GAUSSIAN GRAPHICAL APPROACHES IN ESTIMATION OF BIOLOGICAL SYSTEMS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

EZGİ AYYILDIZ

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
STATISTICS

MAY 2013

Approval of the thesis:

GAUSSIAN GRAPHICAL APPROACHES IN ESTIMATION OF BIOLOGICAL SYSTEMS

submitted by **EZGİ AYYILDIZ** in partial fulfillment of the requirements for the degree of **Master of Science in Statistics Department, Middle East Technical University** by,

Prof. Dr. Canan Özgen
Dean, Graduate School of **Natural and Applied Sciences** _______________

Prof. Dr. İnci Batmaz
Head of Department, **Statistics** _______________

Assoc. Prof. Dr. Vilda Purutçuoğlu
Supervisor, **Statistics Department, METU** _______________

Examining Committee Members:

Prof. Dr. İnci Batmaz
Statistics Department, METU _______________

Assoc. Prof. Dr. Vilda Purutçuoğlu
Supervisor, Statistics Department, METU _______________

Prof. Dr. Gerhard-Wilhelm Weber
Institute of Applied Mathematics, METU _______________

Assoc. Prof. Dr. Barış Sürücü
Statistics Department, METU _______________

Assist. Prof. Dr. Aybar Can Acar
Graduate School of Informatics, METU _______________

Date: _______________

ABSTRACT

GAUSSIAN GRAPHICAL APPROACHES IN ESTIMATION OF BIOLOGICAL SYSTEMS

Ayyıldız, Ezgi
M.S., Department of Statistics
Supervisor : Assoc. Prof. Dr. Vilda Purutçuoğlu

May 2013, 71 pages

The Gaussian Graphical Model (GGM) is one of the well-known deterministic inference methods which is based on the conditional independency of nodes in the system. In this study we consider to implement this approach in small and relatively large networks under different singularity and sparsity conditions. In inference of these systems we perform lasso and L_1-penalized lasso regression approaches and select the best fitted model to the data by using different criteria. Among many alternatives, we apply the F-measure, false positive rate, precision, and recall measures as well as cross validation method in Monte Carlo runs. According to the results of their accuracies and computational time, we choose the best criterion for the inference of realistically complex systems such as the JAK-STAT pathway. In the calculation in case we can face with singularity problem, we evaluate the performance of a recently developed technique for the matrix decompositions. This novel approach also enables us to deal with the computational problems caused by the sparsity of the networks. Finally, apart from the current model selection approaches in the GGM field, we investigate other plausible alternatives for this type of inference problems.

Keywords: Gaussian Graphical Models, Penalty Selection in L_1-Penalized Likelihood, Cross Validation Algorithm, Threshold Gradient Descent Algorithm, Singularity Problem in Covariance Matrix

ÖZ

GAUSSİAN GRAFİKSEL MODELİ İLE BİYOLOJİK SİSTEMLERİN TAHMİNİ

Ayyıldız, Ezgi
Yüksek Lisans, İstatistik Bölümü
Tez Yöneticisi : Doç. Dr. Vilda Purutçuoğlu

Mayıs 2013 , 71 sayfa

Gaussian Grafiksel Modeli, sistem düğümlerinin koşullu bağımsızlığına dayanan, en çok bilinen deterministik tahmin metotlarından biridir. Bu çalışmada, bu yaklaşım, farklı tekillik problemleri ve seyreklik koşulları olan, küçük ve daha büyük sistemler için kullanılmıştır. Bu sistemlerin tahmininde, lasso ve L_1-cezalandırmalı lasso regresyon yaklaşımları uygulanmış ve farklı kriterler kullanarak veriye en çok uyan model seçilmiştir. Bir çok alternatif arasından, F- ölçüsü, yanlış pozitif oranı, kesinlik ve hassasiyet ölçüleri ve çapraz doğrulama metodu Monte Carlo simülasyonları içinde kullanılmıştır. Sonuçların doğruluğu ve hesaplama süreleri göz önünde bulundurularak da JAK-STAT yolağı gibi gerçekçi büyüklükteki karmaşık sistemlerin tahmininde kullanılacak en iyi kriter seçilmiştir. Hesaplamalar sırasında tekillik problemiyle karşılaşılması durumunda, yeni geliştirilen bir matris ayrıştırma tekniğinin performansı değerlendirilmiştir. Bu yeni yaklaşım, aynı zamanda, sistemlerin seyrekliğinden kaynaklanan problemleri çözebilmemizi de sağlamaktadır. Sonunda ise varolan model seçme yaklaşımlarından farklı olarak, bu tür tahmin problemleri için olası başka alternatif yöntemler araştırılmaktadır.

Anahtar Kelimeler: Gaussian Grafiksel Modeli, L_1-Cezalandırılmış Olabilirlikte Ceza Seçimi, Çapraz Doğrulama Algoritması, Eşik Meyil İniş Algoritması, Kovaryans Matrisinde Tekillik Problemi

To my mom and grandma :)

ACKNOWLEDGMENTS

First of all, I would like to express great appreciation to my thesis supervisor Assoc. Prof. Dr. Vilda Purutçuoğlu for her everlasting support, advice and patience during my research. With her smiling face, love and respect to her profession and disciplined stance in academic study, she has always been a role model for me in academic career. It has been a great honor for me to work with her.

I would like to present my grateful thanks to my examining committee members, Prof. Dr. İnci Batmaz, Prof. Dr. Gerhard-Wilhelm Weber, Assoc. Prof. Dr. Barış Sürücü and Assist. Prof. Dr. Aybar Can Acar for their detailed reviews and their constructive comments.

I would also like to thank the Scientific and Technological Research Council of Turkey (TÜBİTAK) for their financial support and the EU 7 Framework Collaborative Project (Project No: 260429) for their research support.

The HPC resources were provided by the Department of Computer Engineering, Middle East Technical University.

I owe my special thanks to my friends Duygu Varol and Melih Ağraz for their full support, motivation and nice friendship. Also, I would like to thank all the members of METU Statistics Department.

I feel great appreciation to Muazzez Gürgan for being such a good cousin. Then, my special thanks go to my friends Elçin Evran, Aybike Gürbüz, Deniz Alioğlu and Özden Sönmez for their perfect friendship.

Finally, my deepest gratitude to my beloved family. I give my special thanks to my wonderful mother, Nuran Ayyıldız for her endless love and unconditional support throughout my life, to my father Haydar Ayyıldız, for teaching mathematics to me and to my brother Mert Ayyıldız, for supporting me through every step of the way.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

BN	Bayesian Network
CME	Chemical Master Equation
CPU	Central Processing Unit
DAG	Directed Acyclic Graphs
DBN	Dynamic Bayesian Network
FDR	False Discovery Rate
FN	False Negative
FP	False Positive
FPR	False Positive Rate
GGM	Gaussian Graphical Model
GRN	Gene Regulatory Netwok
IFN	Type I Interferon
JAK/STAT	Janus Kinase Signal Transducer and Activation of Transcription
MAPK/ERK	Mitogen-Activated Protein Kinase and Extracellular Signal-Regulated Kinase
MCC	Matthews Correlation Coefficient
MCMC	Monte Carlo Markov Chain
MLE	Maximum Likelihood Estimation
ODE	Ordinary Differential Equation
OLS	Ordinary Least Squares
PBN	Probabilistic Boolean Network
PPV	Positive Predictive Value
SVD	Singular Value Decomposition
TGD	Threshold Gradient Descent
TN	True Negative
TNR	True Negative Rate
TP	True Positive
TPR	True Positive Rate
TYK2	Tyrosine Kinase 2

CHAPTER 1

INTRODUCTION

Over the last few decades, biotechnology has been developed rapidly. There are many researches to explore the human genome all over the world. Since the technology is increasing day by day, more data are available about biochemical systems. So new methods are needed to make a sense of this large amount of data. There are several modelling approaches to understand the complex biochemical systems. Graphical models are widely used in the analysis of the gene regulatory networks. In particular, Gaussian graphical model (GGM) is mostly preferred undirected graphical model which suggests a deterministic solution to estimate the true structure of the network (Whittaker, 1990). In such modelling, we need to infer the inverse of the covariance-variance matrix, which is also called the *precision* matrix. In the calculation of the precision, due to the high dimensionality of the biological system of interest, singularity problem is raised. There are a number of methods to unravel this challenge such as the eigenvalue decomposition, singular value decomposition and the Cholesky decomposition. But none of these methods explicitly deals with the actual structure of the linear relationship between the states. Hereby in the first part of the thesis, the eigenvalue decomposition, singular value decomposition and the Cholesky decomposition methods are explained. After that, several approaches which are used to model biological systems are described. Then in order to evaluate these models, the well-known accuracy measures are given.

In the second part of the thesis, since the calculation becomes intractable under singularity problem, we suggest an alternative approach that can deal with this challenge (Ayyıldız et al., 2012). We explain the steps of our proposed algorithm in details. Moreover, a numerical example are given as an implementation of this new method. Then, we work on the inference of the systems via GGM. In GGM, the conditional independencies of two nodes are indicated with the absence of an edge between these nodes and are represented with zero entries in the precision matrix, under the normality assumption. Hence, in inference by GGM, we need to estimate the precision matrix to understand the structure of the system. Different optimization algorithms can be used to achieve this aim. Since the structure of the genetic networks is not fully connected, the related precision matrix has a sparse construction. Thereby, when we estimate the precision matrix, we need to take into account the sparsity condition. In this study, we consider the *k*-cross validation and the threshold gradient descent (TGD) methods since both approaches have been suggested in the literature for the inference of GGM. Whereas they have not been applied yet for the estimation in different sizes of networks and under distinct level of sparsity. Between these two alternatives, basically, the k-cross validation method divides the data into k folds and takes out one fold to compute the likelihood and uses $(k-1)$ folds to estimate the precision matrix. On the other hand, TGD algorithm uses the gradient of loss function which is the negative of the log-likelihood function in order to estimate the off-diagonal entries of the precision matrix. In this approach, the diagonal entries of the precision matrix are inferred via the Newton-Raphson algorithm that is one of the common iterative methods based on the first order derivative of a maximized objective function.

Finally, for the assessment of all the underlying approaches in the analysis, we implement distinct criteria. Firstly, we evaluate the performance of our new matrix-decomposition that is suggested to overcome the singularity problems in covariance matrices in terms of computational demand and accuracy. Then, we assess plausible model selection criteria in inference of the networks via GGM. Here we also propose certain alternative tools (i.e. model selection criteria), which have not applied yet in biological/biochemical networks to the best of our knowledge, but have been previously used in different fields such as in computer engineering. We assess the validity of each candidate criterion with the current ones in terms of accuracy. In this calculation we compare the estimated networks with respect to true ones by using the Monte Carlo runs. On the other hand, in the assessment of the cross validation and threshold gradient descent method, we evaluate their performances regarding distinct model selection criteria from computational time to accuracy. Additionally, we choose the best model selection criterion to infer different sizes of networks. In the end, we perform the best algorithm and the best model selection criterion in inference of a realistically complex JAK/STAT pathway generated by the simulated data. This pathways controls the immune system of eukaryotes and includes a large numbers of genes in its description. The JAK/STAT pathway provides a mechanism for transcriptional regulation by transmission of extracellular information from membrane to nucleus. Hence, many immune disorders are related with the disregulation of this signalling pathway. In the literature, this system has been already estimated by using different stochastic and deterministic methods. Thereby, in this study we aim to get certain new biological findings about this pathway which can be interesting for future researches by implementing a different sort of inference method. On the other side to evaluate the suggested approaches in a real dataset, we select the MAPK/ERK pathway which is one of the very common signal transaction systems in all eukaryotes. This pathway controls the cellular regulation and is widely worked on oncogene researches. In this study we have to use a summary version of this system since the available dataset is observed merely for a small set of proteins throughout the time.

As a result, in the organisation of this thesis, we initially explain the most well-known decomposition methods which are used to solve singularity problem in the following chapter. Then, we present the alternative approaches in modelling biological/biochemical systems and state the assumptions of each method. Also, in the same chapter, we describe GGM and our alternative solutions in inference of the L_1-penalized least absolute shrinkage and selection operator (Lasso) regression which is one of the promising methods in GGM. Moreover, we represent the possible choices with our proposal criterion for the model selection in GGM. Chapter 3 represents details related to our suggested new approach in singularity concept and our suggested optimization algorithms adapted in GGM. In Chapter 4, all the applications and comparisons based on the Monte-Carlo runs are given. Also, the description of the JAK/STAT pathway and corresponding simulated data as well as the brief description of the MAPK/ERK pathway and its available real time-course data are presented. Finally, we sum up our findings and discuss the outputs by suggesting our future works in Chapter 5. On the other hand, our original R functional codes are given in Appendix of the thesis.

CHAPTER 2

BACKGROUND

In this part we present the available literature about three distinct topics that the thesis is concentrated on. The first topic is the matrix decomposition which can be seen as the competitive approaches for our suggested matrix decomposition method and the second one is the existing techniques in modelling biological/biochemical systems as the alternative of GGM. Finally the third subsection presents the mathematical details about the model selection criteria that is used in junction with the inference of the different dimension networks via GGM.

2.1 Singularity and Matrix Decomposition

The high-dimensional variances's structures are commonly observed in many mathematical sciences from linear algebra and statistical calculations to financial and genomic data analysis. For instance in multivariate statistical analysis, the major concern is either the hypothesis testing or inference of population parameters that are based on mainly high-dimensional matrices where the covariances are typically used in the associated expressions (Kutner et al., 2005). Alternatively in microarray studies within the genomic data analysis we need to deal with the intensities from hundreds of genes simultaneously in which the mean and variance estimates of genes are described as high-dimensional matrices. Thereby, when working on such underlying high-dimensional covariance structures, we may face with a singularity problem if the distribution of the dataset is concentrated on a lower-dimensional subspace. Here the singularity problem is observed if there are linear relationships between columns or rows of the matrix, resulting in zero determinant in its calculation. This means that the inverse of the underlying matrix becomes intractable, which leads to a computational problem in the likelihood. For example if the covariance matrix of the data is singular and the distribution function of the data is expressed via this covariance such as the density of the multivariate normal distribution, the associated likelihood function becomes intractable under the singularity of this matrix.

There are a number of methods to unravel the underlying challenge. The most well-known ones can be listed as the eigenvalue decomposition, singular value decomposition and the Cholesky decomposition. These methods are explained in the following parts explicitly. Although there are a number of alternative approaches to unravel the singularity problem, none of them explicitly deals with the actual structure of the linear relationship between the states. In particular, in Bayesian inference, we may have partial observations of some states and the information about all the states should be used for inference about diffusion. In that case, simply focussing on a non-singular subspace of the states, may result in information loss. Instead, in this study, we suggest an alternative method for these major approaches to solve the singularity problem. It considers a lower-dimensional, nonsingular diffusion submatrix and explicitly models the linear relation between its dependent and independent terms. The mathematical details of our proposal method and its application are presented in the following chapters. Here, as stated previously, we merely describe the current and the most well-know matrix decomposition methods in the literature.

2.1.1 Eigenvalue Decomposition

Considering that **A** is a symmetric matrix, **U** denotes a matrix that includes the set of eigenvectors of **A**, and the diagonal matrix **Λ** stores the eigenvalues of **A** in the diagonal elements, we can write the following equality for every **A**, **U** and **Λ**.

$$\mathbf{AU} = \mathbf{U\Lambda}. \tag{2.1}$$

Hereby, the eigenvalue decomposition of **A** can be found by

$$\mathbf{A} = \mathbf{U\Lambda U^{-1}}. \tag{2.2}$$

Accordingly, the square root of **A** can be calculated by taking the square root of **Λ** via

$$\mathbf{A^{1/2}} = \mathbf{U\Lambda^{1/2}U^{-1}}. \tag{2.3}$$

If **A** is non-negative semi-definite, then the eigenvalue decomposition of this matrix always exists and the associated eigenvalues are always positive or zero. The reason can be explained as the following derivation:

Let $e_1, e_2, ..., e_k$ be normalized eigenvectors of **A**. Given a column vector **u**, we can write

$$\mathbf{u} = \alpha_1 e_1 + \alpha_2 e_2 + ... + \alpha_k e_k. \tag{2.4}$$

Thereby,

$$\mathbf{Au} = \alpha_1 \lambda_1 e_1 + \alpha_2 \lambda_2 e_2 + ... + \alpha_k \lambda_k e_k, \tag{2.5}$$

where λ_i associates to e_i. Then, since $e_i^T e_j = 0$ for $i \neq j$ and $e_i^T e_j = 1$ for $i = j$, the following equation can be written as

$$\mathbf{u^T Au} = \alpha_1^2 \lambda_1 + \alpha_2^2 \lambda_2 + ... + \alpha_k^2 \lambda_k. \tag{2.6}$$

Hence, the quantity on the left-handside in Equation (2.6) is nonnegative if all λ_i's are nonnegative.

The correlation, covariance, and cross-product matrices can be given as examples of such types of matrices (Healy, 1986). In this decomposition, the structure of the matrix can disappear if the original matrix is singular and we need to convert this matrix into nonsingular form by reducing the dimension (Johnson and Wichern, 2002; Rencher, 2002).

2.1.2 Singular Value Decomposition

Let **A** be an $(m \times n)$ rectangular matrix which can be written as a product of three matrices, namely, an $(m \times m)$ orthogonal matrix **U**, an $(m \times n)$ diagonal matrix $\mathbf{\Lambda}$ and the transpose of an $(n \times n)$ orthogonal matrix **V**. Here, $\mathbf{\Lambda}$ is a matrix containing singular values in diagonal elements with descending order.

Accordingly, the singular value decomposition (SVD) of **A** can be declared in terms of **U**, $\mathbf{\Lambda}$ and **V** by

$$\mathbf{A} = \mathbf{U}\mathbf{\Lambda}\mathbf{V}^{\mathbf{T}}, \tag{2.7}$$

where the columns of **U** and the columns of **V** are orthogonal eigenvectors of $\mathbf{A}\mathbf{A}^{\mathbf{T}}$ and $\mathbf{A}^{\mathbf{T}}\mathbf{A}$, respectively. Moreover, the columns of **U** are called the left singular vectors and the columns of **V** are called the right singular vector. Thus, the square root of **A** can be written via

$$\mathbf{A}^{\mathbf{1/2}} = \mathbf{U}\mathbf{\Lambda}^{1/2}\mathbf{V}^{\mathbf{T}}. \tag{2.8}$$

Similar to the eigenvalue decomposition, the structure of the original matrix **A** can change if **A** is reconstructed in order to convert the matrix from singular to nonsingular form (Johnson and Wichern, 2002; Rencher, 2002).

2.1.3 Cholesky Decomposition

The Cholesky decomposition method is based on the following theorem. Let **M** be any $(n \times n)$ symmetric, nonsingular matrix and $\mathbf{A} = \mathbf{M}^{\mathbf{T}}\mathbf{M}$, then **A** is positive definite matrix. This theorem implies that every positive definite matrix has the form $\mathbf{M}^{\mathbf{T}}\mathbf{M}$ for some **M**.

If **A** denotes a symmetric and positive definite matrix, the Cholesky decomposition of **A** can be found by an upper triangular matrix **U** having strictly positive diagonal entries such that

$$\mathbf{A} = \mathbf{U}^{\mathbf{T}}\mathbf{U}. \tag{2.9}$$

Here, the matrix **U** is called the *square root* of **A** and the matrix $\mathbf{U}^{\mathbf{T}}$ is called the *Cholesky factor* of **A**. It is unique since **U** has strictly positive diagonal entries. If **A** is also symmetric, then $\mathbf{U}^{\mathbf{T}} = \mathbf{U}$, hereby, $\mathbf{A} = \mathbf{U}\mathbf{U}$. But if **A** is positive semi-definite, i.e., some eigenvalues are zero, we use a numerical tolerance in the decomposition of **A**. In this approach, similar to its previous alternatives, it cannot preserve the original structure of the matrix when the singularity problem is solved by this decomposition and a new nonsingular matrix is defined under the original dimension of **A** (Johnson and Wichern, 2002; Rencher, 2002).

2.2 Modelling Biological Systems

The aim of the system biology is to understand the underlying mechanism of a living organism. An organism can be studied at several different levels from molecular to organs. Even if we work on a molecular level, there are many components that interact with each other. At the molecular level, we deal with which genes are expressed when and under which level.

Thereby, the gene regulatory systems give information about the interactions between DNA, RNA and proteins. The structure of interaction for these molecules is represented by networks. Along with the development of biotechnology in the last decade, there is an exponential increase in biological data. Hence, it is difficult but essential the analysis of the huge amount of data to understand the behavior of all components and interactions. Modelling is the key tool to overcome this challenge. In order to decrease the complexity in biological systems, mathematical models can be used without losing the important features of the system (Wilkinson, 2006).

There are different modelling approaches which implement different kinds of information to represent the state of a system (Bower and Bolouri, 2001; Wit et al., 2013; Purutçuoğlu and Ayyıldız, 2013). Hereby, the basic modelling approaches can be listed as

1. Boolean Network Models,
2. Differential Equation Models,
3. Stochastic Models.

We present the details of each approach in the following parts. Then, we explain the *Gaussian graphical modelling* (*GGM*) which is a sub-model under deterministic approach. But before giving details about GGM, we describe the Bayesian network and probabilistic boolean approach as well since they are the close alternatives of GGM. Then, we explain our strategy of estimation in GGM by giving the mathematical details. An application of the proposal methods is given in Chapter 4.

2.2.1 Boolean Network Models

The Boolean model is the most primitive model to explain the biochemical system. This model uses binary logic in such a way that the state of a model is represented as a list of fully expressed or not expressed genes by denoting 1 and 0, respectively. In this type of network, there is a finite number of states and a Boolean function which determines the next state. These attributes make the Boolean networks deterministic. Since the same components give the same results in deterministic systems, the next state can be identified completely in this model. Furthermore, its binary logic, i.e., 1 and 0 notations, enables us to reduce the noise of the data. Moreover, the Boolean network model has an assumption such that the transitions between genomic states occur synchronously (Whittaker, 1990; Purutçuoğlu and Ayyıldız, 2013).

There are two alternative ways to represent a system via Boolean models:

1. Truth table,
2. Finite-state machine.

In the truth table, the next states are defined by Boolean function which uses three Boolean operators, namely, AND, OR and NOT. The truth table of these Boolean operators that contains all possible combinations of variable *p* and *q* is presented in Table 2.1.

Table 2.1: The truth table of the Boolean operators with variable *p* and *q*.

p	**q**	**p AND q**	**p OR q**	**NOT q**
1	1	1	1	0
1	0	0	1	0
0	1	0	1	1
0	0	0	0	1

On the other hand, the finite-state machine indicates the system as a diagram made of circles and arrows which stand for states and transitions, respectively.

For instance, consider a network consisting of three genes *A, B, C*, and three Boolean functions to determine the next state of each gene as below although this network has $2^3 = 8$ possible states:

A(next) = *A(current)* AND *B(current)*,

B(next) = *A(current)* OR *B(current)*, *C(next)* = NOT *A(current)* OR *B(current)*,

where *A(next)*, *B(next)*, and *C(next)* denote the next state of the associated genes and *A(current)*, *B(current)* and *C(current)* stand for their current states. Hereby, a possible truth table of the Boolean network for the three genes can be generated as in Table 2.2.

The truth table shows that if none of the genes is expressed in current state, C will be expressed in the next state as shown in the first item. On the other hand, as stated previously, a finite-state

Table 2.2: The truth table of the Boolean network for the three genes *A, B, C*.

Current state	**Next state**
000	001
001	011
010	001
011	011
100	010
101	010
110	111
111	111

machine is another way to represent the changes of state. For a given example, the finite-state machine is shown in Figure 2.1.

Since using the binary logic simplifies the data structure, the large regulatory networks can be analyzed via Boolean network model. However, most of the time more complex models which cover the different gene expression level are required.

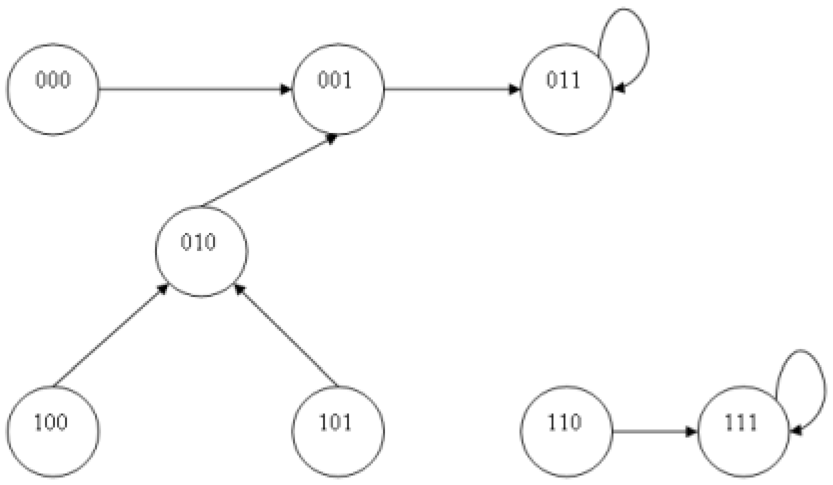

Figure 2.1: Finite-state machine for the Boolean network.

2.2.2 Differential Equation Models

The chemical reactions are the basic parts of the biological modelling. To construct a mathematical model, we need to convert qualitative data into quantitative form. Since the chemical reactions use unique notations to represent complex chemical processes, mathematical models can be applied to analyze the biological structure.

A general chemical reaction can be simply represented via

$$n_A A \xrightarrow{k} n_{A'} A'. \tag{2.10}$$

In Equation (2.10) the molecules which are on the left side of the arrow are called *reactants* and the molecules which are on the right side are called *products*. Also the direction of the arrow gives information about the direction of the process. For instance, in the given reaction, n_A amount of molecules of A is transformed to $n_{A'}$ amount of molecules of A'. The n terms in this expression are called *stoichiometric coefficients*. Moreover, the k value on the reaction arrow is named as the *rate constant*. This value is used to determine the necessary time for the completion or equilibrium of the reaction.

In differential equation models, the concentrations of each chemical type are applied to describe the state of the gene regulation processes. These concentrations are continuous and time-dependent, i.e., they change over time with the rate constant of chemical reactions. In this representation the rates can be calculated according to the ordinary differential equations (ODE's) which can capture the deterministic, i.e., steady state, behavior of the system. For instance, if we consider the following chemical equation in terms of ODE,

$$A \xrightarrow{k} A',$$

the rate of this reaction is expressed as $k[A]$. Here the increase in the concentration of A, i.e., $[A]$, results in a linear increase in the rate of production of A' and the concentration of the reactant decreases while its product increases as the reaction occurs. Therefore, the change in A and A' over time can be displayed as

$$\frac{d[A]}{dt} = -k[A] \quad \text{and} \quad \frac{d[A']}{dt} = k[A]. \tag{2.11}$$

From Equation (2.11), in order to infer the model parameters, which are the rate of laws, for a given set of concentrations, the linear ODE's are solved simultaneously. But when the stoichiometric coefficients increase, ODE's become non-linear equations, resulting in intractable simultaneous equations. Under such conditions, different approximation methods can be implemented and this is one of the major disadvantages of ODE's since the solution set is not unique (Bower and Bolouri, 2001; Purutçuoğlu and Ayyıldız, 2013). Moreover, as ODE's are deterministic, they are not suitable for systems which have more than one possible outcome for the next state.

2.2.3 Stochastic Models

In stochastic models, the level of details applied in modelling is more than the ones for ODE and Boolean approaches in the sense that the description of the model is based on the exact numbers of molecules of each gene type. Hereby, in this model, the changes in the states are described with discrete quantity. Whereas, which changes occur and when they occur are explained probabilistically. That means even if a system has the same initial state, different next states can be produced. Here the probability per unit time of the discrete event, i.e., reaction, is indicated by the *reaction rate constant* which can also give us information about the speed of the reaction. Under this modelling it is accepted that the biological systems have naturally stochastic properties and ODE's become insufficient to explain such randomness (Wilkinson, 2006; Purutçuoğlu and Ayyıldız, 2013).

The stochastic modelling is based on the chemical master equation which is explained in the following part in detail.

2.2.3.1 Chemical Master Equation

When the states represent the number of molecules, the probabilities of being in these states are used as variables in the chemical master equation (CME) approach. Since these variables are function of time, the differential equations can be used to identify the change in probabilities.

Accordingly, in this equation if X indicates the number of molecules, a joint probability distribution $p(X, t)$ can be defined as the probability of being at a state X at time t by

$$p(X, t + \triangle t) = p(X, t)\left(1 - \sum_{j=1}^{m} \alpha_j \triangle t\right) + \sum_{j=1}^{m} \beta_j \triangle t. \tag{2.12}$$

In this equation, m is the number of reactions that occurs in the system. The probability that the reaction j will occur in the interval $[t, t + \triangle t]$ given that the system is in the state X at t is denoted by $\alpha_j \triangle t$. Also the term $\beta_j \triangle t$ indicates the probability that the reaction j will bring the system in state X from another state in $[t, t + \triangle t]$ (Purutçuoğlu and Ayyıldız, 2013).

Thus, the Chemical Master Equation describes the behavior of X by the differential equation as below (Gillespie, 2001; Kampen, 2007; Purutçuoğlu, 2007):

$$\frac{\partial}{\partial t} p(X, t) = \sum_{j=1}^{m} (\beta_j - \alpha_j p(X, t)).$$

CME is useful only when the number of possible states is small. If the number of states increases, the complexity of differential equation system raises rapidly. Hence, solving CME is very difficult both analytically and numerically, except for very small systems. Thus, CME becomes inapplicable for the inference of model parameters. Thereby, under realistically complex system, certain approximation methods such diffusion approximation as given in Equation, or Euler-Maruyama approximation (Purutçuoğlu, 2012) can be implemented. The major idea of all these alternatives is to solve the approximation of CME via certain Monte-Carlo based computational methods (Golightly and Wilkinson, 2005; Wilkinson, 2006; Purutçuoğlu, 2012, Purutçuoğlu and Wit, 2012).

2.2.4 Bayesian Networks

The Gene Regulatory Networks (GRN) have fundamentally stochastic behaviors. Although the ODE and Boolean network models reduce the number of model parameters, they are deterministic. So, they cannot deal with the stochastic behavior of GRNs. The Bayesian Network (BN) modeling is an alternative method to overcome this challenge, since it generates a probabilistic network (Shmulevich and Dougherty, 2010).

The BN's are based on the conditional probability theory since the calculation dependent on the conditional probabilities is more convenient when we deal with the gene expression. Because the probability that a gene having a particular expression level given the activity levels of its potential regulators serves us more information than the absolute probability of a gene expressed at some level. For instance, let's say that when genes A and B are highly expressed and gene C is off, gene D will be highly expressed. This information suggests that genes A and B are activators and gene C is an inhibitor of the gene D expression.

Moreover, the BN's are event-based models in the sense that they use the relationship between the input values and the output states. Hence, a large amount of data is necessary to perform the BN based modeling.

In BN's, the directed acyclic graphs (DAG), as shown in Figure 2.2, are used to represent conditional dependencies or independencies. The nodes, also called *vertices* of DAG, stand for random variables $\mathbf{X} = (X_1, ..., X_n)^T$ and the directed edges display the dependency structure. For every node in DAG , there may be multiple inputs which come from parent nodes and multiple outputs to descendant nodes. Furthermore, the DAG theory uses the Markov assumption saying that when its parents are given, each variable X_i is independent on its non-descendants. Hereby, via the agency of the chain rule of probabilities, any joint distribution can be represented in terms of a product of the local conditional probabilities such that

$$P\{x_1, ..., x_n\} = \prod_{i=1}^{n} P\{x_i | Pa(x_i)\},$$

where $Pa(x_i)$ denotes the Markovian parents of X_i.

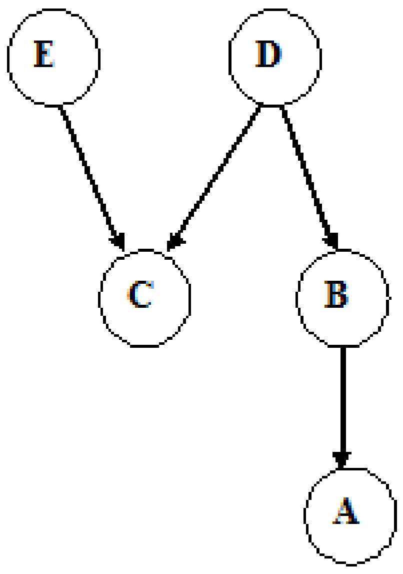

Figure 2.2: Simple representation of the Bayesian Network

For instance, consider a Bayesian network consisting of five nodes as drawn in Figure 2.2. The parents of C are D and E. Also B is only nondescendant of C. Hence, C is independent on B given both D and E. This relation can be described in terms of probabilities as follows:

$$P\{C|B, D, E\} = P\{C|D, E\}.$$

Accordingly, the joint distribution of these five variables can be represented as

$$P\{A, B, C, D, E\} = P\{A|B\}P\{B|D\}P\{C|D, E\}P\{D\}P\{E\}.$$

If we have full data on A, C, D and E, C is called *hidden* node and the behavior of C can be estimated. Moreover, it is not necessary to know D to predict A as long as B is known. Consequently, a Bayesian network gives us an opportunity to express a joint probability distribution with more understandable way. Another advantage of BN's is that in order to build a model, experimental data can be used directly. Furthermore, the uncertainties that come from data are handled by probabilistic properties of BNs. There are many software tools, e.g., Hugin and Bayesian packages in R programming language, to deal with the real life BN models which contain large number of nodes (genes) (Bolouri, 2008). On the other hand, a disadvantage of BN is that it represents a static probabilistic relationship of network. It means that if $X \rightarrow Y$

and $Y \rightarrow X$, according to BN these two networks are equivalent. In addition, feedback networks cannot be represented by using DAGs. This type of networks include information about sequential, i.e., dynamic, behavior of nodes. Hence, dynamic Bayesian networks (DBNs) are used to model the conditional dependencies between nodes over time. So, the time-course data are required to separate the sequences of events (Jong, 2002; Bolouri, 2008).

2.2.5 Probabilistic Boolean Networks

The Probabilistic Boolean network (PBN) is a stochastic generalization of the Boolean networks which enables us to understand the dynamical behavior of biological networks. Indeed, the theory of Markov chains can be used to describe the biological system dynamics. In a first order homogeneous Markov chain, a state at time $t + 1$ depends on only states at time t. Similarly, the expression on the state for each gene at step $t + 1$ is determined by the expression levels of genes at step t.

On the other hand, the *Boolean* term in PBN does not mean the binary quantization like in the Boolean networks. Here, there is a finite quantization in the sense that the components of PBN can be defined and formalized such as a sequence $V = \{x_i\}_{i=1}^{n}$ of n nodes and a sequence $\{f_l\}_{l=1}^{m}$ of vector-valued functions. Hereby, the expression value of a gene is denoted by x_i, where $x_i \in \{0, ..., d - 1\}$, and d represents the number of quantization level. Moreover, a context, or constituent network, of the PBN is determined by each vector-valued function $f_l = (f_l^{(1)}, f_l^{(2)}, ..., f_l^{(n)})$. Also, when the network l selected, the function $f_l^{(i)}$ is used as a predictor of gene i.

Switching the constituent network is determined by a binary random variable ξ. $\xi = 0$ indicates that the current context is continued, whereas, $\xi = 1$ shows that using the selection probability distribution $\{c_l\}_{l=1}^{m}$, a new constituent network is randomly selected. Since the selection is made from all constituent networks, the same context can be selected. It means that the new context does not have to be different from the current one.

In addition to binary random variable, a random perturbation also causes to switch the constituent networks. The external stimulus such as the stress condition or small molecule inhibitors can be given as examples of perturbation for biological systems. In the case of PBN with perturbation, there exists a perturbation probability $P(\gamma_i = 1) = p$ and a perturbation vector $\boldsymbol{\gamma} = (\gamma_1, \gamma_2, ..., \gamma_n)$ where $\gamma_i \in \{0, 1\}$. If the constituent network is determined just by the current network function f_l, the probability of no perturbation equals to $(1 - p)^n$. On the other hand, the probability of a perturbation equals to $1 - (1 - p)^n$. In general, including a perturbation probability in PBN gives more realistic results when we deal with the activity of a gene (Whittaker, 1990; Shmulevich and Dougherty, 2010).

2.2.5.1 State Transition Probabilities

A state transition probability P_{ij} is the probability of passing from one state to another. It is denoted by

$$P_{ij} = Pr\{X_n = j\,|X_{n-1} = i\}. \tag{2.13}$$

As an example, the following state transition matrix belongs to a very small system that contains two states,

$$\begin{array}{cc} & \begin{array}{cc} 1 & 2 \end{array} \\ \begin{array}{c} 1 \\ 2 \end{array} & \begin{pmatrix} 0.3 & 0.7 \\ 0.6 & 0.4 \end{pmatrix}. \end{array}$$

Here, the probability 0.07 indicates the state transition probability P_{12}. It means that if the system is in state 1 currently, the probability of passing state 2 equals to 0.7.

If the state transition probabilities can be computed, the state transition matrix can be also found and the steady-state distribution of the model can be determined. Consequently, the long-run behavior of the model can be discovered. This information gives us an opportunity to answer the questions such as *What is the probability that the gene X will be expressed in the long run?* or *What is the probability that gene Y and gene Z will both be expressed in the long run?* (Shmulevich and Dougherty, 2010).

In order to define the long-run behavior of a Markov chain, the periodicity is an important property. This feature means that the period of a state i equals to k when any return to state i occurs in k time step. An irreducible Markov chain, which refer to the communication of all states with each other, can be classified whether it is a periodic or aperiodic (Gallager, 1996).

A Markov chain has a stationary distribution if a probability distribution $\Pi = (\Pi_1, \Pi_2, ..., \Pi_M)$ satisfies the following equality:

$$\Pi_j = \sum_{i=1}^{M} \Pi_i P^r_{ij},$$

where P^r_{ij} indicates the r-step transition probability and j is an element of the finite state space, $S = \{1, 2, ..., M\}$. Furthermore, the Markov chain has a steady-state distribution if a probability distribution $\Pi = (\Pi_1, \Pi_2, ..., \Pi_M)$ holds the equation below:

$$\lim_{r\to\infty} P^r_{ij} = \Pi_j. \tag{2.14}$$

Equation (2.14) means that no matter what is the starting state, in the long run the probability of the Markov chain being in state i is Π_i if a steady-state distribution exists (Shmulevich and Dougherty, 2010; Purutçuoğlu and Ayyıldız, 2013).

2.2.5.2 Steady-State Analysis of PBN

When the size of the network increases, computing the whole state transition matrix of Markov chain becomes difficult since the number of the state space grows exponentially. Under such challenge, the state transition matrix can be computed by different methods, rather than computing each entry one by one. First method uses the characteristic of the state transition matrix which consists of two parts. A matrix of the state transitions of the constituent Boolean networks creates one part. Other part is a matrix that belongs to perturbations. Computing the perturbation matrix only once is enough due to the fact that it depends on merely the number and the perturbation probability p of genes. Therefore it stays the same for different networks.

Another method deals with the state transition matrix. Since it is a sparse matrix in practice, that is, it contains many zero entries, dealing with only the nonzero entries decreases the computational complexity.

The other method uses the information that the selection probabilities of many constituent Boolean networks are very low. Hereby, if we define a threshold value, considering only the Boolean networks which have greater probabilities than this threshold value, makes the computation easier.

2.2.5.3 Steady-State Analysis via Simulation

It is known that the steady-state distribution can be computed from the state transition matrix of Markov chain. Unfortunately, sometimes this calculation becomes intractable because of the exponential growth of the dimension in the state transition matrix. The Monte-Carlo simulation techniques give an efficient solution for these situations. The procedure of the Monte-Carlo method to determine the steady-state probabilities can be explained into the two parts. Firstly, until the convergence to the stationary distribution is achieved, the Markov chain runs for a long time. Then the proportion of time that the process spends in which parts of the state space is calculated. In order to rely on the results of Monte-Carlo runs, we need to estimate the convergence rate of the process. It means that we need to guarantee that the chain reaches its stationary distribution actually (Ross, 1996).

2.2.6 Gaussian Graphical Models

In a biological system, the interactions between components can be represented by using graphical models. The graphical models consist of a set of nodes which is totally p, and a set of edges. The nodes represent the components, such as proteins and genes, in the system and the edges display the interaction among these components. The nodes can be formalized as a vector $\boldsymbol{Y} = (Y^{(1)}, ..., Y^{(p)})$.

The graphical models can be divided into two groups, namely, directed and undirected graphical models. In directed graphical model, the edges represent both interactions and their directions. Whereas, in undirected graphical model, the edges denotes only interactions but not give information about their directions. A most widely used undirected graphical model is a Gaussian graphical model (GGM). This model makes the assumption that the vector **Y** has a multivariate Gaussian (or Normal) distribution, that is,

$$\boldsymbol{Y} \sim N(\boldsymbol{\mu}, \boldsymbol{\Sigma}), \tag{2.15}$$

where $\boldsymbol{\mu} = (\mu_1, ..., \mu_p)$ denotes the mean, $\boldsymbol{\Sigma}$ shows the $(p \times p)$ variance-covariance matrix whose entries σ_{ij} present the covariances of $Y^{(i)}$ and $Y^{(j)}$.

Many natural phenomena have the multivariate normal distribution. Since the multivariate normal density is generalization of the univariate normal density, it is mathematically tractable. A p dimensional normal density for the vector $\boldsymbol{Y}$ has the following form:

$$f(y) = \frac{1}{(2\pi)^{n/2}|\boldsymbol{\Sigma}|^{1/2}} e^{\{-\frac{1}{2}(y-\mu)^T \boldsymbol{\Sigma}^{-1}(y-\mu)\}}, \quad -\infty < y < \infty. \tag{2.16}$$

In here, $\boldsymbol{\Sigma}$ is a symmetric and positive definite matrix. So, it is invertible. The multivariate normal distribution has several properties. The key properties for a random vector $\boldsymbol{Y}$ can be listed such as (Johnson and Wichern, 2002):

- Linear combinations of the components of $\boldsymbol{Y}$ are normally distributed.
- All subsets of the components have normal distribution.
- If there is zero covariance between components, it means that they are independently distributed.
- The conditional distributions of the components are normal.

In GGM, the conditional independencies of two nodes are indicated with the absence of an edge between these nodes and are represented with zero entries in the precision matrix, under the normality assumption. A *precision* or *concentration matrix* is an inverse of the variance-covariance matrix $\mathbf{\Sigma}$ and is denoted by $\mathbf{\Theta}$ via

$$\mathbf{\Theta} = \mathbf{\Sigma}^{-1} = \theta_{ij}. \tag{2.17}$$

Also the precision matrix is composed of partial covariances. This means that the inverse of the partial variances constitutes the diagonal entries of the precision matrix, such as $\theta_{ii} = 1/var(Y^{(i)} \mid rest)$, and that minus the partial correlation constitutes the scaled off-diagonal entries.

$$\pi_{ij} = \frac{-\theta_{ij}}{\sqrt{\theta_{ii}\theta_{jj}}}, \tag{2.18}$$

where π_{ij} represents the partial correlation of $Y^{(i)}$ and $Y^{(j)}$ when all the other variables are given. The networks that are generated by using partial correlations are called the *gene association network*. Since the genomic data generally have a large number of variables, p, and few samples, n, i.e., $n \ll p$, the application of standard covariance and correlation is not appropriate. Because in *small n, large p* data, the sample covariance estimator $\boldsymbol{S}$ turns a singular matrix, i.e., non-invertible, because of large number of zero eigenvalues. Thus the estimation of positive definited and well-conditioned covariance matrix is a crucial problem.

There are different ways to derive partial correlations of a network. One of these methods is to apply the regression method in such a way that the regression functions for each node against all remaining nodes give the conditional independence structure of the model.

When we divide the Gaussian vector $\boldsymbol{Y}$ into two parts such as $\boldsymbol{Y} = (\boldsymbol{Y}^{(-p)}, Y^{(p)}))$, in which $\boldsymbol{Y}^{(-p)} = (Y^{(1)}, ..., Y^{(p-1)})$ denotes the vector of all nodes except for the last one, we can obtain the conditional distribution of node $Y^{(p)}$ on all the remaining nodes as

$$Y^{(p)} \mid \boldsymbol{Y}^{(-p)} = y \sim N(\mu_p + (\boldsymbol{y} - \boldsymbol{\mu}_{-p})^t \boldsymbol{\Sigma}^{-1}_{-p,-p} \boldsymbol{\sigma}_{-p,p}, \sigma_{p,p} - \boldsymbol{\sigma}^t_{-p,p} \boldsymbol{\Sigma}^{-1}_{-p,-p} \boldsymbol{\sigma}_{-p,p}). \tag{2.19}$$

Here $(\boldsymbol{A})^t$ denotes the transpose of $\boldsymbol{A}$. Accordingly, the mean $\boldsymbol{\mu}$ and covariance $\mathbf{\Sigma}$ can be partitioned such that

$$\boldsymbol{\mu} = \begin{pmatrix} \boldsymbol{\mu}_{-p} \\ \mu_p \end{pmatrix} \text{ and } \mathbf{\Sigma} = \begin{pmatrix} \mathbf{\Sigma}_{-p,-p} & \boldsymbol{\sigma}_{-p,p} \\ \boldsymbol{\sigma}^t_{-p,p} & \sigma_{p,p} \end{pmatrix}, \tag{2.20}$$

where $\boldsymbol{\mu}_{-p}$ stands for the mean vector of all nodes except for the last one and μ_p indicates the mean of the last node. Also $\mathbf{\Sigma}_{-p,-p}$ is the $((p-1) \times (p-1))$ variance-covariance matrix of all nodes except for the last one. $\boldsymbol{\sigma_{-p,p}}$ refers to the $((p-1)\times 1)$ covariance vector of $\boldsymbol{Y^{(-p)}}$ and $Y^{(p)}$ and lastly $\sigma_{p,p}$ is the variance of the last node $Y^{(p)}$. Thereby, the conditional dependence structure is identified by the regression coefficients $\beta = \Sigma^{-1}_{-p,-p}\sigma_{-p,p}$. We can say that $Y^{(p)}$ and $Y^{(j)}$ $(j = 1, ..., p-1)$ are *conditionally independent* when $\beta_j = 0$. In addition there is a relation between the coefficients β and the precision matrix that can be formalized by (Wit et al., 2013),

$$\beta = -\theta_{-p,p}/\theta_{p,p}. \tag{2.21}$$

To measure the strength of the interaction between two entries under the known structure, that is for the given edges, the precision matrix $\boldsymbol{\Theta}$ is estimated. In inference of $\boldsymbol{\theta}$ we can use the maximum likelihood estimation technique. Accordingly, if $f(y_i)$ denotes the joint density function of observation y for the vector $\boldsymbol{Y}$,

$$f(y_i;\boldsymbol{\mu},\boldsymbol{\Sigma}) = (2\pi)^{-n/2}|\boldsymbol{\Sigma}|^{1/2}\exp\left\{-\frac{1}{2}(\boldsymbol{y_i}-\boldsymbol{\mu})^T\boldsymbol{\Sigma^{-1}}(\boldsymbol{y_i}-\boldsymbol{\mu})\right\}, \tag{2.22}$$

the likelihood function $L(\boldsymbol{\mu},\boldsymbol{\Sigma})$ can be formalized by the multiplication of each joint density function such that

$$L(\boldsymbol{\mu},\boldsymbol{\Sigma}) = \prod_{i=1}^{n} f(y_i;\boldsymbol{\mu},\boldsymbol{\Sigma}). \tag{2.23}$$

Then, the log-likelihood function becomes

$$l(\boldsymbol{\mu},\boldsymbol{\Sigma}) = \log(L(\boldsymbol{\mu},\boldsymbol{\Sigma})) = -\frac{n}{2}\log|\boldsymbol{\Sigma}| - \frac{1}{2}\Sigma_{i=1}^{n}(\boldsymbol{y_i}-\boldsymbol{\mu})^T\boldsymbol{\Sigma^{-1}}(\boldsymbol{y_i}-\boldsymbol{\mu}). \tag{2.24}$$

When we replace the variance-covariance matrix by the precision matrix, we obtain the following equation:

$$l(\boldsymbol{\mu},\boldsymbol{\Theta}) = \frac{n}{2}\log|\boldsymbol{\Theta}| - \frac{1}{2}\Sigma_{i=1}^{n}(\boldsymbol{y_i}-\boldsymbol{\mu})^T\boldsymbol{\Theta}(\boldsymbol{y_i}-\boldsymbol{\mu}). \tag{2.25}$$

In Equation (2.25), if we set $\boldsymbol{\mu}$ to $\bar{\boldsymbol{y}}$ as the maximum likelihood estimator of $\boldsymbol{\mu}$,

$$l(\boldsymbol{\mu},\boldsymbol{\Theta}) \;=\; \frac{n}{2}\log|\boldsymbol{\Theta}| - \frac{1}{2}\Sigma_{i=1}^{n}(\boldsymbol{y_i}-\bar{\boldsymbol{y}})^T\boldsymbol{\Theta}(\boldsymbol{y_i}-\bar{\boldsymbol{y}}). \tag{2.26}$$

We can rewrite,

$$\Sigma_{i=1}^{n}(\boldsymbol{y_i}-\bar{\boldsymbol{y}})^T\boldsymbol{\Theta}(\boldsymbol{y_i}-\bar{\boldsymbol{y}}) = \Sigma_{i=1}^{n}\text{Trace}\left[(\boldsymbol{y_i}-\bar{\boldsymbol{y}})^T\boldsymbol{\Theta}(\boldsymbol{y_i}-\bar{\boldsymbol{y}})\right]. \tag{2.27}$$

Since,

$$\begin{aligned}\Sigma_{i=1}^{n}(\boldsymbol{y_i}-\bar{\boldsymbol{y}})^T\boldsymbol{\Theta}(\boldsymbol{y_i}-\bar{\boldsymbol{y}}) &= \Sigma_{i=1}^{n}\text{Trace}\left[(\boldsymbol{y_i}-\bar{\boldsymbol{y}})^T\boldsymbol{\Theta}(\boldsymbol{y_i}-\bar{\boldsymbol{y}})\right],\\ &= \Sigma_{i=1}^{n}\text{Trace}\left[\boldsymbol{\Theta}(\boldsymbol{y_i}-\bar{\boldsymbol{y}})^T(\boldsymbol{y_i}-\bar{\boldsymbol{y}})\right],\\ &= \text{Trace}\left[\boldsymbol{\Theta}\left(\Sigma_{i=1}^{n}(\boldsymbol{y_i}-\bar{\boldsymbol{y}})^T(\boldsymbol{y_i}-\bar{\boldsymbol{y}})\right)\right].\end{aligned}$$

Finally, the log-likelihood function of $\boldsymbol{\Theta}$ can be expressed as

$$l(\boldsymbol{\Theta}) \quad = \quad \frac{n}{2}\log|\boldsymbol{\Theta}| - \frac{n}{2}\text{Trace}(\boldsymbol{S}\boldsymbol{\Theta}), \tag{2.28}$$

where $\boldsymbol{S} = s_{ij}$ denotes the following sample covariance matrix

$$s_{ij} = \frac{1}{n}\Sigma_{k=1}^{n}(y_k^{(i)} - \bar{y}^{(i)})(y_k^{(j)} - \bar{y}^{(j)}). \tag{2.29}$$

Now, by maximizing the log-likelihood $l(\boldsymbol{\Theta})$ under the zero constraints, the precision matrix can be calculated. The maximum likelihood estimator of the precision matrix equals to $\hat{\boldsymbol{\Theta}} = \boldsymbol{S^{-1}}$. This matrix can be also converted into the partial correlation matrix by Equation (2.18) and the absolute value of its entries indicates the strength of interactions.

2.2.6.1 Maximum Likelihood Approach

The major aim of the GGM approach, as most of the statistical questions of interest, is to estimate the model parameters and thereby, define a structure for the selected biological networks for us. A network can be inferred by maximizing the likelihood of observed data. As we mentioned before, in a GGM, the precision matrix $\boldsymbol{\Theta}$ is estimated by the inverse of the sample covariance matrix which maximizes the log-likelihood function since $\boldsymbol{\Theta}$ capture the information of both the direction and strength of the interactions between the genes in the system. Then, the partial correlations can be obtained from this estimated matrix. Finally, to decide whether the partial correlations are significantly different from zero, statistical tests can be applied. There are different alternatives for this purpose. When the true partial correlation is zero, the estimated partial correlation is distributed as follows:

$$f(r,k) = (1-r^2)^{(k-3)/2}\frac{\Gamma(k/2)}{\sqrt{\pi}\Gamma((k1)/2)}, \tag{2.30}$$

where $k = n - p - 1$ denotes the degrees of freedom, n is the sample size, and p displays the number of variables. Moreover, $\Gamma(\cdot)$ refers to the Gamma function. This distribution can be used to test the significance of partial correlations different approaches such as the likelihood ratio or Wald statistics.

The z-transformation given below is another alternative approach to check the validity of the partial correlation (Wit et al., 2013):

$$z(r) = \frac{1}{2}\sqrt{n-p-1}\ln(\frac{1+r}{1-r}). \tag{2.31}$$

Similarly, the following likelihood ratio test can be applicable:

$$LR(r) = -n\log(1-r^2), \tag{2.32}$$

in which $LR(r)$ has an asymptotic χ_1^2 distribution under the null hypothesis of zero partial correlation.

In general, although the MLE method is successful when the full data are available and $\boldsymbol{\Theta}$ is nonsingular, it cannot be applicable for non-invertible $\boldsymbol{\Theta}$ (Whittaker, 1990).

2.2.6.2 Shrinkage Approach

The shrinkage method is an approach to estimate the covariance matrix by using not only the sample covariance but also a target matrix. Thereby, it estimates the covariance $\boldsymbol{\Sigma}$ with the following formula:

$$\boldsymbol{S}^* = \lambda \boldsymbol{T} + (1 - \lambda)\boldsymbol{S_u}, \tag{2.33}$$

where $\boldsymbol{S_u}$ is the unbiased sample covariance ($\boldsymbol{S_u} = \frac{n}{n-1}\boldsymbol{S}$), $\boldsymbol{T}$ is a target matrix that includes smaller number of parameters, and $\lambda \in [0, 1]$ is the shrinkage intensity. Although $\boldsymbol{S_u}$ is an unbiased estimator, it has a large variance. On the other hand, $\boldsymbol{T}$ has a lower variance but higher bias. The shrinkage approach uses these two estimators with a convenient trade-off intensity. This new shrinkage estimator has a lower mean square error then the sample covariance (Wit et al., 2013; Purutçuoğlu and Ayyıldız, 2013).

Accordingly, in this method, in order to find a good estimator to choose the target $\boldsymbol{T}$ and the shrinkage intensity λ, a diagonal scalar matrix is suggested. The *diagonal, unit variance* or *diagonal, common variance* or *diagonal, unequal variance* are the examples of target matrices (Strimmer and Sch*ä*fer, 2005). The last one is widely preferred in genomic problems as it can automatically generates a positive definite shrinkage covariance estimates. Furthermore, the value of λ which minimizes the mean square error can be used as a shrinkage intensity. Hence, the mean square error loss function can be formalized as

$$R(\lambda) = E[\Sigma_{i \neq j}[(1 - \lambda)\boldsymbol{S}_{\boldsymbol{u}_{ij}} - \sigma_{ij}]^2] + E[\Sigma_i(\boldsymbol{S}_{\boldsymbol{u}_{ii}} - \sigma_{ii})^2], \tag{2.34}$$

where $\boldsymbol{S}_{\boldsymbol{u}_{ij}}$ denotes the unbiased sample covariance.

Accordingly, the value of λ to minimize Equation (2.34) is given by

$$\lambda = \frac{\Sigma_{i \neq j} V(s_{u_{ij}})}{\Sigma_{i \neq j} E[s^2_{u_{ij}}]}. \tag{2.35}$$

2.2.6.3 Lasso-Based Approach

The Lasso-based approach is mainly used to estimate a sparse network, i.e., the network has small number of edges, resulting in many zero entries in the adjacency matrix. This matrix shows which nodes are adjacent to which other nodes. It uses binary representation and the entry 1 indicates that corresponding two nodes are adjacent. Because the sparsity is one of the general features in biological networks.

In this approach, the precision matrix is estimated by using a regression-based approach. The networks that are inferred with this method is called the *dependency network*. Although the regression-based approach has important advantages, such as computational efficiency and good approximations to the joint distribution of the variables, it does not guarantee a symmetric variance-covariance matrix.

The sparse graphical models can be obtained by implementing the sparse regression model which searches the sparsity of each node one by one. Hereby, for the given all other variables $\boldsymbol{Y}^{(-p)}$, a regression model for the last node $\boldsymbol{Y}^{(p)}$ is found via

$$\boldsymbol{Y}^{(p)} = \boldsymbol{Y}^{(-p)}\boldsymbol{\beta} + \boldsymbol{\varepsilon}, \tag{2.36}$$

where $\boldsymbol{\varepsilon}$ denotes the independent and normally distributed error.

In standard regression models, a least-square criterion is applied to estimate the coefficients $\boldsymbol{\beta}$. On the other hand, a Lasso-regression model computes the L_1-penalty for $\boldsymbol{\beta}$ such that $||\boldsymbol{\beta}||_1 = \Sigma_i|\beta_{ip}| < \lambda$. Through this difference, the sparsity of the precision matrix can be obtained and the solution is computed by

$$\min_{\beta} \; [||\boldsymbol{Y}^{(p)} - \boldsymbol{Y}^{(-p)}\boldsymbol{\beta}||_2^2 + \lambda_p||\boldsymbol{\beta}||_1] \tag{2.37}$$

with tuning parameter λ_p that enables us to estimate of the parameters $\boldsymbol{\beta}$. The large value of λ_p corresponds to zero coefficients. In Figure 2.3, it can be seen that when the λ_p value is increase, the corresponding network become sparse (Wit et al., 2013; Purutçuoğlu and Ayyıldız, 2013). Since this approach gives exact zero coefficients, there is no need to apply statistical test as implemented in the shrinkage approach. Conversely, non-symmetric results are the main problems of this approach. It means that although $Y^{(j)}$ from the rest can be obtained for zero β_{ij}, β_{ji} is not zero when we predict $Y^{(i)}$ from the rest. In Equation (2.37), $||\cdot||_1$ refers to L_1-norm which means the sum of the absolute values of the columns (Whittaker, 1990; Witten et al., 2011).

One solution to unravel this challenge is to apply AND or OR rules. If we use AND rule, we obtain the zero coefficients when both β_{ij} and β_{ji} are zero. Furthermore, if we use OR rule for one of β_{ij}'s zero is enough to obtain zero coefficients. So, it is seen that the OR rule generates more sparse networks. However, we cannot calculate the actual strength of the edge since these are two different values because of asymmetry.

In the Lasso-regression approach, another important issue is the selection of the penalty parameters λ_i. To get a false positive less than α, the following penalty equation can be calculated.

$$\lambda_i = 2\sqrt{\frac{s_{ii}}{n}}\Phi^{-1}\left(1 - \frac{\alpha}{2p^2}\right), \tag{2.38}$$

where Φ denotes the cumulative distribution function of the standard normal.

The penalized likelihood idea arises because that ordinal least squares (OLS) estimates have problems in high-dimensional data. The prediction accuracy and interpretation are two main problems of OLS. Since, generally the OLS estimates have low bias but large variance, by setting regression coefficients exactly zero, the precision accuracy increases (Tibshirani, 1996). Moreover, if the number of regression coefficients is greater than the observations, it is difficult to obtain an interpretable model. L_1 and L_2 penalties are used to overcome these challenges. L_1 absolute value penalty and L_2 quadratic (ridge) penalty shrink the coefficients towards zero. However, there is a difference between these two penalized estimation method. The regression coefficients are small but non-zero in the L_2 penalized method. On the other side, the L_1 penalized regression sets the coefficients exactly to zero. This feature causes to obtain an interpretable results in the L_1 penalized regression method (Goeman, 2010).

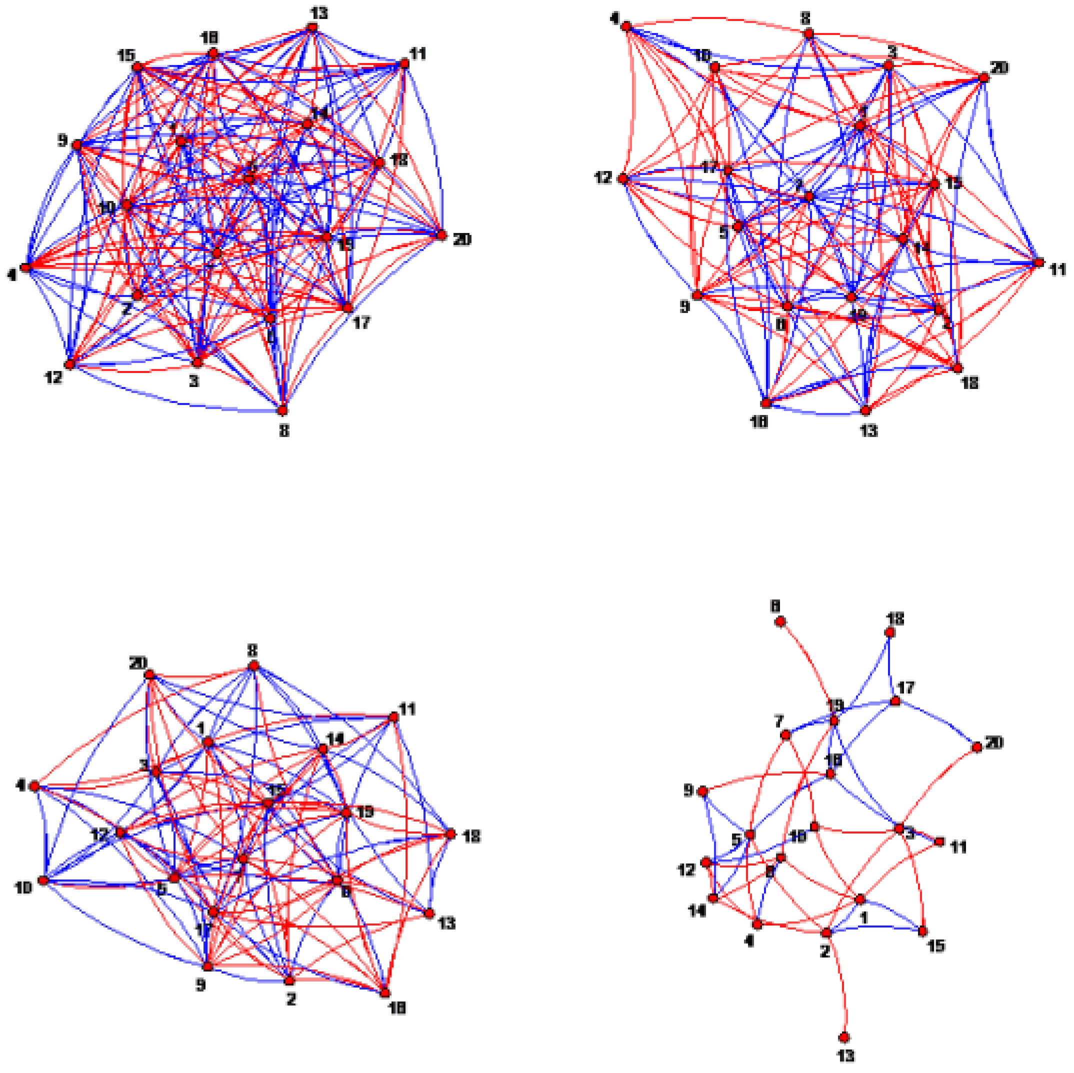

Figure 2.3: Simple representation of a network for different λ_p values (the penalty values λ_p from left to right and from first to second line, i.e., $\lambda_p = 0.01, 0.07, 0.11$ and 0.15, respectively).

2.2.6.4 Graphical Lasso (GLASSO) with L_1-Penalised Likelihood Approach

In order to estimate a sparse and symmetric precision matrix $\mathbf{\Theta}$, the L_1-penalty can be applied on the entries of the precision matrix, rather than the regression coefficients.

According to the Lagrangian dual form, the penalized likelihood optimization is given by

$$\max_{\Theta} \; [log|\mathbf{\Theta}| - Trace(\boldsymbol{S}\mathbf{\Theta}) - \lambda||\mathbf{\Theta}||_1], \tag{2.39}$$

in which λ denotes the non-negative Lagrange multiplier. Lagrange dual form is used to solve optimization problem. It adds the constraints to the objective function via Lagrange multiplier. When the value of λ increases, the sparsity of the precision matrix also increases. In addition, the optimal solution satisfies the requirement of symmetry.

There are several approaches to find λ in Equation (2.39). The *coordinate descent method* is one of these methods. In this approach, different penalty values are used for different entries of the precision matrix. By this way, the equation that is needed to be maximized becomes

$$\max_{\Theta} [log|\mathbf{\Theta}| - Trace(\boldsymbol{S}\mathbf{\Theta}) - ||\mathbf{\Theta} * \mathbf{\Lambda}||_1]. \tag{2.40}$$

Here, $\mathbf{\Lambda} = \lambda_{ij}$ is a matrix of penalty parameters and $(*)$ denotes the dot product of matrices.

The selection of λ that gives the true interaction between the nodes is also crucial for this method. An receiver operating characteristic (ROC) type of curve with different λ values can be drawn to decide the optimal λ. In ROC, the y-axis shows the true positive rate (or sensitivity) and the x-axis presents the false positive rate (or specificity). According to this plot, the value of λ which corresponds to the most top-left point on the curve can be chosen since this particular value has the lowest false positive rate (FPR) and the highest true positive rate (TPR).

Figure 2.4 and Figure 2.5 show the ROC curve for two different dimensional data, namely, 20 by 20 and 40 by 40. In these figures, each curve belongs to different threshold values such as 0.01, 0.05, 0.1 and 0.2. The threshold value is applied to convert the precision matrix into a binary form. Then this binary form enables us to calculate the true positive rate and false positive rate by comparing the estimated results with the true ones. The mathematical details about this calculation is presented in Chapter 4, i.e. in Application part. Here, TPR and FPR are computed for seven different penalty parameters λ based on each threshold value. Hence, the best penalty parameter is chosen by looking at the cut-off point in the ROC curve. For instance, in Figure 2.4 when the threshold value is equal to 0.05, the sixth penalty parameter which approximately corresponds to FPR=0.1 and TPR=0.8 indicates the best fitted penalty value.

The Banerjee methods can be other alternative approach to decide λ. It proposes the following formula for λ which is adapted from Equation 2.38:

$$\lambda(\alpha) = (\max_{i>j} \sqrt{s_{ii}s_{jj}}) \frac{t_{n-2}(\alpha/2p^2)}{\sqrt{n-2+t_{n-2}^2(\alpha/2p^2)}}. \tag{2.41}$$

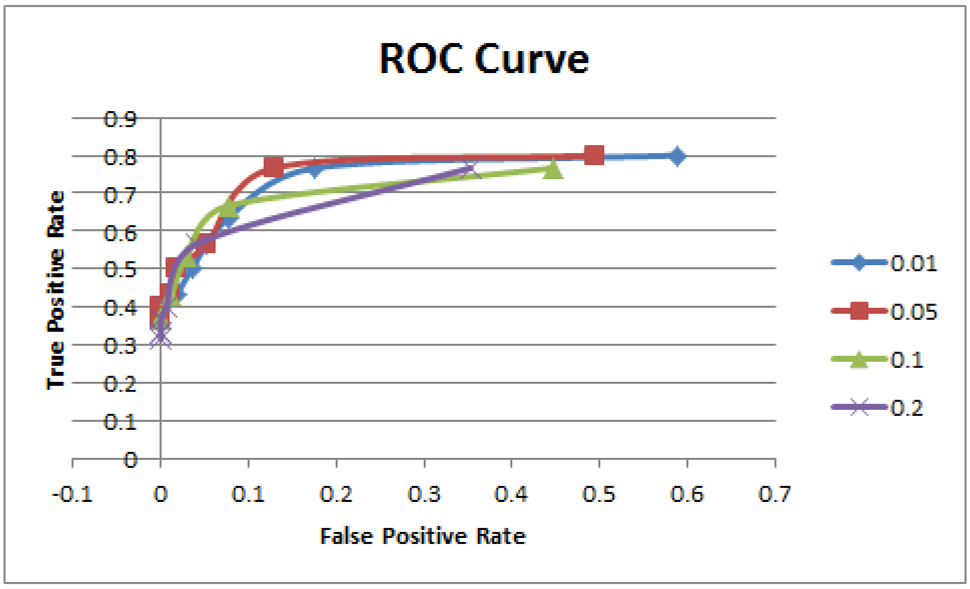

Figure 2.4: ROC curve for the 20 by 20 dimensional simulated data.

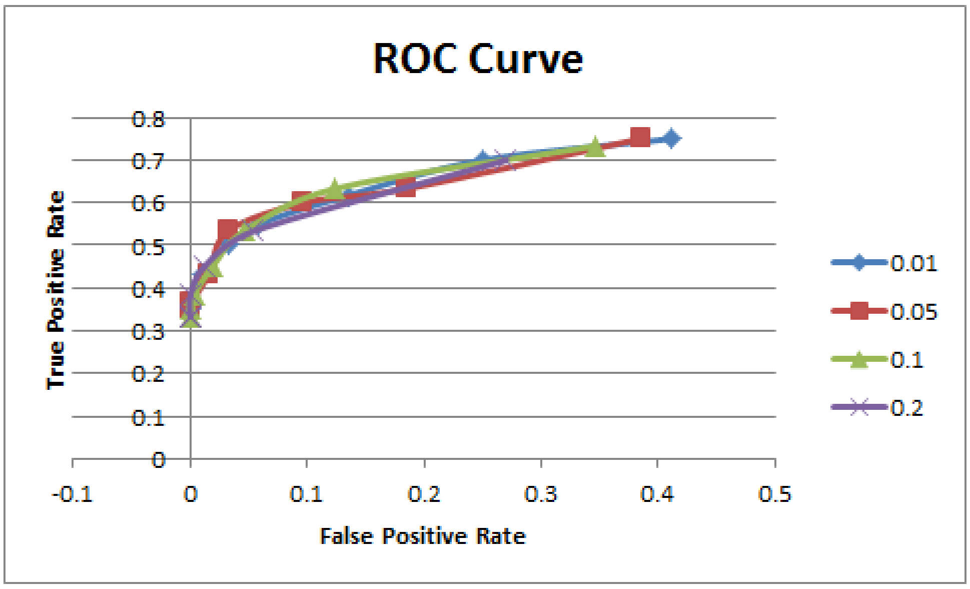

Figure 2.5: ROC curve for the 40 by 40 dimensional simulated data.

Here, $t_{n-2}(\hat{\alpha})$ denotes a student t-distribution with $n - 2$ degrees of freedom. This method is more efficient than the cross-validation method in terms of the computational time. For large datasets Banerjee method is better than the cross-validation method since it relies on the likelihood of the network. Moreover, the Banerjee method considers the false positive rate. But the false discovery rate can be also implemented to select the optimal λ for very sparse networks.

When the dimensionality is too high, that means the number of feature p is extremely larger than the number of observations n, the graphical lasso becomes computationally inefficient. Witten et al. (2011) suggest to write the estimated inverse covariance matrix as the block diagonal form in order to speed up the computation. Because they can describe a standard graphical lasso in each block separately. But necessary and sufficient condition is required so that the estimated inverse covariance matrix can be block diagonal. According to Karush-Kuhn-Tucher conditions (Witten et al., 2011), the condition which maximizes Equation (2.39), is satisfied by the following equality:

$$\mathbf{\Theta^{-1}} - \boldsymbol{S} - \lambda\Gamma(\mathbf{\Theta}) = 0, \tag{2.42}$$

in which $\Gamma(x)$ denotes the subgradient of $|x|$. That means, if $\Theta_{ij} > 0$, $\Gamma(\Theta_{ij})$ equals to 1. If $\Theta_{ij} < 0$, $\Gamma(\Theta_{ij})$ sets to -1 and if $\Theta_{ij} = 0$, the value of $\Gamma(\Theta_{ij})$ lies from -1 to 1.

If the inequality $|S_{ii'}| \leq \lambda$ is satisfied for all $i \in C_k$, $i' \in C_{k'}$, $k \neq k'$, where $C_1, C_2, ..., C_k$ represent a partition of the p features, the solution of the graphical lasso problem becomes the block diagonal matrix with K blocks such that

$$\tilde{\mathbf{\Theta}} = \begin{pmatrix} \Theta_1 & & & \\ & \Theta_2 & & \\ & & \ddots & \\ & & & \Theta_k \end{pmatrix}. \tag{2.43}$$

According to the results of the simulation study (Witten et al., 2011), the algorithms which use the blocking idea are computationally efficient. Especially, if the value of tuning parameter λ increases, the number of nodes that are fully unconnected from all other nodes increases. So, the speed of the algorithm also raises.

In our application, we implement the k-cross validation and the gradient descent algorithm which are adapted to GGM as another novelties of this thesis. In literature, both methods have been previously suggested (Defterli et al., 2013; Wit et al., 2013). Whereas, they have not been implemented in this context and their performances have not been evaluated in details to the best of our knowledge. The mathematical details of both algorithms and alternative model selection methods with their performances based on Monte Carlo runs are presented in the following chapters, as stated beforehand.

2.2.6.5 Low-Order Partial Correlations Approach

In Gaussian graphical model, the low-order partial correlation is applied to infer the full-order partial correlations. In this type of correlations, the correlation of two variables is calculated by using all other variables in the network. However, the low-order partial correlation uses only a subset of the other variables. When the network is sparse, the low-order partial correlation gives a good approximation of full-order ones. Moreover, to calculate the first-order partial correlations, Pearson coefficient and Spearman rank correlation can be implemented. The Pearson correlation coefficient has a normality assumption, whereas, the Spearman rank correlation does not need it.

As we mentioned before, the Gaussian graphical model is based on the normality assumption. If the normality assumption is violated, the GGM could not be used. In this case, the *copula-based models* are suggested in the literature. Briefly a *copula* is a multivariate distribution with uniformly distributed marginals and the copula decomposition of a general multivariate distribution uses the Sklar's theorem. This theorem states that every joint multivariate distribution can be represented by a copula and its marginals (Abbruzzo, 2012; Wit, 2012). The copula-based inference of the conditional independence graphs is one of the ongoing research topics.

2.3 Model Selection Criteria

In this section, we describe the current and our suggested selection criteria, i.e., accuracy measures in this context, in details. The assessments of selection criteria are given in Application Chapter (Chapter 4).

2.3.1 Accuracy Measures

In the literature, there are many measures used to evaluate the accuracy of the binary classification. In this study, we first of all use the well-known accuracy measures such as precision, recall, specificity, false positive rate, false discovery rate and accuracy. Then we assess the performance of F-measure and Matthews correlation coefficient which are typically used in networks in computer sciences. The explicit definitions and formulas related to these measurements are given in the following sections.

In general, the measurements which are listed above are the functions of the following four main values which are true positive, true negative, false positive and false negative value. True positive (TP) implies the number of correctly classified objects that have positive label. The true negative (TN) indicates the number of correctly classified objects that have negative label. On the contrary, the false positive (FP) shows the number of misclassified objects that have negative label and the false negative (FN) presents the number of misclassified objects that have positive label. This information can constitute a confusion matrix as shown in Table 2.3, which represents the actual and predicted classification.

Hereby, as the function of TP and FP, the precision is calculated with the following formula.

$$\text{Precision} = \frac{TP}{TP + FP}. \tag{2.44}$$

Equation (2.44) is the ratio of correctly classified objects with positive label to the total of the

Table 2.3: General confusion matrix.

		Actual class	
		Positive	**Negative**
Predicted class	**Positive**	TP	FP
	Negative	FN	TN

predicted objects as positive class. It is also called the *positive predictive value* (PPV) and higher values indicate better classification.

The recall is calculated by using Equation (2.45) and describes the ratio of correctly classified objects with positive label to the total of the positive class in actual case. It is also named as the *sensitivity* or *true positive rate* (TPR).

$$\text{Recall} = \frac{TP}{TP + FN}. \tag{2.45}$$

Additionally, the specificity denotes the ratio of correctly classified objects with negative label to the total of the objects that are actually in negative class. It is calculated by using Equation (2.46) and is also known as the *true negative rate* (TNR).

$$\text{Specificity} = \frac{TN}{TN + FP}. \tag{2.46}$$

On the other hand, the false positive rate (FPR) is the ratio of misclassified objects with positive label to the total of the negative class in actual case and the following formula is used to calculate FPR.

$$\text{FPR} = 1 - \text{Specificity} = \frac{FP}{TN + FP}. \tag{2.47}$$

Here, smaller values are interpreted as a better classification.

The false discovery rate (FDR), which is one minus the precision, is calculated as in Equation (2.48). It is the ratio of misclassification objects with positive label to the total of the objects predicted as positive class.

$$\text{FDR} = 1 - \text{Precision} = \frac{FP}{TP + FP}. \tag{2.48}$$

Finally, the accuracy is the ratio of correctly classified objects in both labels to the total of all classified objects. It is expressed by the following formula (Arslan, 2011):

$$\text{Accuracy} = \frac{TP + TN}{TP + FP + TN + FN}. \tag{2.49}$$

Apart from these main accuracy measures, we also use the F-measure and the Matthews correlation coefficient as given in the following expression. The F-measure is the harmonic mean of the precision and the recall, and it measures the overlap between actual and predicted classes. It is also called *Sorensen's similarity coefficient* (Labatut and Cherifi, 2011) and calculated via

$$\mathrm{F} = 2\frac{\mathrm{Precision} \times \mathrm{Recall}}{\mathrm{Precision} + \mathrm{Recall}}. \tag{2.50}$$

This value takes zero when there is no overlap between actual and predicted terms, and sets to one when there is a complete overlap.

The *Matthews correlation coefficient* (*MCC*) can be found in Equation (2.51). In the literature, it is accepted as a *balanced measure*. In contrast to other measurements explained above, it ranges from -1 to 1. The value 1 corresponds to the perfect classification and -1 represents the full misclassification (Fan et. al., 2009):

$$\mathrm{MCC} = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}. \tag{2.51}$$

CHAPTER 3

METHODS

In this part we present the novelties in this thesis under two subtopics, namely, the performance comparison of the novel matrix decomposition method and the modelling via the *k*-cross validation with GGM as well as the threshold gradient descent (TGD) algorithm as an alternative of GGM.

3.1 New Singularity Method

We suggest a new updating regime to solve the singularity problem of a covariance matrix **A**, which enables us to reconstitute the covariance structure by working with a lower-dimensional matrix $\mathbf{A}^*$ and its linear recovery matrix **B** (Ayyıldız et al., 2012). Let **A** be a $n \times n$ symmetric matrix of rank $p < n$. Without loss of generality, assume that the first p columns are linearly independent. In practice, this is not a restriction, as the underlying states **X** can always be reordered to achieve this. This means that the matrix **A** can be written as

$$\mathbf{A} = [\mathbf{A_1} \;\; \mathbf{A_2}],$$

where $\mathbf{A_1}$ is an $(n \times p)$ and $\mathbf{A_2}$ an $(n \times (n - p))$ matrix, for which the latter can be written as

$$\mathbf{A_2} = \mathbf{A_1 B},$$

for a $(p \times (n - p))$ matrix **B**, since $\mathbf{A_2}$ is linearly dependent on $\mathbf{A_1}$ by the assumption of rank p of **A**. In other words,

$$\mathbf{A} = \mathbf{A_1}[\mathbf{I_p} \;\; \mathbf{B}]$$

for p dimensional identity matrix $\mathbf{I_p}$. By decomposing $\mathbf{A_1}$ in an upper $(p \times p)$ dimensional and a lower $((n - p) \times p)$ dimensional matrix, $\mathbf{A_1} = [\mathbf{A_{11}} \;\; \mathbf{A_{12}}]^{\mathbf{T}}$, we can write **A** as

$$\mathbf{A} = \begin{bmatrix} A_{11} & A_{11}B \\ A_{12} & A_{12}B \end{bmatrix}.$$

Given the symmetry of **A**, we get that $\mathbf{A_{12}^T} = \mathbf{A_{11}B}$ and so

$$\mathbf{A} = \begin{bmatrix} A_{11} & A_{11}B \\ B^T A_{11}^T & B^T A_{11}^T B \end{bmatrix}. \tag{3.1}$$

The crucial element of the decomposition in Equation (3.1) is that the distribution can be described by means of the $(p \times p)$ non-singular matrix $\mathbf{A_{11}}$, whereas, the matrix $\mathbf{B}$ of the linear coefficients allows us to reconstitute the linearly dependent elements.

For example, if this method is used in MCMC (Monte-Carlo Markov Chain) inference of stochastic differential equations, a singular diffusion matrix can be first transformed into a nonsingular and lower-dimensional form and the diffusion matrix can be updated under such lower-dimensional space. Then the updated and independent columns are re-applied in the generation of the omitted columns, which are eliminated previously due to the dependency, by using the linear relations encoded in matrix $\mathbf{B}$. Hereby, the steps of the algorithm for matrix $\mathbf{A}$ can be listed as follow:

1. Each linearly dependent column $\mathbf{A} = [\mathbf{A_1} \;\; \mathbf{A_2}]$ is identified by checking the columns of $\mathbf{A}$ from left to right.
2. The dependent columns, $\mathbf{A_2}$, totally p, are described as the linear combination of independent columns.
3. A new $(n-p) \times (n-p)$ dimensional covariance matrix $\mathbf{A}^*$ is defined by eliminating p dependent columns and rows from $\mathbf{A_{n \times n}}$.

3.1.1 Numerical Example

In genomics, finance and several other fields, the aim is to explain the change in state of a gene, a stock or other variable by applying a deterministic drift term plus a stochastic diffusion term, involving a Brownian motion. In inferring multivariate stochastic differential equations (Golightly and Wilkinson, 2005; Purutçuoğlu and Wit, 2008) of the form

$$dX_t = \boldsymbol{\mu}(X_t, t)dt + \boldsymbol{\sigma}(X_t, t)dB_t, \tag{3.2}$$

the aim is to obtain estimates of the drift $\boldsymbol{\mu}$ and diffusion matrix $\boldsymbol{\sigma}$. The stochastic differential equation is the extension of the deterministic differential equation by adding stochastic influences (Øksendal, 2003). Since many natural phenomena have random dynamics, such an expression, which can deal with the randomness from the natural structure of the system, becomes crucial. For instance, the biological processes like the transcription and translation of the proteins can be only explained by this type of randomness.

Whereas, the inference problem of $\boldsymbol{\mu}$ is relatively straightforward. But, in inference for $\boldsymbol{\sigma}$ can involve computational problems. In fact, especially, in high-dimensional problem, the term $\boldsymbol{\sigma}(X_t, t)$ can be computationally singular, or even exactly singular for particular values of X_t

and t. In practice, this can occur when the number of financial instruments increases or when the number of genes included in the model becomes large. Also as the system converges to its stationary distribution, we are often faced with the problem of linear dependency between the state space quantities that causes a singularity in diffusion matrix.

As an implementation of the new method, we give the following numerical example by applying Equation (3.2). Let the total number of substrates in the system be 3, $n = 3$, and the time step dt is taken as $dt = 0.1$ unit. Also the drift term $\boldsymbol{\mu}$ and diffusion matrix $\boldsymbol{\sigma}$ set to $\boldsymbol{\mu} = [3 \;\; 5 \;\; 7]^T$ and

$$\boldsymbol{\sigma} = \begin{bmatrix} 4 & 16 & 0 \\ 16 & 64 & 0 \\ 0 & 0 & 9 \end{bmatrix},$$

respectively. By controlling the column of $\boldsymbol{\sigma}$ from left to right, we see that the second column of $\boldsymbol{\sigma}$ is four times of its first column, thereby, the corresponding $\boldsymbol{B}$ vector which shows the linear dependence between the second column and the remaining columns is $\boldsymbol{B} = [4 \;\; 0]$. In order to get the non-singular submatrix, we decompose the diffusion matrix $\boldsymbol{\sigma}$ as

$$\boldsymbol{\sigma} = [\boldsymbol{\sigma_1} \;\; \boldsymbol{\sigma_2}],$$

where

$$\boldsymbol{\sigma_1} = \begin{bmatrix} 4 & 0 \\ 0 & 9 \\ 16 & 0 \end{bmatrix} \text{ and } \boldsymbol{\sigma_2} = \begin{bmatrix} 16 \\ 0 \\ 64 \end{bmatrix}.$$

Since $\boldsymbol{\sigma_2}$ is linearly dependent on $\boldsymbol{\sigma_1}$, the following equality is satisfied.

$$\boldsymbol{\sigma_2} = \boldsymbol{\sigma_1}\mathbf{B} = \begin{bmatrix} 4 & 0 \\ 0 & 9 \\ 16 & 0 \end{bmatrix} \begin{bmatrix} 4 \\ 0 \end{bmatrix}.$$

Hereby, we can obtain an upper (2×2) dimensional matrix and a 2-dimensional vector by decomposing $\boldsymbol{\sigma_1} = [\boldsymbol{\sigma_{11}} \;\; \boldsymbol{\sigma_{12}}]^T$ via

$$\boldsymbol{\sigma_{11}} = \begin{bmatrix} 4 & 0 \\ 0 & 9 \end{bmatrix} \text{ and } \boldsymbol{\sigma_{12}} = \begin{bmatrix} 16 & 0 \end{bmatrix}.$$

Hence, we can generate the non-singular and lower-dimensional matrix $\boldsymbol{\sigma^*}$ by $\boldsymbol{\sigma^*} = \boldsymbol{\sigma_{11}}$. Then, in order to obtain $\boldsymbol{\Delta}_j = \boldsymbol{\sigma^{*1/2}} \times dB_t$ according to Equation (3.2), resulting in the calculation of the diffusion term for the independent substrates changed by the Brownian motion, we generate two random values from normal distribution with mean zero and variance as the time step 0.1, i.e. $dB_t \sim N(0, 0.1)$. Then we multiply them by $\boldsymbol{\sigma^{*1/2}}$. The values of dB_t are found as $dB_t = [-0.524 \;\; -0.251]$ from R programme language with *rnorm* function. Accordingly, $\boldsymbol{\Delta}_j$ associated to the first and third substrates are obtained as $\boldsymbol{\Delta}_{j=1,3} = (-1.048, -0, 753)$.

On the other hand for the update of the dependent term, i.e., the corresponding value of the second term, we calculate $\boldsymbol{\Delta_{i=2}} = \boldsymbol{\alpha} \times \boldsymbol{\Delta_j} = [4 \;\; 0] \times \binom{-1.048}{-0.753} = -4.192$. And finally, the change in state based on the diffusion model is computed as follows:

$$\begin{aligned}
\boldsymbol{\Delta X_t} &= \boldsymbol{\mu} \times d_t + \boldsymbol{\Delta} \\
\boldsymbol{\Delta X_t} &= [3 \;\; 5 \;\; 7] \times 0.1 + [-1.048 \;\; -4.192 \;\; -0.753] \\
&= [-0.748 \;\; -3.692 \;\; -0.053]
\end{aligned}$$

In the further analysis if the result is used for a simulation purpose, this change is applied for the update of the system in the next state. Hereby, the new state $\boldsymbol{X_{new}}$ can be generated by adding the current state $\boldsymbol{X_{old}}$ to the change $\boldsymbol{\Delta X_t}$, i.e., $\boldsymbol{X_{new}} = \boldsymbol{X_{old}} + \boldsymbol{\Delta X_t}$ (Purutçuoğlu and Wit, 2006; Purutçuoğlu, 2007; Ayyıldız et al., 2012).

3.2 Suggested Optimization Algorithms Adapted in GGM

In this section, we explain the k-cross validation and the gradient descent algorithm in the application of the inference in GGM. As we mentioned before, these two methods have been suggested in the literature. However, they have not been applied in GGM context yet. The mathematical details of both algorithms are given in the following part and their Monte-Carlo results are presented in Application Chapter.

3.2.1 *K*-Cross Validation

The cross-validation method is an alternative approach to decide on the penalty parameter, i.e., λ in the L_1-penalized likelihood function. This method is based on the maximization of the log-likelihood function with respect to $\boldsymbol{\Theta}$. Hereby, the cross validation separates the available data into the two parts, namely, training set and test, i.e., validation set. The training set is used to estimate the parameters and the validation set is applied to test it.

In the k-fold cross validation, we randomly split the data into k folds which have equal sizes. In the literature, generally k equals to 10 (Friedman et al., 2008; Wit et al., 2013) in such a way that $(k-1)$ folds are chosen as a training set and the parameter estimation is done based on these data. Then, the remaining 1 fold is assigned as a test set. In this stage, the likelihood function is computed. After that, these steps are repeated for k times. In each time, a different fold becomes a test set. Finally, the value of λ, which maximizes the likelihood is determined as the optimal penalty parameter.

Accordingly, we list the steps of the k-fold cross validation method as below:

1. Split the data randomly into k folds.

2. Estimate the precision matrix by using a glasso model from $(k-1)/k$ of the data.
3. Use this matrix to compute the likelihood on the $1/k$ of the data taken out. Here, apply the estimate of mean and covariance values found from $(k-1)$ data.
4. Repeat the steps 2 and 3 for all folds so that we get k likelihoods from each k fold.
5. Sum all likelihoods for each candidate penalty λ.
6. Choose the value of λ that maximizes this likelihood.

3.2.2 Threshold Gradient Descent Algorithm

As we mentioned earlier, the Gaussian graphical model (GGM) is based on the partial correlation as a measure of independence between two genes. Since the structure of the genetic networks is not fully connected, the related precision matrix has a sparse construction. Thereby, when we estimate the precision matrix, we need to take into account the sparsity condition. Furthermore, the estimation of the precision matrix in GGM faces with small n and large p problem. So, while the number of variables p is large with respect to the sample size n, the maximum likelihood estimation method (MLE) can be applied to estimate the precision matrix. On the other side, if n is smaller than p, MLE does not give unique solution. Therefore, new procedures are required to overcome the challenges of multiple estimators in MLE. One of these alternative procedures can be the threshold gradient descent (TGD) method (Streib and Dehmer, 2008; Li and Gui, 2006). TGD penalizes the estimation of the off-diagonal elements in the sparse precision matrix by the following way.

First of all, we split the precision matrix $\mathbf{\Theta}$ into the two parts, namely, diagonal vector and off-diagonal vector. Let $\boldsymbol{\theta^d} = \theta_{11}, ..., \theta_{pp}$ represent the vector of diagonal elements of the precision matrix $\mathbf{\Theta}$ and $\boldsymbol{\theta^o} = \{\theta_{ij}\}_{i \neq j}$ present the vector of off-diagonal elements of $\mathbf{\Theta}$. Hence, the upper triangular part of $\mathbf{\Theta}$ includes $q = p(p-1)/2$ elements as the precision needs to be symmetric. Then, the likelihood function takes the following form:

$$\mathbf{\Theta}(\boldsymbol{\theta^d}, \boldsymbol{\theta^o}) = \frac{n}{2} \log |\mathbf{\Theta}| - \frac{1}{2} \sum_{k=1}^{n} \boldsymbol{X^{(k)\prime}} \mathbf{\Theta} \boldsymbol{X^{(k)}}, \tag{3.3}$$

where $\boldsymbol{X^{(k)}}$ denoted the vector of the kth observation.

To deal with the sparsity of the precision matrix, the negative of the log-likelihood function can be defined as a loss function via

$$l(\boldsymbol{\theta^d}, \boldsymbol{\theta^o}) = -\mathbf{\Theta}(\boldsymbol{\theta^d}, \boldsymbol{\theta^o}). \tag{3.4}$$

Also the gradient of the loss function with respect to $\mathbf{\Theta}$ can be written as

$$\frac{\partial l}{\partial \mathbf{\Theta}} = \frac{n}{2} \mathbf{\Theta^{-1}} - \frac{1}{2} \sum_{k=1}^{n} \boldsymbol{X^{(k)}} \boldsymbol{X^{(k)\prime}}. \tag{3.5}$$

After setting the initial values of $\boldsymbol{\theta^d}$ to unit vector and $\boldsymbol{\theta^o}$ to zero vector, we start to update the off-diagonal elements of the precision matrix $\boldsymbol{\theta^o}$ by using the gradient descent step such that

$$\hat{\Theta}^o(\gamma + \triangle\gamma) = \hat{\Theta}^o(\gamma) + \triangle\gamma d(\gamma). \tag{3.6}$$

Here, $\hat{\Theta}^o(\gamma)$ denotes the value of θ^o for the current γ, $\triangle\gamma$ shows very small increment ($\triangle\gamma > 0$), and $d(\gamma)$ describes the direction of the tangent vector which corresponds to a descent direction at each step.

Friedman and Popescu (2004) define $d(\gamma)$ as the multiplication of two functions such that

$$d(\gamma) = w_j(\gamma) \cdot g_j(\gamma) \tag{3.7}$$

for $j = 1, 2, ..., q$, in which

$$w_j(\gamma) = I[|g_j(\gamma)| \geq \lambda \cdot \max_{1\leq k\leq q} |g_k(\gamma)|]. \tag{3.8}$$

In Equation (3.8), $I[\cdot]$ denotes an indicator function and λ stands for a threshold parameter used to adjust the diversity of the values of $w_j(\gamma)$. λ takes the value between 0 and 1. Thus, as the diversity increases, λ is closer to 1 meaning that $\lambda = 1$ implies the sparsest graph.

On the other hand, in order to update the diagonal elements of the precision matrix, these diagonal entries are maximized with respect to the log-likelihood function in Equation (3.3). In this stage, different iterative methods can be performed. Here, we choose the Newton-Raphson approach for this purpose. The steps of the Newton-Raphson algorithm are explained in the following part.

In conclusion, we can summarize the TGD algorithm in six steps as described below:

1. Initialy set $\theta^o(0) = 0$, $\theta^d(0) = 1$, and $\gamma = 0$.
2. Calculate $g(\gamma) = -\partial l/\partial\theta^o$ by using the current θ^o and θ^d.
3. Compute $w_j(\gamma) = I[|g_j(\gamma)| \geq \lambda \cdot \max_{1\leq k\leq q} |g_k(\gamma)|]$ and $d(\gamma)$ by using $w(\gamma)$ and $g(\gamma)$ in Equation (3.7) and Equation (3.8), respectively.
4. Update $\boldsymbol{\theta^o}$ via $\theta^o(\gamma + \triangle\gamma) = \theta^o(\gamma) + \triangle\gamma d(\gamma)$ and γ via $\gamma = \gamma + \triangle\gamma$.
5. Update $\boldsymbol{\theta^d}$ by using Newton-Raphson iterations.
6. Repeat steps from 2 to 5 until the convergence is achieved.

On the other hand, the Newton-Raphson algorithm which we use in the update of $\boldsymbol{\theta^d}$ can be described as follows:

The Newton-Raphson is a commonly used iterative method. This method is based on the first order derivative of a function which is maximized. Accordingly, the Newton-Raphson equation is derived from the first order Taylor series expansion via

$$f(x_{n+1}) = f(x_n) - f^{'}(x_n)(x_{n+1} - x_n). \tag{3.9}$$

Here, if $f(x_{n+1})$ is taken to be 0, the equation takes the following form,

$$0 \cong f(x_n) - f^{'}(x_n)(x_{n+1} - x_n). \tag{3.10}$$

Hence, the Newton-Raphson equation can be written as

$$x_{n+1} = x_n - \frac{f(x_n)}{f^{'}(x_n)}, \tag{3.11}$$

in which x_n denotes the current estimate, x_{n+1} represents the next estimate, and $f^{'}(\cdot)$ is the first derivative of $f(\cdot)$.

This equation is used iteratively until the convergence is satisfied. Although Newton-Raphson is a commonly performed approach, there is a drawback of being affected by these initial values. Thus, it is crucial to choose a feasible and good initial value in a reasonable range.

In the following chapter, we implement these two methods in inference of the small and large systems via GGM. The newly adapted R codes of both algorithms are presented in Appendix.

CHAPTER 4

APPLICATION AND RESULTS

4.1 Simulation Study for New Singularity Method

In order to compare the performance of the eigenvalue decomposition, singular value decomposition and the Cholesky decomposition method, we use simulated datasets and generate different covariance matrices which are symmetric and singular. In the matrix generation, firstly, we produce independent columns one by one from left to right. Then, these columns are used to construct the linearly dependent columns. From the empirical studies we observe that the location of dependent and independent columns does not make any difference in our method since whatever the order of the dependent and independent columns, we always check each column of matrices from left to right sequentially and compare whether each proposal column is linearly dependent on previous columns. To obtain the symmetric matrix from the underlying singular matrix, the covariance matrix is calculated. These matrices are generated under two different scenarios. In the first plan, we generate matrices whose 80 % of the columns are independent and the remaining 20 % of columns has linear dependency. Then in the second scenario, we use matrices whose 80 % of columns are linearly dependent, and the remaining ones are taken as independent. Under both scenarios, we apply matrices having different dimensions, namely, 50 by 50, 100 by 100, 400 by 400, 500 by 500, and 750 by 750. In all computational works, we carry in the R programming language version 2.10.0 and execute our codes on Core 2 Duo 2.0 GHz processor.

To compare the performance of alternative methods with our proposal approach, we check the CPU (Central Processing Unit) times in seconds. In order to estimate the CPU times of each method, we consider two scenarios. In the first scenario, we iterate the algorithm 1000 times for each matrix structure with distinct dimensions and take the mean of the underlying Monte Carlo outputs once the selected methods decompose the singular matrices and detect the linear relationships between their columns. However, because of the time restriction and computational inefficiency, we iterate the algorithm 250 times for the matrix dimension 750 by 750.

Table 4.1 and Table 4.2 show the mean CPU times of each matrix decomposition. From the results it is seen that there is no difference between methods per iteration on average in both data plans. On the other hand, in the second scenario, rather than computing each iteration separately, we report the total CPU time of 1000 Monte Carlo iteration.

Although the new method has more computational steps, it is seen that there is no difference in mean CPU time of methods for low-dimensional like (50×50) and moderately high-dimensional matrices such as the matrices under (100×100) and (400×400) dimensions. Whereas, if we consider very high dimension like (500×500) or (750×750), the new method slightly looses more time with respect to other methods under both scenarios. For instance, the mean CPU times for the 750 by 750 matrix under the first scenario with 250 iterations are 0.008, 0.008, 0.002, and 2.147 for the eigenvalue, singular value, Cholesky, and our new method, respectively. Similarly the mean CPU times for this matrix dimension under the second scenario are 0.014, 0.014, 0.003, and 0.785 for the same order of decomposition methods. On the other hand, when we increase the number of dependent columns for the same dimensional matrix, it is seen that all of the methods own smaller CPU times. However, the mean CPU time of our method decreases more sharply. This result can be interpreted that our new method can be preferable when the singularity of the matrix becomes a severe challenge. Since CPU time of each iteration is an infinitesimal, when we take the mean of the underlying Monte Carlo outputs, the results become zero. So the differences between decomposition methods cannot be detected exactly. Thereby, we decide to calculate total CPU time of 1000 Monte Carlo iterations. In this calculations, we record the total time only to decompose singular matrices for eigenvalue, singular value, and Cholesky methods, and keep the time to both decompose the singularity and detect the linear relationship between columns for our suggested approach. We use the same two scenarios and the same dimensions as before.

Table 4.1: Comparison of mean CPU times under 1000 Monte-Carlo runs for eigenvalue, singular value, and Cholesky decompositions with new method based on different dimensional matrices under the first scenario.

	Dimensions			
Methods	(50×50)	(100×100)	(400×400)	(500×500)
Eigen-value decomposition	0	0	0	0.001
Singular value decomposition	0	0	0	0.001
Cholesky decomposition	0	0	0	0.000
Our method	0	0	0	0.318

Table 4.3 shows CPU times of matrices with 4 different sizes under the first plan. From the results it is observed that although the new method has more computational steps, there is no difference in CPU times for low like (50×50) and moderately high-dimensional matrices such as the matrices under (100×100) and (400×400) dimensions. Whereas, if we consider very high dimensions like (500×500) or (750×750), the new method slightly looses more time with respect to other methods under both scenarios. For instance, CPU times for the (750×750) matrix under the first scenario with 250 iterations are 0.008, 0.008, 0.002 and 2.147 for the eigenvalue, singular value, Cholesky and our new method, respectively. Similarly, CPU times

Table 4.2: Comparison of mean CPU times under 1000 Monte-Carlo runs for eigenvalue, singular value, and Cholesky decompositions with new method based on different dimensional matrices under the second scenario.

Methods	**Dimensions** (50 × 50)	(100 × 100)	(400 × 400)	(500 × 500)
Eigen-value decomposition	0	0	0	0.007
Singular value decomposition	0	0	0	0.007
Cholesky decomposition	0	0	0	0.002
Our method	0	0	0	0.041

for this matrix dimension under the second plan are found as 0.014, 0.014, 0.003 and 0.785 for the same order of decomposition's methods. On the other side, when we increase the number of dependent columns under the same dimension as seen in Table 4.4, we observe that all methods use less CPU times. However, the CPU time of our method decreases more sharply and its becomes the most efficient method in computation when both dependency and dimension are big. This findings can be interpreted that our new method can be mainly preferable when the singularity becomes a severe challenge for the matrix.

Table 4.3: Comparison of total CPU times under 1000 Monte-Carlo runs for eigenvalue, singular value and Cholesky decompositions with new method based on different dimensional matrices under the first scenario.

Methods	**Dimensions** (50 × 50)	(100 × 100)	(400 × 400)	(500 × 500)
Eigenvalue decomposition	0.05	0.03	34.86	27.28
Singular value decomposition	0.03	0.02	13.11	30.72
Cholesky decomposition	0.01	0.03	15.81	27.11
Our method	0.01	0.02	37.68	265.74

Table 4.4: Comparison of total CPU times under 1000 Monte-Carlo runs for eigenvalue, singular value and Cholesky decompositions with new method based on different dimensional matrices under the second scenario.

Methods	**Dimensions** (50 × 50)	(100 × 100)	(400 × 400)	(500 × 500)
Eigenvalue decomposition	0.03	0.03	2.85	34.60
Singular value decomposition	0.02	0.05	1.47	29.35
Cholesky decomposition	0.03	0.04	0.09	23.48
Our method	0.05	0.05	0.94	9.44

In conclusion, from the assessment we have observed that all methods have the same CPU times for low and moderately high-dimensional matrices. But, when the dimension increases and the singularity is not seen in the majority of the columns, our approach is comparable to the other methods. However, if we also raise the percentage of dependent columns, i.e., the singularity, becomes severe in realistically large matrices, our proposal method becomes more efficient.

Besides the simulation study, we also compare the Cholesky decomposition and our proposed method, theoretically. Running times, i.e., $T(n)$, are calculated by using Θ function. Here, n represents the dimension of the matrix. In Θ function, asymptotic growth rate is bounded both above and below. Running time of the Cholesky decomposition is $\Theta(n^3)$ (Santos and Chu, 2003). Moreover, required time for our algorithm changes with the rank of the matrix. In the worst case, when the rank equals to the number of columns of a matrix, i.e., all columns are independent, the running time of our proposed method is $\Theta(n^4)$. However, in the best case, when the rank equals to 1, the required time for our method is $\Theta(n^2)$. These results support that when the percentage of dependent columns increases, our proposed method becomes more efficient.

Moreover, since our proposed method enables us to keep the same structure of the original entries of matrix by saving the linear relationship between columns, it can be more effectively applied in the reconstruction of the nonsingular version of the original matrix as needed in simulation studies (Purutçuoğlu and Wit, 2006) or modelling of the complex network structure by stochastic differential equation (Golightly and Wilkinson, 2005) or implementing the diffusion model in financial stochastic volatility models (Chib et al., 2006). Because in this approach, the preserved information of the linear relationship in the actual matrix can be used to regenerate eliminated columns. We believe that apart from these exemplified scenarios of singularities, our proposal algorithm of decomposition can be helpful for other singularity problems faced with in linear algebra and multiple statistical analysis.

4.2 Monte-Carlo Runs for the Model Selection Criteria

In order to compare the accuracy measures which are explained in Chapter 2, we use simulated datasets and estimate precision matrices of these data by using the L_1-penalized likelihood approach. Then we calculate each accuracy measure. First of all, we generate true precision matrices under two different scenarios. In the first scenario, the precision matrix has positive entries which correspond to positive relations between genes. In the second scenario, it has negative entries which represent negative relations between genes. Under both scenarios, we apply matrices having different dimensions, namely, 5 by 5, 10 by 10, 20 by 20, 40 by 40 and 50 by 50. The precision matrices have different sparsity percentages. In literature, there are several ways to generate a sparse matrix in distinct sciences. In graphical models, a *band matrix* is commonly used. This matrix is a matrix structure whose diagonal and parallel to diagonal entries are nonzero and remaining terms set to zero. According to the band width, which determines how many parallel diagonals exist rather than main diagonal, the matrix takes different names. For instance, if the band width is zero, the matrix is called as a *diagonal* *matrix*. If the band width equals to one, the matrix is called as a *tridiagonal*. The following matrix $\boldsymbol{A}$ represents the structure of the tridiagonal matrix as illustration,

$$\boldsymbol{A} = \begin{pmatrix} x & x & 0 & 0 & 0 \\ x & x & x & 0 & 0 \\ 0 & x & x & x & 0 \\ 0 & 0 & x & x & x \\ 0 & 0 & 0 & x & x \end{pmatrix}. \tag{4.1}$$

Here, x denotes the nonzero entries. Thus our precision matrices have a special form which is similar to the tridiagonal matrix. The only difference between our precision and the tridiagonal matrix is that the right upper corner and left lower corner entries in our precision matrix are nonzero whose general representation is given below,

$$\boldsymbol{B} = \begin{pmatrix} x & x & 0 & 0 & x \\ x & x & x & 0 & 0 \\ 0 & x & x & x & 0 \\ 0 & 0 & x & x & x \\ x & 0 & 0 & x & x \end{pmatrix}. \tag{4.2}$$

For all different dimensional matrices, we use the form in Equation (4.2). So the matrices which we deal with have distinct sparsity percentages. The 5 by 5, 10 by 10, 20 by 20, 40 by 40 and 50 by 50 matrices have 40%, 70%, 85%, 92.5% and 94% sparsity, respectively. Then, we estimate precision matrices by using the L_1-penalized likelihood with different parameters. In order to compute true positive, true negative, false positive and false negative values as described on Chapter 2, we convert the true and estimated precision matrices into a binary form via a threshold value. In the calculation, we take this threshold as 0.01 since in most of the biological studies whose real time-course data are not available and the information about the system is uncertain, the expression values which may be higher than zero are typically taken as the candidate genes in the system. Accordingly, by setting all the estimated values less than 0.01 to zero, and allowing the remaining as 1, we enforce the system to be complex, i.e., as highly connected as possible, since the most of realistic biochemical systems have such structure from their natures. That is, the genes have dense connections and even some of them are densely connected by generating hubs (Barabasi and Oltvai, 2004). Moreover, the proteins in the different systems are interconnected with each other in different levels as well (Wit et al., 2013). Thus, we iterate the algorithm 10000 times and take the mean of the underling Monte-Carlo outputs. In each iteration, we choose the best score for each measure among different scores which correspond to the different penalizing constant λ parameters. In the L_1-penalized likelihood method, there is no specific way to determine upper boundary of λ. The selection of λ differs according to sample covariance matrix. We choose 0.1 as a minimum value of λ. Also, the maximum value of sample covariance matrix is used as a maximum value of λ. After that, for each matrix, we divide these ranges with equal width and obtain 7 different λ values. In all computational works, we carry in the R programming language version 2.15.1 and execute our original codes on Core i5 3.10 GHz processor.

Table 4.5: Comparison of precision, recall, specificity and accuracy under 10000 Monte-Carlo runs based on different dimensional matrices under only positive off-diagonal entries (first scenario).

Dimensions	**Precision**	**Recall**	**Specificity**	**Accuracy**
Perfection Level	1	1	1	1
(5×5)	0.9734	0.9180	0.9668	0.8038
(10×10)	0.9635	0.8968	0.9911	0.8385
(20×20)	0.9703	0.8554	0.9975	0.9077
(40×40)	0.9817	0.7939	0.9995	0.9515
(50×50)	0.9852	0.7712	0.9996	0.9611

Table 4.6: Comparison of false discovery rate, false positive rate, F-measure and Matthews correlation coefficient under 10000 Monte-Carlo runs based on different dimensional matrices under only positive off-diagonal entries (first scenario).

Dimensions	**FDR**	**FPR**	**F-measure**	**MCC**
Perfection Level	0	0	1	1
(5×5)	0.0266	0.0332	0.8374	Not applicable
(10×10)	0.0365	0.0089	0.6917	0.6098
(20×20)	0.0297	0.0025	0.6021	0.5896
(40×40)	0.0183	0.0005	0.5511	0.5813
(50×50)	0.0148	0.0004	0.5399	0.5795

When we compare the results of Monte-Carlo runs, it is seen that both specificity and accuracy close to 1 when the dimension of matrices increases in the first and second scenarios. Whereas recall and F-measure decrease substantially. Also, the precision is not affected by change in dimension of matrix. Since the false discovery rate (FDR) equals to one minus the precision and the false positive rate (FPR) is found by one minus the specificity, their behaviors are similar to the precision and the specificity, respectively. The only difference is that FDR and FPR are calculated with respect to falsely classified objects, thereby, smaller values are interpreted as better classification.

On the other hand, if any two values within true positive, true negative, false positive and false negative value are zero, the Matthews correlation coefficient (MCC) is not calculated because of the fact that the denominator of its function becomes zero. This situation frequently happens in small matrices such as the matrix with dimension 5 by 5.

Additionally, for 10 by 10, 20 by 20, 40 by 40 and 50 by 50 dimensional matrices, the speci-

Table 4.7: Comparison of precision, recall, specificity and accuracy under 10000 Monte-Carlo runs based on different dimensional matrices under only negative off-diagonal entries (second scenario).

Dimensions	**Precision**	**Recall**	**Specificity**	**Accuracy**
Perfection Level	1	1	1	1
(5×5)	0.9973	0.9192	0.9973	0.7825
(10×10)	0.9946	0.8967	0.9989	0.8364
(20×20)	0.9944	0.8545	0.9996	0.9072
(40×40)	0.9960	0.7941	0.9999	0.9513
(50×50)	0.9968	0.7707	0.9999	0.9609

Table 4.8: Comparison of false discovery rate, false positive rate, F-measure and Matthews correlation coefficient under 10000 Monte-Carlo runs based on different dimensional matrices under only negative off-diagonal entries (second scenario).

Dimensions	**FDR**	**FPR**	**F-measure**	**MCC**
Perfection Level	0	0	1	1
(5×5)	0.0027	0.0027	0.8225	Not applicable
(10×10)	0.0054	0.0011	0.6862	0.6054
(20×20)	0.0056	0.0003	0.5987	0.5871
(40×40)	0.0040	0.0000	0.5489	0.5792
(50×50)	0.0031	0.0000	0.5378	0.5779

ficity is the best accuracy measure. Furthermore, from the tables, it is seen that the precision gives the best result for 5 by 5 matrix. However, there is a very small difference between precision and specificity for this matrix size as well. Therefore, we conclude that under both scenarios, the specificity can be taken as the best measure for the model selection.

4.3 GGM via Suggested Optimization Algorithms

In this section, we apply the *k*-cross validation and threshold gradient descent algorithms to estimate the precision matrix. The steps of these algorithms are explicitly defined in previous chapter and corresponding R codes of both algorithms are presented in Appendix.

In order to compare the results, we use the same simulated dataset as the observation matrix with L_1-penalized likelihood approach. So, we have different matrices which have distinct dimensions, namely, 5 by 5, 10 by 10, 20 by 20, 40 by 40 and 50 by 50. In addition, our true precision matrices are generated by using two scenarios as previously described. Hereby, in the first scenario, the precision matrices have positive entries and in the second one, they include negative entries. After the precision matrices are estimated by the *k*-cross validation and TGD methods, we convert these matrices into a binary form via same threshold value, 0.01. Then, we calculate TP, TN, FP and FN values by comparing true and estimated precision matrices. Finally, we compute the specificity which is chosen the best accuracy measure of the binary classification in the previous section. F-measure is also computed in order to make unbiased comparison. These specificity and F-measure values based on the mean of the Monte-Carlo outputs of 10000 runs.

4.3.1 GGM via *K*-Cross Validation

In the application of the cross validation, we use the 10-fold cross validation method. As we mentioned above, we use the same generated dataset with the L_1-penalized likelihood approach. So, each matrix includes 10 observations. Since 10-fold cross validation iterates 10 times, in each iteration, 9 observations are used to constitute the training set and the remaining 1 observation is taken as a test set. From the training set, the mean and covariance terms are calculated. Moreover, for each penalty value, the precision matrix is estimated via the L_1-penalized likelihood approach. Then, by using these parameters, we calculate the likelihood of the test set and sum up these 10 likelihood values. After that, we choose the best penalty which gives the maximum likelihood. Finally, the precision matrix which correspond to the best penalty parameter is estimated. Then, the accuracy measures, namely, specificity and F-measure, are calculated. The mean values of specificity and F-measure under 10000 Monte Carlo runs based on different dimensional matrices and different scenarios of precision matrix are presented in Table 4.9.

Table 4.9: Comparison of specificity and F-measure under 10000 Monte-Carlo runs based on different dimensional matrices under two different scenarios for the precision matrix.

	Scenario 1		**Scenario 2**	
Dimensions	**Specificity**	**F-measure**	**Specificity**	**F-measure**
(5×5)	0.6887	0.7112	0.8095	0.7500
(10×10)	0.8835	0.6100	0.8456	0.6344
(20×20)	0.9405	0.5521	0.9607	0.5614
(40×40)	0.9661	0.5076	0.9792	0.5210
(50×50)	0.9727	0.4957	0.9816	0.5091

The results in Table 4.9 show that there is no big difference in specificity values between two scenarios except 5 by 5 dimensional matrix. However, specificity values differ with respect to the matrix dimension. If the dimension of the matrix increases, the sparsity percentage of matrices also increases in our generated matrix form. Hence, the specificity values close to 1 when the matrix dimension increases. On the other side, when the matrix dimension increases, the F-measure values decreases. However, these decreases are not sharp in high-dimensional matrices. Additionally, the total CPU time of 10000 Monte-Carlo iteration is recorded to check the performance of the 10-fold cross validation method. Table 4.10 shows the total CPU times of different scenarios. There is no huge difference in required CPU times for different dimensional matrices.

Table 4.10: Comparison of total CPU's under 10000 Monte-Carlo runs based on different dimensional matrices and two different scenarios for the precision matrix.

Dimensions	Scenario 1	Scenario 2
(5×5)	4.02	4.04
(10×10)	3.62	3.98
(20×20)	3.69	3.52
(40×40)	3.71	3.90
(50×50)	4.45	5.24

4.3.2 GGM via Threshold Gradient Descent

As we mentioned before, TGD method penalizes the estimation of the off-diagonal elements of the precision matrix. For each type of matrix, initially, we use the identity matrix as the precision. Then we update the off-diagonal elements by using TGD steps which are explained in Section 3.2.2. Later, we estimate the diagonal entries of this updated matrix by using the Newton-Raphson algorithm.

In order to run TGD algorithm, first of all, we have to define the threshold parameter, λ and the increment, $\triangle\gamma$. The threshold parameter determines the sparsity of the estimated precision matrix and takes the value between 0 and 1. When it closes to 1, the estimated matrix becomes more sparse. Therefore, we choose three different λ values, namely, 0.3, 0.7 and 1, which correspond to sparse, moderately sparse and highly sparse matrices, respectively. Also, $\triangle\gamma$ is taken to be 10^{-3} in all Monte-Carlo runs in order to grid the estimation space comprehensively. In addition, the identity matrix is chosen as an initial matrix in TGD algorithm. Then, the off-diagonal elements of this matrix is updated by using TGD steps. After that, the Newton-Raphson algorithm is implemented to estimate diagonal elements of the precision matrix. To do this, we use *maxLik* function which is built under *maxLik* package of the R programming language. This function *maxLik(loglik,initial,method)* needs 3 inputs in its calculation, *loglik* indicates the log-likelihood function, *initial* represents initial values of parameters and *method* shows the maximization method which is Newton-Raphson in our case. The log-likelihood function can be represented as

$$\boldsymbol{\Theta}(\boldsymbol{\theta}^d, \boldsymbol{\theta}^o) = \frac{n}{2} \log |\boldsymbol{\Theta}| - \frac{1}{2} \sum_{k=1}^{n} \boldsymbol{X}^{(k)'} \boldsymbol{\Theta} \boldsymbol{X}^{(k)}. \tag{4.3}$$

Moreover, the diagonal elements of the previously updated matrix are assigned as initial values. In the first iteration, they equal to 1 since the identity matrix is implemented. However, if the precision matrix which is estimated by Newton-Raphson is not positive definited, the algorithm turns back to the beginning of the *maxLik* function and changes the initial values. Since Newton-Raphson is a general algorithm, it does not deal with the positive definited property of the precision matrix. Hence, to guarantee the positive definited precision, we decide on changing the initial values if we do not access the inner steps of the *maxLik* function.

Accordingly, in order to stay in the reasonable limits, we add small fluctuations to the diagonals and use these new values as initials of parameters as long as the new proposal initials satisfies the nonsingularity of the matrix. Here, the fluctuation terms that are added to the initials under singularity are randomly generated from the standard normal distribution.

Finally, the estimated precision matrices are converted to the binary form with threshold value, 0.01. Then, since the TGD method is very demanding in CPU time, the specificity and F-measure values are calculated for 3000 matrices. Also, we compute the specificity and F-measure values for 5 by 5, and 10 by 10 matrices under 10000 Monte-Carlo runs and we observe that there is no big difference in values between 3000 and 10000 runs and the change can be observable merely after five or six digit after the comma on the decimal side. The mean values of the specificity under 3000 Monte-Carlo runs are presented in Table 4.11 and Table 4.13. Moreover, Table 4.12 and Table 4.14 show the mean values of the F-measure under 3000 Monte-Carlo runs.

Table 4.11: Comparison of specificity under 3000 Monte-Carlo runs based on different dimensional matrices and different threshold values λ for only negative off-diagonal entries (second scenario).

Dimensions	$\lambda = 0.3$	$\lambda = 0.7$	$\lambda = 1$
(5×5)	0.4487	0.7917	0.9015
(10×10)	0.6190	0.9187	0.9771
(20×20)	0.7212	0.9691	0.9942
(40×40)	0.7902	0.9876	0.9990

Table 4.12: Comparison of F-measure under 3000 Monte-Carlo runs based on different dimensional matrices and different threshold values λ for only negative off-diagonal entries (second scenario).

Dimensions	$\lambda = 0.3$	$\lambda = 0.7$	$\lambda = 1$
(5×5)	0.6695	0.5851	0.5468
(10×10)	0.4770	0.4899	0.4959
(20×20)	0.3334	0.4583	0.4890
(40×40)	0.2351	0.4552	0.4934

Table 4.11, Table 4.12, Table 4.13 and Table 4.14 show that there are no differences in terms of specificity and F-measure values between two scenarios. On the other hand, when we look at the different threshold values for all dimensional matrices, it is seen that the specificity values increase as threshold value increase. Since our generated matrices are sparse, the threshold value which equals to 1 gives the best specificity and F-measure values.

When we compare the cross validation and the threshold gradient descent algorithms in terms

Table 4.13: Comparison of specificity under 3000 Monte-Carlo runs based on different dimensional matrices and different threshold values λ for only positive off-diagonal entries (first scenario).

Dimensions	$\lambda = 0.3$	$\lambda = 0.7$	$\lambda = 1$
(5×5)	0.4330	0.7896	0.9003
(10×10)	0.6194	0.9206	0.9772
(20×20)	0.7230	0.9698	0.9942
(40×40)	0.7951	0.9879	0.9990

Table 4.14: Comparison of F-measure under 3000 Monte-Carlo runs based on different dimensional matrices and different threshold values λ for only positive off-diagonal entries (first scenario).

Dimensions	$\lambda = 0.3$	$\lambda = 0.7$	$\lambda = 1$
(5×5)	0.6756	0.5882	0.5457
(10×10)	0.4761	0.4916	0.4962
(20×20)	0.3343	0.4588	0.4890
(40×40)	0.2373	0.4561	0.4934

of specificity, TGD method with threshold parameter 1 gives better results than the cross validation approach. Even if the threshold parameter is equal to 0.7, the results of TGD method are better than the ones for the cross validation method. When we compare these two algorithms in terms of F-measure, CV gives better results than TGD method for low dimensional matrices. However, for the high dimensional matrices, the specificity and F-measure values of both methods are close to each other.

On the other side, if we compare Table 4.10 and Table 4.15, a huge difference between the cross validation and the threshold gradient descent algorithms in terms of CPU times can be seen. Although total CPU's of TGD algorithm belongs to 3000 Monte-Carlo runs, it is higher than total CPU's of cross validation algorithm under 10000 Monte-Carlo runs. The reason of this great CPU requirement of TGD algorithm depends on the Newton Raphson algorithm which is used to estimate diagonal entries of the precision matrix. If the precision matrix which is estimated by Newton-Raphson is not positive definited, the algorithm turns back to the beginning of the *maxLik* function and changes the initial values. Thus, TGD algorithm requires many iterations to estimate diagonal entries.

As a result, in high-dimensional matrices, the cross validation method can be more convenient with less loss of accuracy but more gain in CPU time.

Table 4.15: Comparison of total CPU times under 3000 Monte-Carlo runs based on different dimensional matrices and two different scenarios for the precision matrix under threshold value, $\lambda = 1$.

Dimensions	**Scenario 1**	**Scenario 2**
(5×5)	15.643	16.409
(10×10)	25.587	24.878

4.4 Application of the JAK/STAT Pathway

4.4.1 Description of the JAK/STAT Pathway

When a living organism encounters with pathogens such as viruses, complex responses occur in host cells. So the defence mechanisms of the host cells decide on generations of the outcome as an antiviral response. The crucial parts of the innate antiviral response are Type I interferons (IFN). The interferons represent the family of small proteins generated by vertebrate cells in response to viral infection and as cytokines during an immune response (Lawrence, 2010). Accordingly, the Type I interferons are used to treat the hepatitis B and C virus infections (Maiwald et al., 2010). An important signalling pathway activated by IFN is the JAK/STAT pathway. Thereby, many immune disorders are related with disregulation of this signalling pathway. Here, JAK stands for the Janus kinase and STAT denotes the signal of the transducer and the activation of transcription. The JAK/STAT pathway provides a mechanism for transcriptional regulation by transmission of extracellular information from membrane to nucleus. The very simple representation of this pathway can be seen in Figure 4.1.

In mammalian, there are four members of the Janus kinase family, namely, JAK1, JAK2, JAK3 and tyrosine kinase 2(TYK2). STATs are proteins which are translocated to the nucleus and play the role of the transcription factors when they are phosphorylated by JAKs. There are seven STATs components, which are called as STAT1, STAT2, STAT3, STAT4, STAT5A, STAT5 and STAT6 in the system. Since the amino acid sequences of STATs are various, they have different functions in responses to extracellular signaling proteins. Due to its importance in particular immune system of living cells, it is widely worked on the treatment of illness (Shuai and Liu, 2003; Maiwald et al., 2010). The hepatitis B is one of the illnesses which is caused by the malfunction of the JAK/STAT pathway. Hereby, in this thesis, we take this system as the pathway to be modelled and inferred via GGM with distinct approaches. In the following parts, initially, we describe the simulated data used in this study since no real data have been available yet. Then, we perform in inference of the network by GGM in junction with cross validation and gradient descent algorithm.

4.4.2 Description of the Simulated Data

In this study, we generate a time-course dataset by using the Gillespie algorithm. In order to represent the stochastic behaviour of a system, the Gillespie algorithm uses the chemical master equation as described in Chapter 2. This algorithm simply generates a random value from the exponential distribution to determine the time of the next reaction, $\Delta t \sim exp(h_0(Y))$ (Purutçuoğlu et al., 2011). The summation of total hazards in the system, h_0, is used as the rate of the exponential distribution. Here, Y indicates the number of molecules for each protein and t denotes the change in time. After the time of the next reaction is calculated, the reaction type is chosen randomly. In the end, by using the time and reaction type, the system is updated.

Finally, in order to describe the JAK/STAT system, we generate a dataset which contains 10 time points for 38 proteins. The list of these proteins can be seen in Table 4.16. We run this algorithm until the total simulation time T set to 200 and take the last integer 10 time points from 190 to 199 so that all the proteins can reach in their steady state conditions and we get these values as the time-course data for the system. In simulation, initial number of molecules of each protein and the reaction rate constants are assigned as in Maiwald et al. (2010). To show the changes in states through the observed time points, we draw the plots for each protein. These plots are given in Appendix B.

Table 4.16: List of proteins used in the description of the JAK/STAT pathway as in the study of Maiwald et al. (2010).

Symbols	Name of proteins	Symbols	Name of proteins
P1	Receptor IFNAR1	P20	IRF9n
P2	TYK	P21	Free TFBS
P3	Receptor Tyk Complex	P22	Occupied TFBS
P4	Receptor IFNAR2	P23	mRNAn
P5	JAK	P24	mRNAc
P6	Receptor Jak Complex	P25	SOCS
P7	IFN_free	P26	Stat2n_IRF9
P8	IFNAR dimer	P27	STAT2n
P9	Active Receptor Complex_Stat2c	P28	CP
P10	STAT2c_IRF9	P29	ISGF-3c_CP
P11	Active Receptor Complex_STAT2c	P30	Stat1c*_Stat2c*_CP
P12	IRF9c	P31	NP
P13	STAT2c	P32	Stat1n*_Stat2n*_NP
P14	STAT1c	P33	ISGF-3n_NP
P15	Active Receptor Complex_STAT2c_STAT1C	P34	Occupied TFBS_NP
P16	STAT1c*_STAT2c*	P35	PIAS
P17	ISGF-3c	P36	PIAS_ISGF-3n
P18	ISGF-3n	P37	STAT1n
P19	STAT1n*_STAT2n*	P38	IFN_influx

4.4.3 Inference of the JAK/STAT Pathway via GGM

In this section, we apply the *k*-cross validation and threshold gradient descent algorithms to estimate the JAK/STAT system. Since there is no real data in literature about this pathway, the simulated data are preferred for the analysis. The description of simulated data is explicitly defined in previous section. The true precision matrix for the JAK/STAT pathway is generated by using the reaction list given in Maiwald et al. (2010). This precision matrix has the binary form such that the absence of an edge between two proteins is represented with zero entries. After the precision matrices are estimated by the *k*-cross validation and TGD methods, we convert these matrices into a binary form via the same threshold value, i.e., 0.01. Then, by comparing true and estimated precision matrices, TP, TN, FP and FN values are calculated. Finally, we compute the specificity values. The results of the threshold gradient descent algorithm based on the three different threshold values are be shown in Table 4.17. Since the JAK/STAT system is sparse, the threshold value which equals to one gives the best specificity values. On the other side, the specificity value based on the cross validation algorithm is computed as 0.9848.

Indeed in order to represent the complex biological systems like the JAK/STAT pathway by relatively small dimension, the coloured graphs can be used. Here, to obtain a sparse network, a special kind of the coloured graphs, called the *factorial graphs* (Abbruzzo, 2012), is implemented. This method creates subpartitions of the naturally partitioned networks in such a way that the edges indicated with the same colours represent the specific elements of the conditional correlation matrix. By this way, the number of parameter estimated decreases as the coloured sub-graph is used as the summary of the actual partitioned network.

Table 4.17: Comparison of specificity for the JAK/STAT pathway based on three different threshold values λ via TGD algorithm.

Dimensions	$\lambda = 0.3$	$\lambda = 0.7$	$\lambda = 1$
(38×38)	0.9440	0.9660	0.9700

4.5 Application of the MAPK/ERK Pathway

4.5.1 Description of the MAPK/ERK Pathway

The MAPK/ERK (mitogen-activated protein kinase, also called ERK, i.e. extracellular signal-regulated kinase) pathway is another important signal transduction system that transmits the signals from the surface of the cell to the nucleus, resulting in changing the gene expression levels on the pathway. This pathway affects the growth control and the cell death of all eukaryotes (Purutçuoğlu and Wit, 2012). Thereby, the malfunction of this network causes many disorders in the cell such as cancer. The MAPK/ERK pathway is a large system which includes many proteins whose main components are Raf, Ras, MEK and ERK proteins. The very simple representation of this pathway can be seen in Figure 4.2. More biological details about the system can be found in Purutçuoğlu and Wit (2008; 2012).

4.5.2 Inference of the MAPK/ERK Pathway via GGM

In this study, we analyze the MAPK/ERK pathway by using a real time-course dataset. The dataset is taken from Purutçuoğlu and Wit (2012) and is collected under 8 time-points for 6 proteins, namely, Ras.GTP, Ras.GDP, Raf.A_m, MEK.p2, ERK and ERK.p2 whose biological descriptions are presented in Purutçuoğlu and Wit (2008; 2012). Here, the k-cross validation and the threshold gradient descent algorithms are applied to estimate this pathway. As implemented in previous studies, the binary form of the true precision matrix is constructed by looking at the edges between proteins that is generated from the quasi reaction list of the system. On the other side, the estimated precision matrix is converted into a binary form via the threshold value. Then, the accuracy measures, namely, specificity and F-measure, are computed by comparing true and estimated precision matrices of the k-cross validation and TGD method.

By using TGD algorithm, the specificity and F-measure values are computed as 0.9130 and 0.5720, respectively. On the other hand, the specificity and F-measure values based on the cross validation algorithm equal 1 and 0.2670, respectively. From the estimation, it is observed that the cross validation method seems better in terms of the specificity value. However, the difference between these two methods is not significant. In addition, F-measure in cross validation is significantly smaller than F-measure in TGD. To sum up, when we check both accuracy measures simultaneously, we conclude that TGD algorithm gives better results.

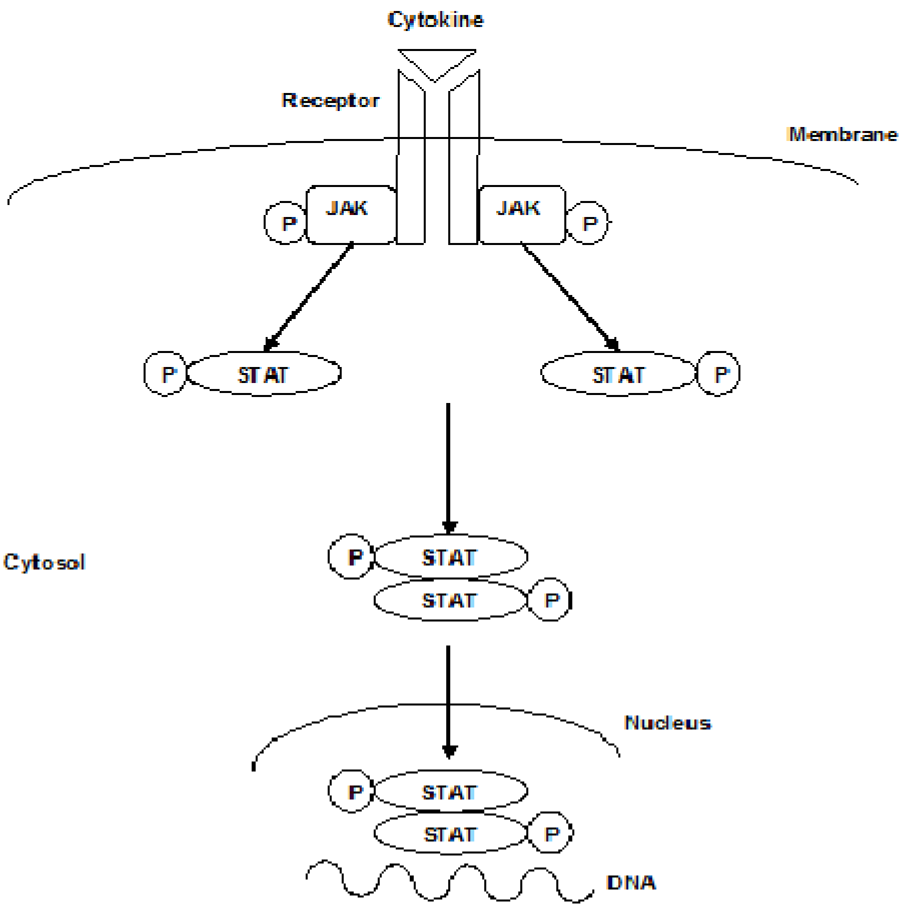

Figure 4.1: Simple representation of the structure of the JAK/STAT pathway.

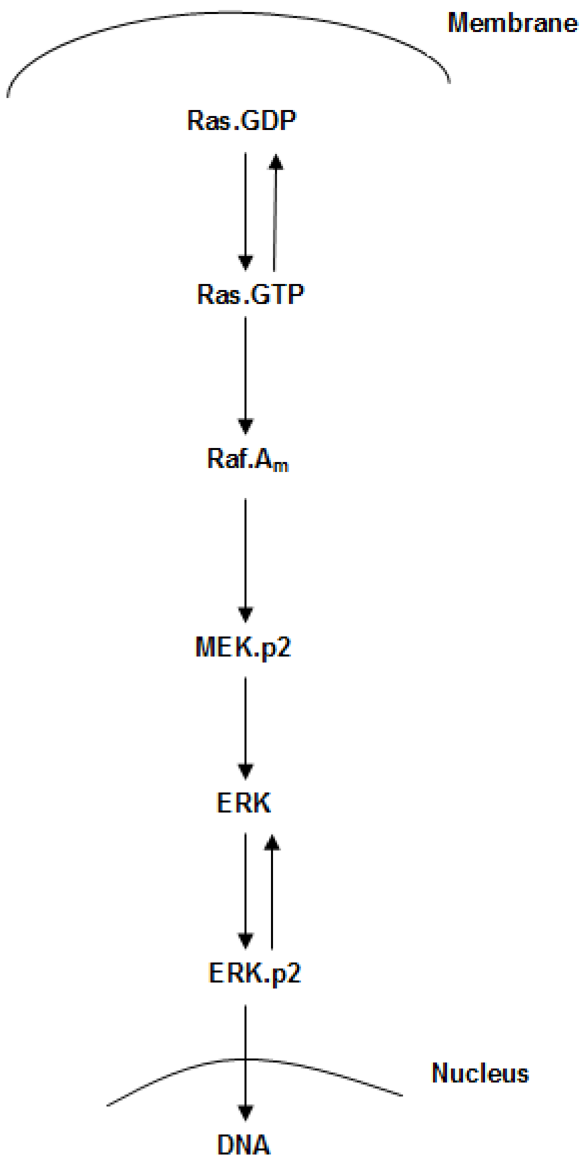

Figure 4.2: Simple representation of the structure of the MAPK/ERK pathway.

CHAPTER 5

CONCLUSION AND OUTLOOK

In this study, it is aimed to model the complex biochemical systems by using the Gaussian graphical model. In order to estimate the true structure of the network, GGM is widely used for the undirected graphical model. In GGM, the precision matrix which is the inverse of covariance matrix is used to represent the structure of the network. The zero entries in the precision matrix indicate the conditional independencies of two nodes, i.e., genes, molecules. Whereas, since the biochemical systems have high-dimensions, the singularity problem usually occurs in the calculations of the precision matrix.

In the first part of this thesis, the background information about three main topics are given. First of all, the most well-known matrix decomposition methods, namely, eigenvalue decomposition, singular value decomposition and Cholesky decomposition methods are explained. Then, several methods which are used to model biological systems are described. In addition, Gaussian graphical model, which is our main interest in this study, is explained in details. After that, the well-known accuracy measures are given with explicit definitions and formulas.

In the second part of this study, initially, our suggested approach to unravel the singularity problem is explained. Then, the k-cross validation and the threshold gradient descent algorithms are implemented in GGM to estimate the precision matrix. These two methods have been suggested in the literature, but they have not been applied in GGM context yet. The cross validation method is used to decide on the best penalty parameter, λ which is necessary in the estimation of the precision matrix via the L_1-penalized likelihood function. We use the 10-fold cross validation method. This method splits the data into 10 parts, then it chooses 9 of them as a training set and uses remaining one as a test set. In the end, the precision matrix is estimated via the L_1-penalized likelihood approach with he chosen penalty. On the other side, TGD algorithm applies the gradient loss function which is the negative of the log-likelihood function to estimate off-diagonal entries of the precision matrix. Moreover, the Newton Raphson algorithm is applied to estimate the diagonal entries. The adapted R codes of both algorithms are presented in Appendix.

In the third part of this study, firstly, the performance of our new method is compared with three decomposition methods, namely, eigenvalue decomposition, singular value decomposition and Cholesky decomposition in terms of the computational time. These decomposition methods are commonly used in the literature. There are two main advantages of our new method. The first one is that by saving the linear relationship between columns, our alternative method preserves the structure of the original entries of matrix. The second advantages is that when the percentage of dependent columns and dimension of the matrix increase, our suggested method becomes more efficient. Moreover, when the percentage of dependent columns is not high, our proposed method is comparable to the other three methods.

Secondly, the best model selection criteria is chosen by comparing several accuracy measures, namely, precision, recall, specificity, false positive rate, false discovery rate, accuracy, F-measure and Matthews correlation coefficient. All of these measures are the functions of true positive, true negative, false positive and false negative values. In order to achieve this aim, first of all, the precision matrices are estimated by using the L_1-penalized likelihood approach with different parameters. Then, the true and estimated precision matrices are converted into a binary form via a threshold value. Hence, all accuracy measures are performed for different matrix dimensions. In the end, the specificity measure is chosen as the best model selection criterion.

After that, the k-cross validation and the threshold gradient descent methods are compared in terms of accuracy and computational time by using Monte-Carlo runs. Then, we conclude that TGD algorithm is better than the cross validation algorithm in terms of the specificity for each dimensional matrix. However, TGD method requires more CPU time. In high dimensional matrices, the specificity and F-measure values of both methods are close to each other. So, the implementation of the cross validation method can be more appropriate.

In the end, the cross validation and TGD algorithms are performed in estimation of the JAK/STAT pathway which is an important signalling pathway activated by Type I interferon. The Type I interferon is used to treat the hepatitis B and C virus infections. Hence, the disregulation of the JAK/STAT pathway causes many immune disorders. So, it is important to create network to better understand this pathway. Since there is no real data in the literature about the underlying system, the simulated data are used in our study. By using Gillespie algorithm that is based on the chemical master equation and enables us to successfully generate the stochastic manner of the system, we generate a time-course dataset which contains 10 time points for 38 proteins. In simulation of the dataset via Gillespie, the necessary reaction list, reaction rate constants and initial number of molecules for each protein are taken from Maiwald et al. (2010). Then, in order to check the performance of the methods under real data, we use the brief description of the MAPK/ERK pathway that has 8 time points for 6 proteins. This pathway is one of the well-known systems in cellular growth control of all eukaryotes and is typically worked on in details for oncogene researches. From both applications we observe that, similar to previous findings, the threshold gradient descent algorithm estimates the true quasi structure of the pathway better than the k-cross validation method within GGM and the specificity measure can successfully capture the plausible penalty value that specifies the inferred structure of the system.

As the future study, we consider to estimate the complex systems via GGM under distinct level of singularity problem by combining our suggested approach which can deal with the singularity problem with the current codes for GGM. In addition, the Newton Raphson algorithm which is used in TGD method can be improved to gain in CPU time. Because in the current version of the Newton-Raphson algorithm, the singularity problem is not handled smartly and the algorithm can stuck even in the first iteration if the initial values cause singularity challenge. Therefore, if the algorithm can suggest new nonsingular estimators, the speed of TGD can increase significantly.

Finally, we believe that the way to deal with the inference problem in GGM can be also solved via the logistic regression method, in which the connection of genes can be represented via conditional probabilities of all other genes. We consider that this approach can be plausible at least for small networks.

REFERENCES

[1] Abbruzzo, A. (2012). *Graphical Models for Estimating Dynamic Network*. Ph.D Thesis, University of Groningen, Netherlands.

[2] Arslan, H. (2011). *Machine Learning Methods for Promoter Region Prediction*. Master Thesis, Graduate School of Natural and Applied Sciences, Computer Engineering Department, METU, Ankara, Turkey.

[3] Ayyıldız, E., Purutçuoğlu, V., and Wit, E. (2012). *A Short Note on Resolving Singularity Problems in Covariance Matrices*. International Journal of Statistics and Probability, 1 (2), 113-118.

[4] Barabasi, A. L., and Oltvai, Z. N. (2004). *Network Biology: Understanding the Cell's Functional Organization*. Nature Reviews Genetics, 5, 101–113.

[5] Bolouri, H. (2008). *Computational Modeling of Gene Regulatory Networks: a Primer*. Imperial College Press.

[6] Bower, J. M., and Bolouri, H. (2001). *Computational Modeling Genetic and Biochemical Networks*. Cambridge, Massachusetts Institute of Technology Press.

[7] Chib, S., Nardari, F., and Shephard, N. (2006). *Analysis of High Dimensional Multivariate Stochastic Volatily Models*. Journal of Econometrics, 134, 341–371.

[8] Defterli, O., Purutçuoğlu, V., and Weber, G.-W. (2013). *Advanced Mathematical and Statistical Tools in the Dynamic Modelling and Simulation of Gene-Environment Networks*, preprint: 1-22. Chapter in: Modeling, Optimization, Dynamics and Bioeconomy. Editor: D. Zilberman and A. Pinto. Springer-Verlag.

[9] Fan, J., Feng, Y., and Wu, Y. (2009). *Network Exploration via the Adaptive Lasso and Scad Penalties*. The Annuals of Applied Statistics, 3, (2), 521–541.

[10] Friedman, J. H., Hastie, T., and Tibshirani, R. (2008). *Sparse Inverse Covariance Estimation with the Graphical Lasso*. Biostatistics, 9, (3), 432–441.

[11] Friedman, J. H., and Popescu, B. E. (2004). *Gradient Directed Regularization*. Technical Report, Stanford University.

[12] Galleger, R. G. (1996). *Discrete Stochastic Processes*. Boston, Kluwer Academic Publishers.

[13] Gillespie, D. T. (2001). *Approximate Accelerated Stochastic Simulation of Chemically Reacting Systems*. Journal of Chemical Physics, 115, 4, 1716–1733.

[14] Goeman, J. J. (2010). *L_1 Penalized Estimation in the Cox Proportional Hazards Model*. Biometrical Journal, 52, 1, 70-84.

[15] Golightly, A., and Wilkinson, D. J. (2005). *Bayesian Inference for Stochastic Kinetic Models Using a Diffusion Approximation*. Biometrics, 61 (3), 781–788.

[16] Healy, M. J. R. (1986). *Matrices for Statistics*. Oxford, Oxford University Press.

[17] Johnson, R. A., and Wichern, D. W. (2002). *Applied Multivariate Statisticsl Analysis*. Upper Saddle River, N.J., Prentice Hall.

[18] Jong, H. D. (2002). *Modeling and Simulation of Genetic Regulatory Systems: A Literature Review*. Journal of Computational Biology, 9 (1), 67–103.

[19] Kampen, N. G. Van. (2007). *Stochastic Processes in Physics and Chemistry* . Amsterdam, Elsevier.

[20] Kutner, M. H., Nachtsheim, C. J., Neter, J., and Li, W. (2005). *Applied Linear Statistical Models*. Upper Saddle River, N.J., Prentice Hall.

[21] Labatut, V., and Cherifi, H. (2011). *Accuracy Measures for the Comparison of Classifiers*. 5th International Conference on Information Technology, Amman, Jordan.

[22] Lawrence, E. (2008). *Henderson's Dictionary of Biology*. New York, Pearson Benjamin Cummings Prentice Hall.

[23] Lawrence, E., Girolami, M., Rattray, M., and Sanguinetti, G. (2010). *Learning and Inference in Computational Systems Biology*. Cambridge, Massachusetts Institute of Technology Press.

[24] Li, H., and Gui, J. (2006). *Gradient Directed Regularization for Sparse Gaussian Concentration Graphs with Applications to Inference of Genetic Networks*. Biostatistics, 7, 2, 302–317.

[25] Maiwald, T., Schneider, A., Busch, H., Sahle, S., Gretz, N., Weiss, TS., Kummer, U., and Klingmüller, U. (2010). Combining Theoretical Analysis and Experimental Data Generation Reveals IRF9 as a Crucial Factor for Accelerating Interferon α-Induced Early Antiviral Signalling. The FEBS Journal, 277 (22), 4741–54.

[26] O'Shea, J. J., Gadina, M., and Schreiber, R. D. (2002). *Cytokine Signaling in 2002: New Surprises in the Jak/Stat Pathway*. Cell, 109, 121–131.

[27] Øksendal, B.K. (2003). *Stochastic Differential Equations: An Introduction with Application*. Springer, Berlin.

[28] Purutçuoğlu, V. (2007). *Bayesian Methods for Gene Network Analysis*. Ph.D Thesis, Mathematic and Statistics Department, Lancaster University, United Kingdom.

[29] Purutçuoğlu, V. (2011). *Stochastic Modelling and Parameter Estimation of the HCV Network*. Proceeding of the 16th INFORMS Applied Probability Conference, Stockholm, Sweden, 81–82.

[30] Purutçuoğlu, V. (2012). *Inference of the Stochastic MAPK Pathway by Modified Diffusion Bridge Method*. Central European Journal of Operations Research, 1–15.

[31] Purutçuoğlu, V. (2012). *Estimating Network Kinetics of the MAPK/ERK Pathway Using Biochemical Data*. Mathematical Problems in Engineering, 1–34.

[32] Purutçuoğlu, V., and Ayyıldız, E. (2013). *Mathematical Modelling of Biological Networks* (in Turkish), preprint: 1-26. Chapter in: Bioinformatics II: Application Areas. Editor: Allmer, J. Nobel Publisher.

[33] Purutçuoğlu, V., Erdem, T., and Weber, G.-W. (2011). *Inference of the JAK/STAT Gene Network via Graphical Models*. Proceeding of the 23rd International Conference on Systems Research, Informatics and Cybernetics, Baden, Germany, 46–50.

[34] Purutçuoğlu, V., and Wit, E. (2006). *Exact and Approximate Stochastic Simulation of the MAPK Pathway and Comparisons of Simulations' Results*. Journal of Integrative Bioinformatics, 3, 231-243.

[35] Purutçuoğlu, V., and Wit, E. (2008). *Bayesian Inference for the MAPK/ERK Pathway by Considering the Dependency of the Kinetic Parameters*. Bayesian Analysis, 3(4), 851–886.

[36] Purutçuoğlu, V., and Wit, E. (2012). *Estimating Network Kinetics of the MAPK/ERK Pathway using Biochemical Data*. Mathematical Problems in Engineering, 1–34.

[37] Rencher, A. C. (2002). *Methods of Multivariate Analysis*. New York, John Wiley and Sons.

[38] Ross, S. M. (1996). *Stochastic Processes*. New York, Wiley.

[39] Santos, E.E, and Chu, P. (2003). *Efficient and Optimal Parallel Algorithms for Cholesky Decomposition*. Journal of Mathematical Modelling and Algorithms, 2, 217–234.

[40] Shmulevich, I., and Dougherty, E.R. (2010). *Probabilistic Boolean Networks: The Modeling and Control of Gene Regulatory Networks*. Philadelphia, Society for Industrial and Applied Mathematics.

[41] Shuai, K., and Liu, B. (2003). *Regulation of JAK/STAT Signalling in the Immune System*. Nature Reviews Immunology, 3, 900–911.

[42] Strimmer, K., and Schäfer, J. (2005). *A Shrinkage Approach to Large-Scale Covariance Matrix Estimation and Implication for Functional Genomics*. Statistical Applications in Genetics and Molecular Biology, 10 (1), Article 24.

[43] Streib, F. E., and Dehmer, M. (2008). *Analysis of Microarray Data: A Network-Based Approach*. Weinheim, Wiley.

[44] Tibshirani, R. (1996). *Regression Shrinkage and Selection via the Lasso*. Journal of the Royal Statistical Society, 58, 1, 267–288.

[45] Turner, T. E., Schnell, S., and Burrage, K. (2004). *Stochastic Approaches for Modelling in Vivo Reactions*. Computational Biology and Chemistry, 28, 165–178.

[46] Vera, J., Bachmann, J., Pfeifer, A.C., Becker, V., Hormiga, J.A., Darias, N.V.T., Timmer, J., Klingmüller, U., and Wolkenhauer, O. (2008). *A Systems Biology Approach to Analyse Amplification in the JAK2/STAT5 Signalling Pathway*. BMC Systems Biology, 2, 38, 1–13.

[47] Watkins, D. S. (1991). *Fundamentals of Matrix Computations*. New York, John Wiley and Sons.

[48] Whittaker, J. (1990). *Graphical Models in Applied Multivariate Statistics*. New York, John Wiley and Sons.

[49] Wilkinson, D. J. (2006). *Stochastic Modelling for Systems Biology*. Boca Raton, FL, Taylor and Francis.

[50] Wit, E., Vinciotti, V., and Purutçuoğlu, V. (2013). *Statistics for Biological Networks: How to Infer Networks from Data*. Chapman and Hall/CRC (in publication).

[51] Wit, E., Vinciotti, V., and Purutçuoğlu, V. (2010). *Statistics for Biological Networks: Short Course Notes*. 25th International Biometric Conference (IBC), Florianopolis, Brazil.

[52] Witten, D., Friedman, J., and Simon, N. (2011). *New Insights and Faster Computations for the Graphical Lasso*. Journal of Computational and Graphical Statistics, 20 (4), 892–900.

APPENDIX A

R CODES

A.1 R CODES FOR NEW METHOD TO SOLVE SINGULARITY

The following R codes are used to solve singularity problem of a matrix with our new method. *Omitsingular* function takes a matrix called *diffusion* as an input covariance matrix and return its square root as an output of the *Omitsingular* function. The output of the function is called as *sqrt.diffusion.prop* in these codes.

```
omitsingular < − function(diffusion){
          out.prop< − NULL
          dependent.structure.prop < − NULL
```

```
          Identifying the linearly dependent columns
```

```
          for(h1in 2:ncol(diffusion)){
               rank.diffus.prop < − qr(diffusion[,(1:h1)%w/o%out.prop])$rank
               if(rank.diffus.prop ! = h1-length(out.prop)){
                     cols.A.prop < −(1:(h1-1))%w/o%out.prop
                     A.prop < − diffusion[,cols.A.prop]
                     b.prop < − diffusion[,h1]
                     dependent.structure.prop[[h1]] < − rep(0, ncol(diffusion))
                     dependent.structure.prop[[h1]][cols.A.prop]< −
                     round(qr.solve(A.prop,b.prop),5)
                     out.prop < − c(out.prop, h1)
               }
          }
```

```
          Obtaining low dimensional matrix
```

```
          new.subst.prop < − ncol(diffusion) - length(out.prop)
          rest.prop < −(1:ncol(diffusion))%w/o%out.prop
          diffusion.prop < − diffusion[rest.prop,rest.prop]
```

```
        sqrt.diffusion.prop < −chol(diffusion.prop)
        return(sqrt.diffusion.prop)
}
```

A.2 R CODES FOR *K*-CROSS VALIDATION ALGORITHM

The following R codes are used to compute the *k*-cross validation algorithm. *CrossValidation* function takes a matrix called *observe* as an input observation matrix and takes a vector called *penalty* as an input penalty scores. The *CrossValidation* function returns the best penalty, which maximizes the likelihood function, as an output. The output of the function is called as *best.penalty* in these codes.

```
CrossValidation < −function(observe,penalty){
    size.all< −dim(observe)
    n< −size.all[1]
    p< −size.all[2]
    training.mat< −matrix(0,nrow=(n*0.9), ncol=p)
    test.mat< −matrix(0,nrow=(n*0.1), ncol=p)
    likelihood< −rep(0,length=length(penalty))
    for(i3in 1:length(penalty))
        {
        test.likelihood< −rep(0,length=(n*0.1))
```

Obtaining the test and the training sets

```
        for(i4 in seq(1,n,(n*0.1)))
            {
            test.mat< −observe[i4:(i4+n*0.1-1),]
            training.mat< −observe[-(i4:(i4+n*0.1-1)),]
            cov.mat< −cov(trainingt.mat)
```

Estimating the precision matrix and choosing the best penalty

```
            train.glasso< −glasso(cov.mat,rho=penalty[i3],penalize.diagonal=F)
            if(is.na(train.glasso$loglik))
                { test.likelihood[i4]< −0}
            else
                {train.pre< −test.glasso $wi
                train.mean< −apply(training.mat,2,mean)
                train.mean< −matrix(train.mean,nrow=1,ncol=5)
                tt< −as.matrix(training.mat-test.mean)
                cov.precision< −1/(n*0.1)*tt%*%train.pre%*%t(tt)
                test.likelihood[i4]< −(n*0.1)/2*log(det(train.pre))-(n*0.1)
                                        2*sum(diag(cov.precision))}
            }
        likelihood[i3]< −sum(test.likelihood)
        }
        max.likelihood< −max(likelihood)
        ind< −which(likelihood==max.likelihood)
        best.penalty< −penalty[ind]
        return(best.penalty)
}
```

A.3 R CODES FOR THRESHOLD GRADIENT DESCENT ALGORITHM

The following R codes belongs to threshold gradient descent algorithm. *GradientDescent* function takes a matrix called *observation* as an input observation matrix and return the estimated precision matrix as an output of the *GradientDescent* function. The output of the function is called as *estimated.precision* in these codes.

```
GradientDescent < −function(observation){
        X< −observation
        size.all< −dim(observation)
        n< −size.all[1]
        p< −size.all[2]
        gamma< −0
        delta.gamma< −0.001
        lambda< −0.3
        q< −p*(p-1)/2
```

Calculating log-likelihood function

```
        loglike < −function(param){
                diag(prop.prec)< −param
                l1< −n/2*log(det(prop.prec))-1/2*sum(diag(X%*%prec%*%t(X)))
                l1}
        prec< −diag(p)
        sumX< −matrix(0,p,p)
```

Updating the precision matrix by using the gradient descent step

---

```
for(i2 in 1:n){
      sumX< −sumX+X[i2,]%*%t(X[i2,])}
for(i1 in 1:200){
      g< −1/2*sumX-(n/2)*solve(w)
      goffdiag< −g[upper.tri(g,diag=FALSE)]
      offdiag< −prec[upper.tri(g,diag=FALSE)]
      w< −matrix(0,1,q)
      for(i3in 1:q) {
           if(abs(goffdiag[i3])>=lambda*max(abs(goffdiag)))
                 w[i3]< −1
           else
                 w[i3]< −0 }
      d< −w*goffdiag
      prec.off< −offdiag+delta.gamma*d
      c< −matrix(0,nrow=p,ncol=p)
      c[upper.tri(c,diag=FALSE)]< −prec.off
      e< −c+t(c)
      diag(e)< −diag(prec)
      prec< −e
      gamma< −delta.gamma+gamma
      condition< −-1
      while(condition<=0){
            prop.prec< −prec
            eigen.initial< − -10
            while(eigen.initial<0){
                  eigen.initial< −0
                  det.check< −det(prop.prec)
                  if(sum(det.check<=0)! =0){
                        fluc< −rnorm(p,0,1)
                        start.var< −diag(prop.prec)+fluc
                        diag(prop.prec)< −start.var
                        eigen.initial< − -10 }
                  else{
                        eigen.initial< −10 }
            }
            diag.start< −diag(prop.prec)
            est.diag< −maxLik(loglike,start=diag.start,method="nr")
            diag(prop.prec)< −coef(est.diag)
            if(det(prop.prec)>0){
                 condition< −1
                 prec< −prop.prec }
            else{
                 condition< −-1 }
```

```
                }
            }
            estimated.precision< −prec
            return(estimated.precision)
}
```

APPENDIX B

GRAPHICS FOR PROTEINS

The number of molecules is generated from the stochastic simulation of the JAK/STAT pathway via the Gillespie algorithm. The results are taken as the last 10 integer time points under the total time $T = 200$ (i.e., t =191, 192, 193, 194, 195, 196, 197, 198, 199 and 200) in the Gillespie outputs. The full names of proteins are presented in Table 4.16 in the main study.

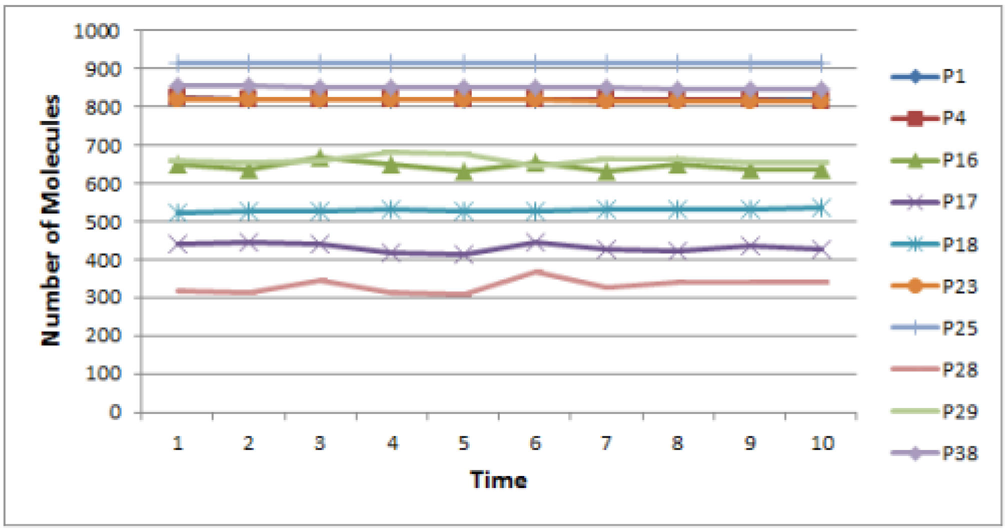

Figure B.1: Changes in the number of molecules for proteins P1, P4, P16, P17, P18, P23, P25, P28, P29 and P38.

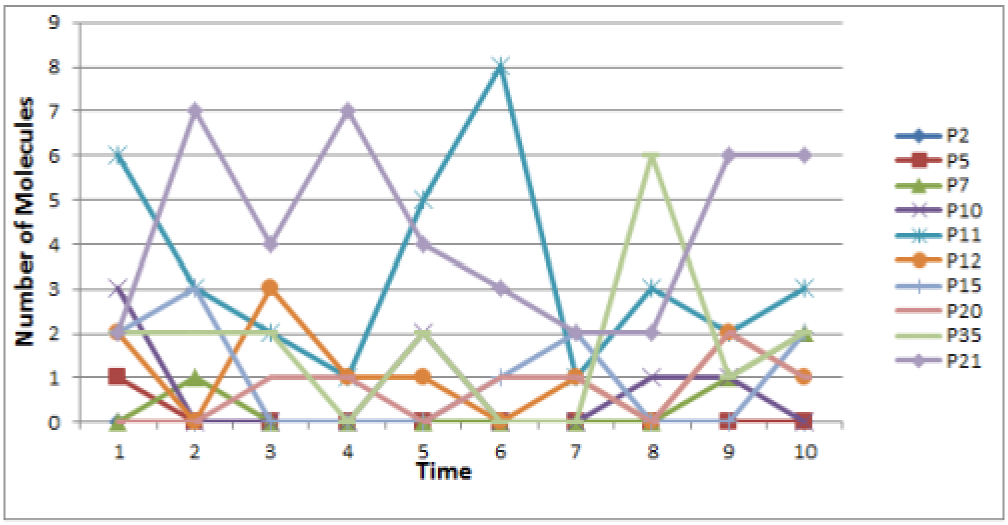

Figure B.2: Changes in the number of molecules for proteins P2, P5, P7, P10, P11, P12, P15, P20, P21 and P35.

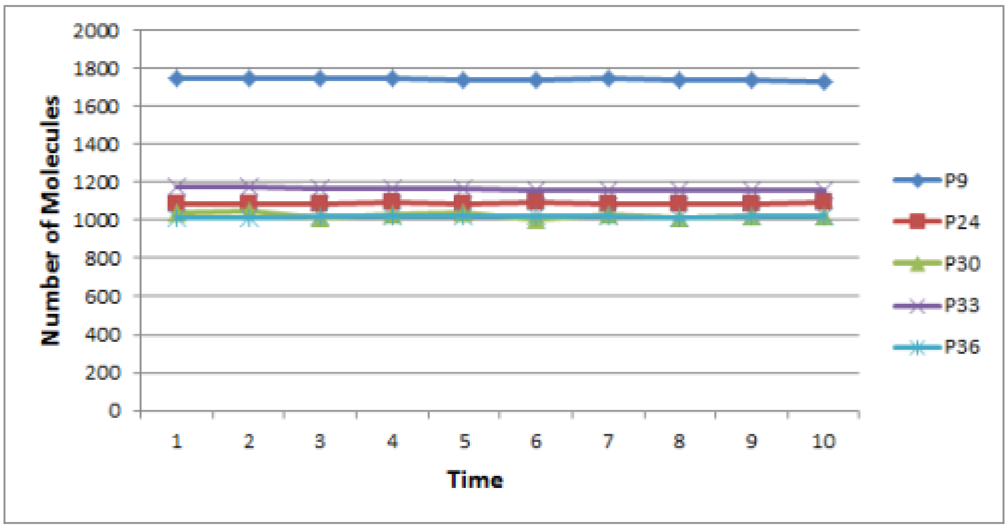

Figure B.3: Changes in the number of molecules for proteins P9, P24, P30, P33, and P36.

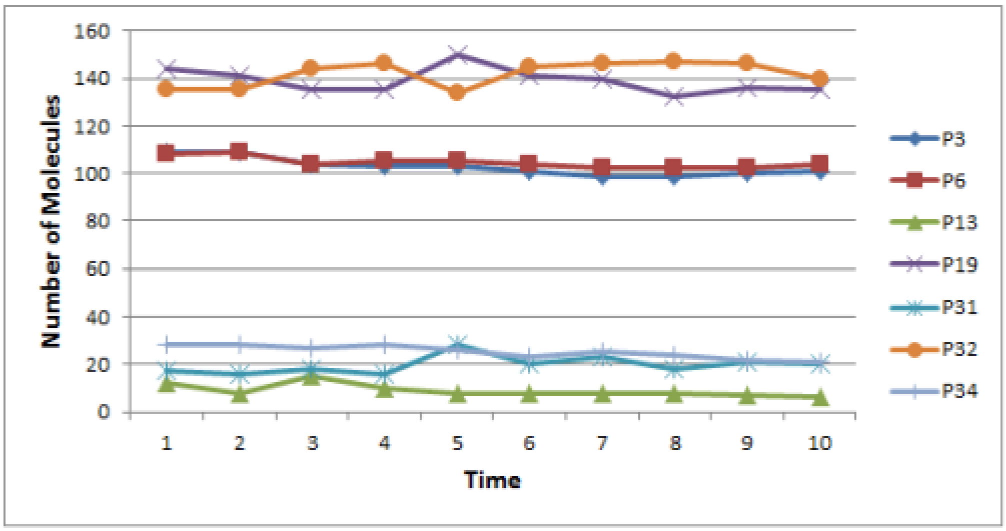

Figure B.4: Changes in the number of molecules for proteins P3, P6, P13, P19, P31, P32 and P34.

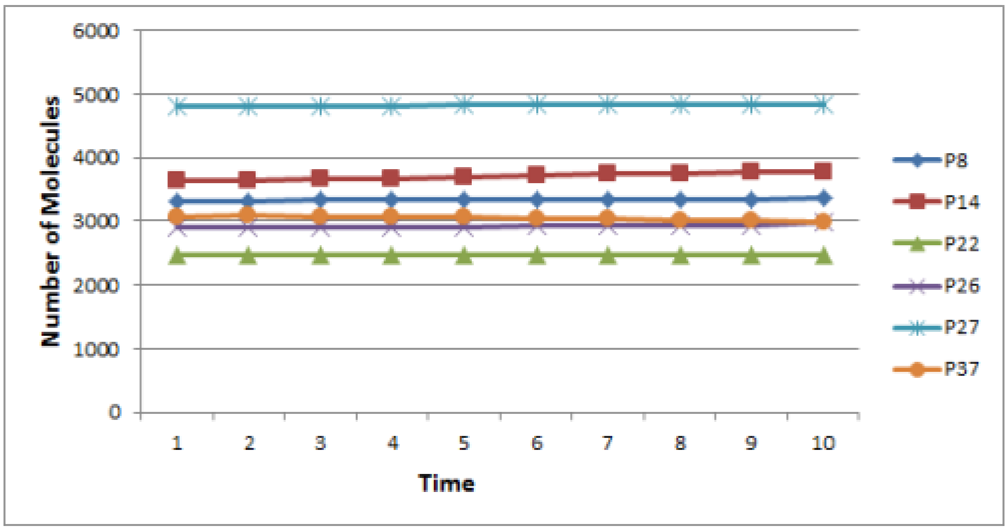

Figure B.5: Changes in the number of molecules for proteins P8, P14, P22, P26, P27 and P37.