) **and sample size**$= 100$. The curves are medians over 100 replicates. Results generated with different values of ω_2 are displayed within each panel (plain, dashed and dotted lines). In all panels, the curves were generated with a varying tuning parameter ω_1 values. The proposal and the GGL penalty are plotted in different colors and arranged in rows. The Kullback-Leibler divergence, of the estimated networks from the true ones, are plotted against ω_1 values in column 1 (panels a1-d1). The red rug lines on the x-axis indicate the non-convergence of the ADMM algorithm. The total number of edges and differential edges (DE) detection are plotted in column 2 (panels a2-d2). The RoC curves of the edges and differential edges (DE) are plotted in column 3 (panels (a3-d3) and 4 (a4-d4) respectively. The black diagonal lines in panels (a3-d3 and a4-d4) represent the perfect chance.

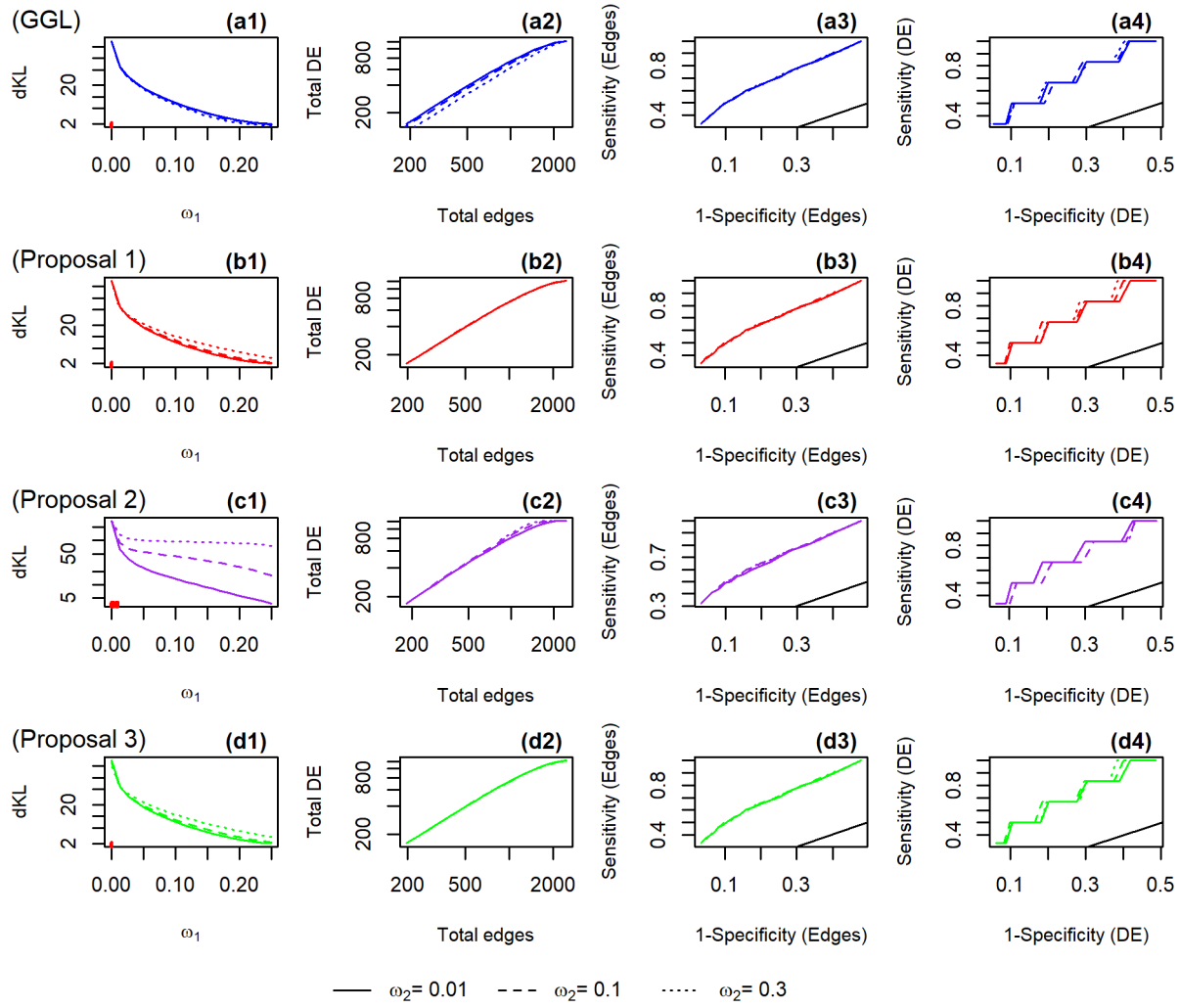

Figure 7.5.16: **Performance of the proposals for precision matrices of dimension** (50×50) **and sample size**$= 50$. The curves are medians over 100 replicates. Results generated with different values of ω_2 are displayed within each panel (plain, dashed and dotted lines). In all panels, the curves were generated with a varying tuning parameter ω_1 values. The proposal and the GGL penalty are plotted in different colors and arranged in rows. The Kullback-Leibler divergence, of the estimated networks from the true ones, are plotted against ω_1 values in column 1 (panels a1-d1). The red rug lines on the x-axis indicate the non-convergence of the ADMM algorithm. The total number of edges and differential edges (DE) detection are plotted in column 2 (panels a2-d2). The RoC curves of the edges and differential edges (DE) are plotted in column 3 (panels (a3-d3) and 4 (a4-d4) respectively. The black diagonal lines in panels (a3-d3 and a4-d4) represent the perfect chance.

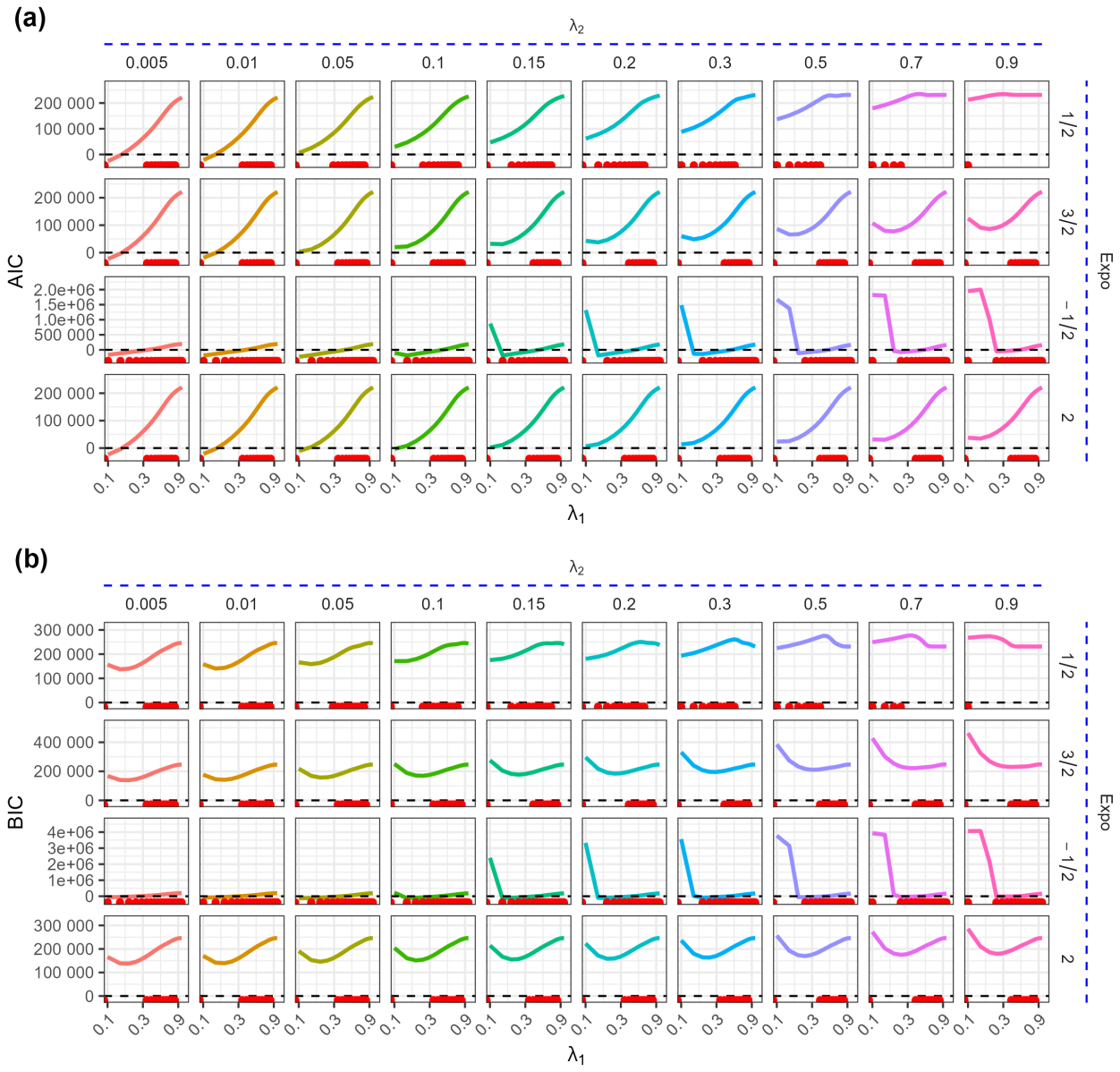

Figure 7.5.17: **Search for the optimal tuning parameters pair.** AIC (a) and BIC (b) criteria for each pair of the tuning parameters λ_1 and λ_2. The tested λ_2 values and the penalties are panelled in columns and rows, respectively. The curves were generated with a varying tuning parameter λ_1. The red rug lines on the x-axis indicate the non-convergence of the ADMM algorithm.

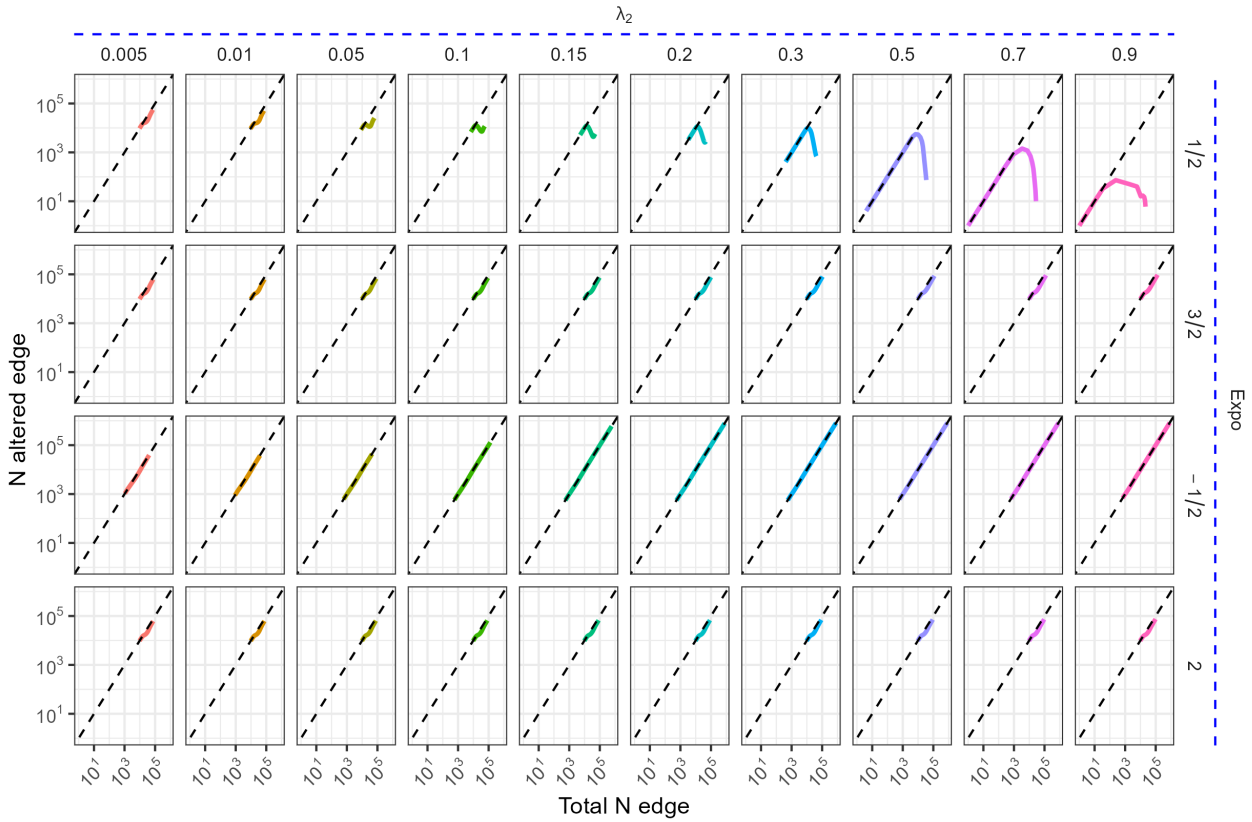

Figure 7.5.18: **Edges discovery and altered edge across classes**. The tested λ_2 values and the penalties are panelled in columns and rows, respectively. The curves were generated with a varying tuning parameter λ_1.

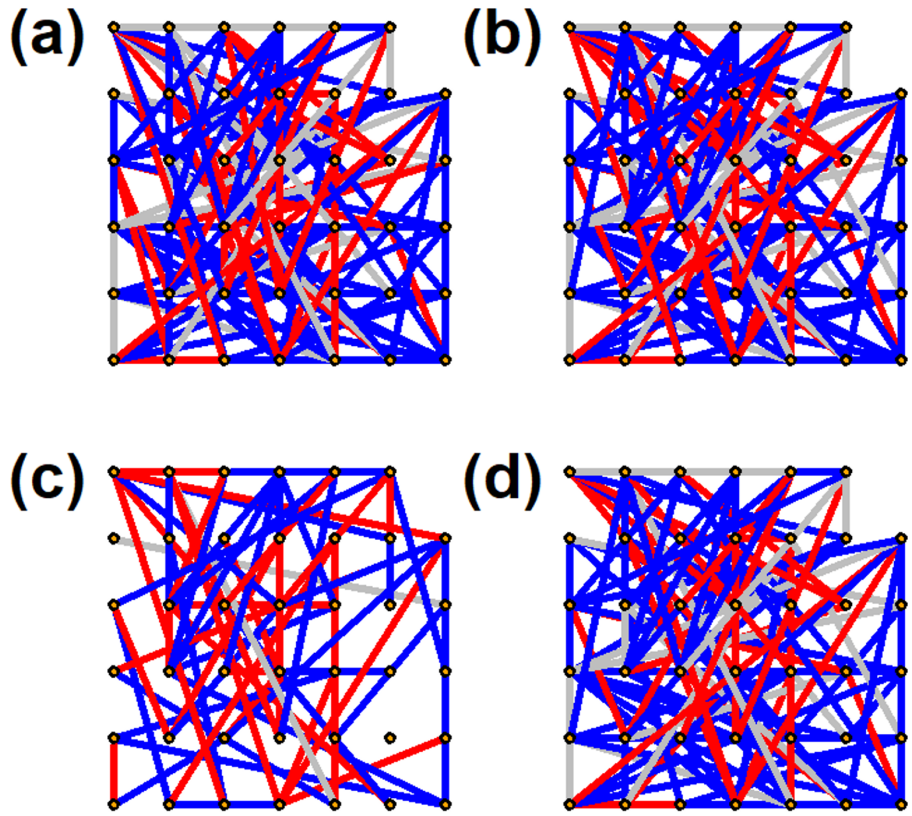

Figure 7.5.19: **Networks of the nodes adjacent to the STAT6 gene**. The networks are the results of the GGL (a), Proposal 1 (b), Proposal 2 (c) and Proposal 3 (d) regularization method. The nodes are indicated with the orange circles. All nodes are identical and positioned identically across the panels. The blue and red lines illustrate the edges that are found only in class 1 and 2, respectively, while the gray ones are those common to both classes.

7.6 Supplementary tables

				Edges		DE	
Penalty Exponent	p	n	dKL (MAD)	Sensitivity (MAD)	FPR (MAD)	Sensitivity (MAD)	FPR (MAD)
"1/2" (GGL)	50	50	2 (0)	0.340 (.044)	.030 (.007)	0.333 (.247)	.057 (.014)
	100	50	1910 (6)	1.000 (.000)	.488 (.000)	1.000 (.000)	.493 (.000)
		100	3 (0)	0.444 (.045)	.017 (.007)	0.500 (.148)	.030 (.012)
	500	50	6337 (8)	0.541 (.012)	.128 (.000)	0.648 (.055)	.220 (.001)
		100	3012 (5)	0.713 (.013)	.210 (.000)	0.833 (.027)	.322 (.001)
		500	6 (0)	0.656 (.007)	.006 (.002)	0.630 (.027)	.007 (.002)
"3/2" (Proposal 1)	50	50	2 (0)	0.350 (.044)	.035 (.005)	0.333 (.247)	.066 (.010)
	100	50	1904 (6)	1.000 (.000)	.488 (.000)	1.000 (.000)	.493 (.000)
		100	3 (0)	0.439 (.038)	.019 (.007)	0.500 (.148)	.036 (.011)
	500	50	6189 (7)	0.551 (.013)	.132 (.000)	0.648 (.055)	.227 (.001)
		100	3008 (4)	0.714 (.013)	.211 (.000)	0.833 (.027)	.325 (.001)
		500	7 (0)	0.651 (.006)	.010 (.000)	0.667 (.055)	.012 (.000)
"-1/2" (Proposal 2)	50	50	26 (2)	0.420 (.044)	.093 (.008)	0.500 (.247)	.184 (.015)
	100	50	180 (4)	0.429 (.038)	.109 (.003)	0.600 (.148)	.215 (.007)
		100	51 (1)	0.566 (.038)	.100 (.003)	0.700 (.148)	.193 (.004)
	500	50	2641 (8)	0.261 (.012)	.037 (.000)	0.370 (.055)	.075 (.000)
		100	1528 (8)	0.444 (.017)	.070 (.000)	0.574 (.055)	.139 (.001)
		500	192 (1)	0.711 (.009)	.074 (.000)	0.796 (.055)	.144 (.001)
"2" (Proposal 3)	50	50	2 (0)	0.350 (.044)	.035 (.005)	0.333 (.247)	.066 (.010)
	100	50	1879 (7)	1.000 (.000)	.488 (.000)	1.000 (.000)	.493 (.000)
		100	3 (0)	0.439 (.030)	.019 (.007)	0.500 (.148)	.035 (.012)
	500	50	6086 (6)	0.555 (.012)	.134 (.000)	0.667 (.055)	.229 (.001)
		100	3005 (4)	0.714 (.013)	.211 (.000)	0.833 (.027)	.325 (.001)
		500	7 (0)	0.651 (.007)	.010 (.000)	0.667 (.055)	.012 (.000)

Table 7.6.1: **Performance of the proposals as a function of the number of features and sample size**. The optimal combination of tuning parameters, based on the AIC, were selected from each run/replicate. The results are medians over 100 replicates. The Kullback-Leibler divergence (dKL), the sensitivity and the false positive rate (FPR=1-specificity) of edges and differential edges detection are provided.

GGL		
FBXO7|SFTPC	OAZ2|THOC1	CLK3|SLPI
CLK3|GJB3	CLK3|ING1	UBE4B|FKBP15
AURKAPS1|SLC35E1	MPI|BRE	MPI|NCOA1
MPI|MPPE1	MPI|DCP1A	MAPKAPK3|DNAJB12
EDEM1|MICALL1	ATG13|CYB561	GJB3|ELK1
BCL9|CRTC3	DOLK|NAT10	FIG4|HIF1AN
DMTN|PI4KB	BRE|TAF5L	SFTPC|AMBRA1
MPPE1|EIF1	TPSB2|MAP2K5	MPPE1|NFATC2IP
TPSB2|TMEM265	GIPC1|RMDN3	CYB561|RHD
IKBKG|RPL28	B4GALT3|YWHAZ	NCOA1|MLX
TRIM27|COQ6	ATP6V0A1|LMNB2	ATP6V0A1|TRAFD1
CA5B|IRGQ	AMBRA1|POLR2C	MBOAT2|HDAC3
KLHDC3|EIF2B4	NAT10|C19orf66	COQ6|FAM192A
DCP1A|DCTPP1	PHF10|STK19	FAM192A|EPS15L1
PHF10|MIIP	AMBRA1|SNX11	
Proposal 1 ("3/2") and Proposal 3 ("2")		
CLK3|SLPI	CLK3|ING1	FIG4|HIF1AN
BRE|TAF5L	MPPE1|EIF1	B4GALT3|YWHAZ
FAM192A|COQ6		
Proposal 2 ("-1/2")		
GOT2|SYN2	HADH|POGZ	FBXO7|SFTPC
CHD4|C10orf76	INPPL1|POLR2C	EIF4G3|CCDC51
PLOD3|NOTCH3	IPO13|RPL28	FIG4|TTPAL
BCL9|RAE1	AKR1C1|WDR48	CCL14|C19orf66
TREX1|TJAP1	SFTPC|SLC9A1	CSNK1A1|ATN1
MPPE1|B4GALT3	RPL28|IKBKG	RHD|HS1BP3
DCTN1|MTMR14	EIF1|GON4L	MARCKS|HPS6
XPO6|TNIP2	Trp53|SDHAF1	ZNF672|XKR8
CDCA4|MLYCD		

Table 7.6.2: **Sign-altered edges in the estimated graphical models**. Probes were mapped to their respective genes. Each pair represents one edge in the network. The vertical bar separates each node across the edge.

Bibliography

[1] O. Banerjee and L. El Ghaoui, "Model selection through sparse maximum likelihood estimation for multivariate Gaussian or binary data," *Journal of Machine Learning Research*, vol. 9, pp. 485–516, 2008.

[2] D. Barber, *Bayesian Reasoning and Machine Learning*. Cambridge University Press, 2012.

[3] A. Benfenati, E. Chouzenoux, L. Duval, J.-C. Pesquet, and A. Pirayre, "A review on graph optimization and algorithmic frameworks," Research Report, 2018. [Online]. Available: `https://hal.archives-ouvertes.fr/hal-01901499`.

[4] A. Blake, P. Kohli, and C. Rother, "Introduction to Markov random fields," in *Markov Random Fields for Vision and Image Processing*. The MIT Press, 2018, ch. 1, pp. 1–28.

[5] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends in Machine Learning*, vol. 3, no. 1, pp. 1–122, 2010.

[6] P. Bühlmann and S. van de Geer, "Graphical modeling," in *Statistics for High-Dimensional Data*, Springer Series in Statistics, Springer Berlin Heidelberg, 2011, ch. 13.

[7] R. Castelo and A. Roverato, "A Robust Procedure For Gaussian Graphical Model Search From Microarray Data With p Larger Than n," vol. 7, pp. 2621–2650, 2006.

[8] H. Chen and B. M. Sharp, "Content-rich biological network constructed by mining PubMed abstracts," *BMC Bioinformatics*, vol. 5, no. 1, pp. 1–13, 2004.

[9] M. Crawshaw, "Multi-task learning with deep neural networks: A survey," 2020, ArXiv: 2009.09796.

[10] P. Csermely, T. Korcsmáros, and R. Nussinov, "Intracellular and intercellular signaling networks in cancer initiation, development and precision anti-cancer therapy. RAS acts as contextual signaling hub," *Seminars in cell & developmental biology*, vol. 58, p. 55, 2016.

[11] P. Danaher, *Package 'JGL' Performs the Joint Graphical Lasso for Sparse Inverse Covariance Estimation on Multiple Classes*, 2018. [Online]. Available: `https://cran.r-project.org/web/packages/JGL/index.html`.

[12] P. Danaher, P. Wang, and D. M. Witten, "The joint graphical lasso for inverse covariance estimation across multiple classes," *Journal of the Royal Statistical Society. Series B: Statistical Methodology*, vol. 76, no. 2, pp. 373–397, 2014.

[13] J. N. Darroch and D. Ratcliff, "Generalized Iterative Scaling for Log-Linear Models," *The Annals of Mathematical Statistics*, vol. 43, no. 5, pp. 1470–1480, 1972.

[14] Y. Delgado-Ramirez, V. Colly, G. V. Gonzalez, and S. Leon-Cabrera, "Signal transducer and activator of transcription 6 as a target in colon cancer therapy (Review)," *Oncology Letters*, vol. 20, no. 1, pp. 455–464, 2020.

[15] A. P. Dempster, "Covariance Selection," *Biometrics*, vol. 28, no. 1, p. 157, 1972.

[16] D. Dokuzoglu and V. Purutçuoglu Gazi, "Comprehensive analyses of gaussian graphical model under different biological networks," *Acta Physica Polonica A*, pp. 1106–1111, 2017.

[17] D. L. Donoho and J. M. Johnstone, "Ideal spatial adaptation by wavelet shrinkage," *Biometrika*, vol. 81, no. 3, pp. 425–455, 1994.

[18] M. Drton and M. D. Perlman, "Multiple testing and error control in Gaussian graphical model selection," *Statistical Science*, vol. 22, no. 3, pp. 430–449, 2007.

[19] D. Edwards, *Introduction to Graphical Modelling*. Springer New York, 2000, Springer Texts in Statistics.

[20] J. Friedman, T. Hastie, H. Höfling, and R. Tibshirani, "Pathwise coordinate optimization," *The Annals of Applied Statistics*, vol. 1, no. 2, pp. 302–332, 2007.

[21] J. Friedman, T. Hastie, and R. Tibshirani, "Sparse inverse covariance estimation with the graphical lasso," *Biostatistics*, vol. 9, no. 3, pp. 432–441, 2008.

[22] J. Friedman, T. Hastie, and R. Tibshirani, *glasso: Graphical lasso for R*, R package manual, URL: `https://CRAN.R-project.org/package=glasso`, 2019.

[23] K. J. Friston, "Functional and Effective Connectivity: A Review," *Brain Connectivity*, vol. 1, no. 1, pp. 13–36, 2011.

[24] J. Guo, E. Levina, G. Michailidis, and J. Zhu, "Joint estimation of multiple graphical models," *Biometrika*, vol. 98, no. 1, pp. 1–15, 2011.

[25] A. M. Gustafson, R. Soldi, C. Anderlind, M. B. Scholand, J. Qian, X. Zhang, K. Cooper, D. Walker, A. Mcwilliams, L. Gang, E. Szabo, J. Brody, P. P. Massion, M. E. Lenburg, L. Stephen, A. H. Bild, and A. Spira, "Airway PI3K pathway activation is an early and reversible event in lung cancer development," vol. 2, no. 26, 26ra25, 2010.

[26] N. J. Higham, "Cholesky factorization," *Wiley interdisciplinary reviews: computational statistics*, vol. 1, no. 2, pp. 251–254, 2009.

[27] A. E. Hoerl and R. W. Kennard, "Ridge regression: Applications to nonorthogonal problems," *Technometrics*, vol. 12, no. 1, pp. 69–82, 1970.

[28] S. Højsgaard, D. Edwards, and S. Lauritzen, *Graphical models with R*. Springer US, 2012.

[29] J. Honorio, T. Jaakkola, and D. Samaras, "Simultaneous and group-sparse multi-task learning of Gaussian graphical models," 2015, arXiv:1207.4255.

[30] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An Introduction to Statistical Learning*. Springer-Verlag New York, 2013.

[31] H. Jiang, X. Fei, H. Liu, K. Roederand, J. Lafferty, L. Wasserman, X. Li, and T. Zhao, *huge: High-dimensional undirected graph estimation*, R package manual, URL: `https://CRAN.R-project.org/package=huge`, 2020.

[32] P. F. Jonsson and P. A. Bates, "Global topological features of cancer proteins in the human interactome," *Bioinformatics (Oxford, England)*, vol. 22, no. 18, pp. 2291–2297, 2006.

[33] E. D. Kolaczyk, *Statistical Analysis of Network Data*. New York, NY: Springer New York, 2009, Springer Series in Statistics.

[34] S. L. Lauritzen, "Multivariate normal models," in *Graphical models*, Clarendon Press, 1996, ch. 5, pp. 123–157.

[35] W. Lee and Y. Liu, "Joint estimation of multiple precision matrices with common structures," *Journal of Machine Learning Research*, vol. 16, pp. 1035–1062, 2015.

[36] H. Liu, K. Roeder, and L. Wasserman, "Stability Approach to Regularization Selection (StARS) for High Dimensional Graphical Models," *Advances in neural information processing systems*, vol. 24, no. 2, pp. 1432–1440, 2010.

[37] F. Markowetz and R. Spang, "Inferring cellular networks – a review," *BMC Bioinformatics*, vol. 8, no. Suppl 6, S5, 2007.

[38] N. Meinshausen and P. Bühlmann, "High-dimensional graphs and variable selection with the Lasso," *Annals of Statistics*, vol. 34, no. 3, pp. 1436–1462, 2006.

[39] K. Ming Tan, *hglasso: Learning graphical models with hubs*, R package manual, URL: `https://rdrr.io/cran/hglasso/man/hglasso.html`, 2019.

[40] K. Ming Tan, P. London, K. Mohan, M. Fazel, D. Witten, S.-I. Lee, and D. Witten Tan, "Learning graphical models with hubs," *Journal of Machine Learning Research*, vol. 15, pp. 3297–3331, 2014.

[41] M. E. J. Newman, "The structure and function of complex networks," *Society for Industrial & Applied Mathematics Review*, vol. 45, no. 2, pp. 167–256, 2003.

[42] J. S. Oliveira, C. G. Bailey, J. B. Jones-Oliveira, D. A. Dixon, D. W. Gull, and M. L. Chandler, "A Computational Model for the Identification of Biochemical Pathways in the Krebs Cycle," *Journal of computational biology*, vol. 10, no. 1, 2003.

[43] M. R. Osborne, B. Presnell, and B. A. Turlach, "On the lasso and its dual," *Journal of Computational and Graphical Statistics*, vol. 9, no. 2, pp. 319–337, 2000.

[44] G. A. Pavlopoulos, M. Secrier, C. N. Moschopoulos, T. G. Soldatos, S. Kossida, J. Aerts, R. Schneider, and P. G. Bagos, "Using graph theory to analyze biological networks," *BioData Mining*, vol. 4, no. 1, pp. 1–27, 2011.

[45] J. Pearl, "Probabilistic reasoning in intelligent systems : Networks of plausible inference," in *The Morgan Kaufmann series in representation and reasoning*, Morgan Kaufmann, 1988.

[46] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, PrettenhoferPeter, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research*, vol. 12, no. 85, pp. 2825–2830, 2011. [Online]. Available: `http://jmlr.org/papers/v12/pedregosa11a.html`.

[47] J. Peng, P. Wang, N. Zhou, and J. Zhu, "Partial Correlation Estimation by Joint Sparse Regression Models," *Journal of the American Statistical Association*, vol. 104, no. 486, p. 735, 2009.

[48] H. Qiu, F. Han, H. Liu, and B. Caffo, "Joint estimation of multiple graphical models from high dimensional time series," *Journal of the Royal Statistical Society. Series B: Statistical Methodology*, vol. 78, no. 2, pp. 487–504, 2016.

[49] P. Ravikumar, M. J. Wainwright, G. Raskutti, and B. Yu, "High-dimensional covariance estimation by minimizing ℓ_1-penalized log-determinant divergence," *Electronic Journal of Statistics*, vol. 5, pp. 935–980, 2011.

[50] M. Scutari and K. Strimmer, "Introduction to Graphical Modelling," in *Handbook of Statistical Systems Biology*, John Wiley & Sons, Ltd, 2011, ch. 11, pp. 235–54.

[51] N. Simon and R. Tibshirani, "Discriminant Analysis with Adaptively Pooled Covariance," 2011.

[52] A. Spira, J. E. Beane, V. Shah, K. Steiling, G. Liu, F. Schembri, S. Gilman, Y. M. Dumas, P. Calner, P. Sebastiani, S. Sridhar, J. Beamis, C. Lamb, T. Anderson, N. Gerry, J. Keane, M. E. Lenburg, and J. S. Brody, "Airway epithelial gene expression in the diagnostic evaluation of smokers with suspect lung cancer," *Nature medicine*, vol. 13, no. 3, pp. 361–366, 2007.

[53] L. E. Sucar, *Probabilistic Graphical Models*. Springer London, 2015, Advances in Computer Vision and Pattern Recognition.

[54] E. B. Sudderth, "Graphical models for visual object recognition and tracking," Unpublished Ph.D. thesis, Massachusetts Institute of Technology, 2006.

[55] M. A. Sustik, B. Calderhead, and J. Clavel, *glassoFast: Fast graphical lasso*, R package manual, URL: `https://CRAN.R-project.org/package=glassoFast`, 2018.

[56] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society. Series B: Statistical Methodology*, vol. 58, no. 1, pp. 267–288, 1996.

[57] R. Tibshirani, M. A. Saunders, S. Rosset, J. Zhu, and K. Knight, "Sparsity and smoothness via the fused lasso," *Journal of the Royal Statistical Society. Series B: Statistical Methodology*, vol. 67, 2005.

[58] R. J. Tibshirani, "The lasso problem and uniqueness," *Electronic Journal of Statistics*, vol. 7, no. 1, pp. 1456–1490, 2012.

[59] T. W. Tong and K. Lange, "Coordinate descent algorithms for lasso penalized regression," *The Annals of Applied Statistics*, vol. 2, no. 1, pp. 224–244, 2008.

[60] N. Wermuth, "Model Search among Multiplicative Models," *Biometrics*, vol. 32, no. 2, pp. 253–263, 1976.

[61] J. Whittaker, "The inverse variance," in *Graphical Models in Applied Multivariate Statistics*, Wiley, 1990, ch. 5, pp. 120–150.

[62] E. Wit and J. McClure, *Statistics for microarrays: design, analysis and inference*. John Wiley & Sons, 2004.

[63] D. M. Witten and R. Tibshirani, "Covariance-regularized regression and classification for high dimensional problems," *Journal of the Royal Statistical Society. Series B: Statistical Methodology*, vol. 71, no. 3, pp. 615–636, 2009.

[64] D. M. Witten and R. Tibshirani, "Penalized classification using Fisher's linear discriminant," *Journal of the Royal Statistical Society. Series B: Statistical Methodology*, vol. 73, no. 5, pp. 753–772, 2011.

[65] S. J. Wright, "Coordinate descent algorithms," *Mathematical Programming*, vol. 151, no. 1, pp. 3–34, 2015.

[66] M. Yuan and Y. Lin, "Model selection and estimation in the Gaussian graphical model," *Biometrika*, vol. 94, no. 1, pp. 19–35, 2007.

[67] I. S. Zeng and L. Thomas, *sparsenetgls: Using Gaussian graphical structue learning estimation in generalized least squared regression for multivariate normal regression*, R package manual, URL: `https://www.bioconductor.org/packages/devel/bioc/vignettes/sparsenetgls/inst/doc/vignettes_sparsenetgls.html`, 2021.

[68] S. D. Zhao, T. T. Cai, and H. Li, "Direct estimation of differential networks," *Biometrika*, vol. 101, no. 2, pp. 253–268, 2014.