EFFICIENT AND ROBUST ALGORITHMS FOR STATISTICAL INFERENCE IN GENE REGULATORY NETWORKS

A Dissertation

by

AMINA NOOR

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Chair of Committee,	Erchin Serpedin
Co-Chair of Committee,	Mohamed Nounou
Committee Members,	Byung Jun Yoon
	Aydin I. Karsilayan
	Tiffani L. Williams
Head of Department,	Chanan Singh

December 2013

Major Subject: Electrical Engineering

ABSTRACT

Inferring gene regulatory networks (GRNs) is of profound importance in the field of computational biology and bioinformatics. Understanding the gene-gene and gene-transcription factor (TF) interactions has the potential of providing an insight into the complex biological processes taking place in cells. High-throughput genomic and proteomic technologies have enabled the collection of large amounts of data in order to quantify the gene expressions and mapping DNA-protein interactions.

This dissertation investigates the problem of network component analysis (NCA) which estimates the transcription factor activities (TFAs) and gene-TF interactions by making use of gene expression and Chip-chip data. Closed-form solutions are provided for estimation of TF-gene connectivity matrix which yields advantage over the existing state-of-the-art methods in terms of lower computational complexity and higher consistency. We present an iterative reweighted ℓ_2 norm based algorithm to infer the network connectivity when the prior knowledge about the connections is incomplete.

We present an NCA algorithm which has the ability to counteract the presence of outliers in the gene expression data and is therefore more robust. Closed-form solutions are derived for the estimation of TFAs and TF-gene interactions and the resulting algorithm is comparable to the fastest algorithms proposed so far with the additional advantages of robustness to outliers and higher reliability in the TFA estimation.

Finally, we look at the inference of gene regulatory networks which which essentially resumes to the estimation of only the gene-gene interactions. Gene networks are known to be sparse and therefore an inference algorithm is proposed which imposes a sparsity constraint while estimating the connectivity matrix. The online estimation lowers the computational complexity and provides superior performance in terms of accuracy and scalability.

This dissertation presents gene regulatory network inference algorithms which provide computationally efficient solutions in some very crucial scenarios and give advantage over the existing algorithms and therefore provide means to give better understanding of underlying cellular network. Hence, it serves as a building block in the accurate estimation of gene regulatory networks which will pave the way for finding cures to genetic diseases.

DEDICATION

To my mother.

ACKNOWLEDGEMENTS

Praise be to God, the Cherisher and Sustainer of the Worlds. I am very grateful for His infinite blessings and mercy.

I extend my special thanks to my advisors Dr. Erchin Serpedin and Dr. Mohamed Nounou, for their guidance and support. I am particularly indebted to Dr. Serpedin for his constant motivation and encouragement. I am also grateful to Dr. Byung-Jun Yoon, Dr. Aydin I. Karsilayan and Dr. Tiffani L. Williams for serving on my committee and providing valuable advice.

I would like to thank my colleagues at the Department of Electrical and Computer Engineering particularly, Sabit, Bilal, Ali, Fang-Han, Sang-Woo, Yi and Aaron who provided a nice and friendly work environment. In addition, I want to thank my friends in College Station, Amina and Sevgi, who have made my stay here a wonderful experience.

In the end, I want to thank my family for their continued support, encouragement and prayers. I am highly indebted to my parents for their immeasurable sacrifices. I also want to thank Uncle Mansoor for his help and guidance. I am extremely grateful to my in-laws and extended family, particularly my grandmother, Aunty Tahmina and Uncle Ishtiaq for their prayers and support. I would like to extend my heart-felt thanks to my brother, Umar, and my sister, Mariyum, for their love, prayers and confidence in me. I extend my immense gratitude to my husband, Aitzaz, for his unconditional support and for always standing by me - thank you for everything, I am forever grateful to you!

TABLE OF CONTENTS

LIST OF FIGURES

FIGURE Page

LIST OF TABLES

TABLE Page

1. INTRODUCTION

The post-genomic era is marked by the availability of a deluge of genomic data that has enabled researchers to look towards new dimensions for understanding the complex biological processes governing the life of a living organism [22,32,36,82,83]. The various life-sustaining functions are performed via a collaborative effort involving DNA, RNA and proteins. Genes and proteins interact with themselves and each other and orchestrate the successful completion of a multitude of important tasks. Understanding how they work together to form a cellular network in a living organism is extremely important in the field of molecular biology. Two important problems in this considerably nascent field of computational biology are the inference of gene regulatory networks and the estimation of transcription factor activities (TFAs).

Gene regulation is one of the many fascinating processes taking place in an living organism whereby the expression and repression of genes are controlled in a systematic manner. With the help of the enzyme RNA polymerase, DNA transcribes into mRNA which may or may not translate into proteins. It is found that in certain special cases mRNA is reverse transcribed to DNA. The processes of transcription and translation are schematically represented in Fig. 1.1, where the interactions in black show the most general framework and the interactions depicted in red occur less frequently. Transcription factors (TFs), which are a class of proteins play the significant role of binding onto the DNA and thereby regulate their transcription. Since the genes may be coding for TFs and/or other proteins, a complex network of genes and proteins is formed. The level of activity of a gene is measured in terms

*Part of this section is reprinted from "An Overview of the Statistical Methods Used for Inferring Gene Regulatory Networks and Protein-Protein Interaction Networks," Amina Noor, Erchin Serpedin, Mohamed Nounou, Hazem Nounou, Nady Mohamed, and Lotfi Chouchane, Advances in Bioinformatics, 2013, doi:10.1155/2013/953814, 2013. Copyright 2013, Amina Noor et al.

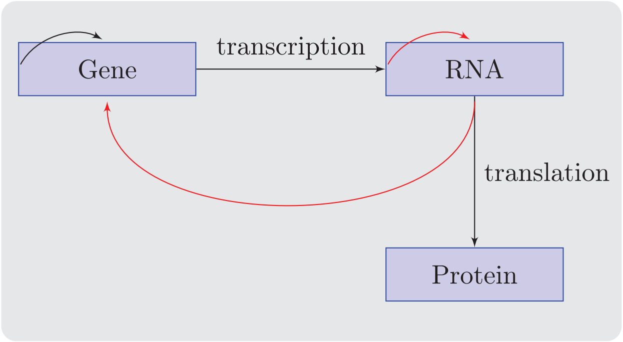


Figure 1.1: Central dogma of molecular biology

of the amount of resulting functional products, and is referred to as *gene expression*. The recent high-throughput genomic technologies are able to measure the gene expression values and have provided large scale data sets, which can be used to obtain insights into how the gene networks are organized and operated. One of the most encountered representations of gene regulatory networks is in terms of a graph, where the genes are depicted by its nodes and the edges represent the interactions between them. The gene regulatory network (GRN) inference problem consists of understanding the underlying system model [7, 20, 27, 38, 54]. Simply stated, given the gene expression data, the activation or repression actions by a set of genes on the other genes need to be identified. There are several issues associated with this problem, including the choice of models that capture the gene interactions sufficiently well, followed by robust and reliable inference algorithms that can be used to derive decisive conclusions about the network. The inferred networks vary in their sophistication depending on the extent and accuracy of the prior knowledge available and the type of models used in the process. It is also important that the gene networks thus inferred should possess the highly desirable quality of reproducibility in order to have a high degree of confidence in them. A sufficiently accurate picture of gene interactions could pave the way for significant breakthroughs in finding cures for various genetic diseases including cancer.

Many statistical methods have been applied extensively to solving gene regulatory network inference problems. We first look at the biological data available for inferencing and then explore the different ways in which the data is utilized.

1.1 Available Biological Data

The post-genomic era is distinguished by the availability of huge amount of biological data sets which are quite heterogenous in nature and difficult to analyze [82]. It is expected that these data sets can aid in obtaining useful knowledge about the underlying interactions in gene-gene networks and estimating the TFAs. In the sequel, some of the main types of data used for inference of genomic networks are discussed.

1.1.1 Gene Expression Data

Of all the available datasets, gene expression data is the most widely used for gene regulatory network inference. Gene expression is the process that results in functional transcripts, e.g., RNA or proteins, while utilizing the information coded on the genes. The level of gene expression is an important indicator of how active a gene is and is measured in the form of gene expression data. Similarity in the gene expression profiles of two genes advocates some level of correlation between them.

1.1.1.1 cDNA-Microarray Data

One way of generating cDNA-microarray data is via the DNA microarray technology, which is by far the most popular method employed for this purpose. The number of data samples is in general much smaller than the number of genes. A main drawback associated with cDNA-microarray data is the noise in the observed gene expressions. Although the gene expression values should be continuous, the inability to measure them accurately suggests the use of discretized values.

1.1.1.2 RNA-Seq Data

The recent advancement of sequencing technologies has provided the ability to acquire more accurate gene expression levels [42]. RNA-Seq is a novel technology for mapping and quantifying transcriptomes, and it is expected to replace all the contemporary methods because of its superiority in terms of time, complexity and accuracy. The gene expression estimation in RNA-Seq begins with the reverse transcription of RNA sample into cDNA samples, which undergo high throughput sequencing, resulting into short sequence reads. These reads are then mapped onto the reference genome using a variety of available alignment tools. The gene expression levels are estimated using the mapped reads, and several algorithms have been proposed in the recent literature to find efficient and more accurate estimates of the gene expression levels. The gene expression data obtained in this manner have been found to be much more reproducible and less noisy as compared to the cDNA microarrays. The expression estimation process using RNA-Seq is depicted in Fig. 1.2.

1.1.2 ChIP-chip Data

ChIP-chip data, which is an abbrevation of Chromatin immunoprecipitation and microarray (chip), investigates the interactions between DNA and proteins. This data provides information about the DNA-binding proteins. Since some of the genes encode for transcription factors (TFs) which in turn regulate some other genes and/or proteins, this information comes in handy for the inference of gene networks [54] and TFA estimation. However, generating the ChIP-chip data for large genome would be technically and financially difficult.

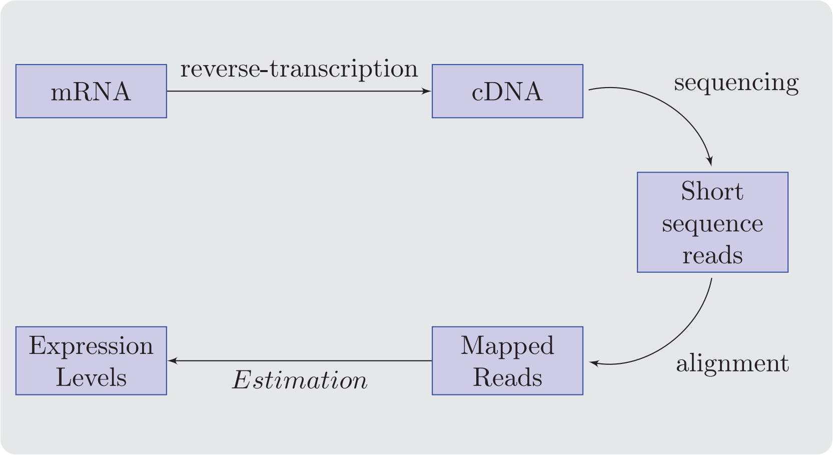


Figure 1.2: Expression estimation in RNA-Seq

1.1.3 Other Data Sets

Apart from the data sets described above, gene deletion and perturbation data are worth mentioning here. Perturbation data set is generated by performing an initial perturbation and then letting the system to react to it [75]. The gene expression values at the following time instants and at steady-state are measured thereby obtaining the response of the genes to the specific perturbation which could be the increase or decrease of the expression level of all or certain genes. Gene deletion dataset, as the name indicates, involves deleting a gene and measuring the resulting expression level of other genes. This data may effectively uncover simple direct relationships [75].

1.2 Modeling and Inferring Gene Regulatory Networks

Gene regulatory networks capture the interactions present among the genes. Accurate and reliable estimation of gene networks is significantly crucial and can reap far-reaching benefits in the field of medicinal biology, e.g., in terms of developing personalized medicines. Several statistical methods have been used for inference of gene regulatory networks of which probabilistic graphical models is the most important one.

1.2.1 Probabilistic Graphical Modeling Techniques

Probabilistic graphical models have emerged as a useful tool for reverse engineering gene regulatory networks. A gene network is represented by a graph $\mathbf{G} = (V, E)$, where V represents the set of vertices (genes), and E denotes the set of edges connecting the vertices. The vertices of the graph are modeled as random variables and the edges signify the interaction between them. The expression value of gene i is denoted by X_i and the total number of genes in the network is denoted by N. Bayesian networks, dynamic Bayesian networks, and Markov networks, are some of the methods representative of this class.

1.2.2 Information Theoretic Methods

Information theoretic methods have provided some of the most robust and reliable algorithms for gene network inference, and form the basis of a standard in this field [37,52,79]. A particular advantage associated with these methods is their ability to work with minimal assumptions about the underlying network. This is in contrast with the probabilistic graphical modeling techniques, e.g., a Markov network provides an undirected network, while Bayesian networks are not able to incorporate cycles or feedback loops. These drawbacks are not present in the case of information theoretic methods.

1.2.3 State-space Representation Models

One of the earliest and widely used methods of modeling gene networks is by employing the state-space representation models [71]. As opposed to other classes, all the methods belonging to this class model the dynamic evolution of the gene network. These models generally consist of two sets of equations, the first set of equations representing the evolution of the hidden state variables denoted by $\mathbf{z}(t)$, and the second set of equations relating the hidden state variables with the observed gene expression data, denoted by $\mathbf{x}(t)$.

The simplest model for state-space equations is the linear Gaussian model given by [71], [69]

$$\begin{aligned} \mathbf{z}(t) &= \mathbf{A}\mathbf{z}(t-1) + \mathbf{v}(t) \\ \mathbf{x}(t) &= \mathbf{C}\mathbf{z}(t) + \mathbf{w}(t)\ , \end{aligned} \tag{1.1}$$

where $\mathbf{A}$ is a matrix representing the regulatory relations between the genes and t stands for the discrete time points. Difference equations are used in place of differential equations because discrete observations are available in the gene expression data. The noise components $\mathbf{v}(t)$ and $\mathbf{w}(t)$ represent the system and the measurement noise, respectively, and are assumed to be Gaussian. The noise models the uncertainty present in the estimated gene expression data. The matrix $\mathbf{C}$ is generally considered to be an identity matrix. Inference in gene networks modeled by the state space representation (1.1) can be performed using standard Kalman filter updates. The simplicity of the state-space model avoids over-fitting of the network, and therefore, it provides reliable results. Use of more sophisticated state-space models to infer gene networks will be studied in this dissertation.

1.3 TFA Estimation Using Network Component Analysis

As opposed to looking at the gene-gene interactions only, it is important to understand the impact of proteins on the gene expression as well, and to deal with an

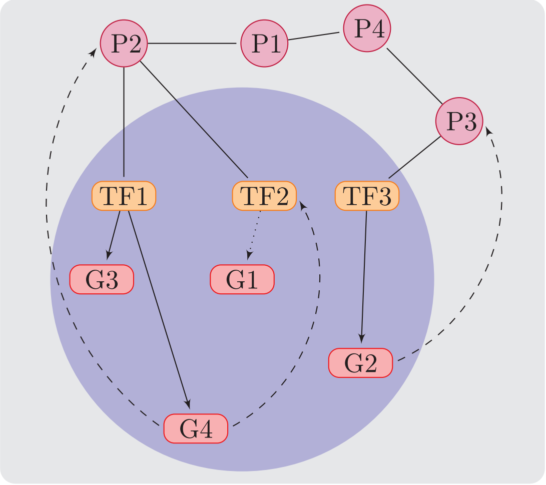


Figure 1.3: An integrated cellular network

integrated cellular network as shown in Fig. 1.3. Transcription factor activity (TFA), which is defined as the concentration of its subpopulation with DNA binding ability, controls the transcriptional regulation [25]. The correlation between TFAs and TF expression level is modified at the post-transcriptional and post-translational stage. It is, therefore, much harder to measure TFA profiles experimentally, and scientists have resorted to computational methods for their estimation [74]. Hence, in addition to looking at the gene-gene interactions, it is imperative to understand how the genes and proteins such as transcription factors are interacting. The relationship between TFAs and mRNA degradation is modeled by a power-law rate expression [28, 63]

$$\frac{dE_i(t)}{dt} = k_{promoter} \prod TFA_j(t)^{CS_{ij}} - k_{degradation} \prod DFA_k(t)^{CS_{ik}}.E_i(t) \ , \qquad (1.2)$$

where $E_i(t)$ is the gene expression level, TFA_j is the j^{th} TF activity, DFA_k is the k^{th} degradation factor activity and CS denotes the control strength of TF j on gene i. Assuming that the time scale allows for a quasi-steady state approximation, (1.2) can be expressed as

$$E_i(t) = \frac{k_{promoter} \prod TFA_j(t)^{CS_{ij}}}{k_{degradation} \times \prod DFA_k(t)^{CS_{ik}}} \ . \tag{1.3}$$

Without loss of generality, denoting both $TFA_j(t)$ and $DFA_j(t)$ as $TFA_j(t)$, (1.3) can be re-written as

$$E_i(t) = \frac{k_{promoter}}{k_{degradation}} \prod TFA_j(t)^{CS_{ij}} \ . \tag{1.4}$$

A log-linear relationship is obtained between the TF and genes by dividing (1.4) by a reference point:

$$\frac{E_i(t)}{E_i(0)} = \prod_{j=1}^{L} \left[\frac{TFA_j(t)}{TFA_j(0)} \right]^{CS_{ij}}$$

Taking logartithm on both sides, the above equation can be expressed in matrix form as:

$$\log[Er] = [CS]\log[TFAr] \ . \tag{1.5}$$

Several statistical techniques including principal component analysis (PCA) [26] and independent component analysis (ICA) [12] have been used to deduce useful information from sets of biological data. However, the successful application of these algorithms hinges on the assumptions of orthogonality and independence between the signals, which do not hold for biological signals in practice [10]. In fact, some prior information is usually available for many systems, and it should be incorporated in the system model, e.g., ChIP-chip data indicates which TFs and genes are known to interact.

1.3.1 Network Component Analysis

Network Component analysis deals with making use of the prior information available about the TF-gene interactions. Using (1.5), the gene regulatory network can be modeled linearly as follows [34]

$$\boldsymbol{Y} = \boldsymbol{A}\boldsymbol{S} + \boldsymbol{\Gamma} , \tag{1.6}$$

where $\boldsymbol{Y}$ is the $N \times K$ gene expression data matrix, $\boldsymbol{A}$ is the $N \times M$ control strength or connectivity matrix, and $\boldsymbol{S}$ is the $M \times K$ matrix denoting the TFAs. The uncertainties in the observation data are assumed to be Gaussian [10, 24], and are represented by the entries of the noise matrix $\boldsymbol{\Gamma}$. Genes and TFs are known to interact in a dynamic and non-linear manner; however, a log-linear relationship provides a good approximation. Since a particular TF regulates only a few other genes, the connectivity matrix $\boldsymbol{A}$ is expected to be sparse. The problem then boils down to estimating $\boldsymbol{S}$ and $\boldsymbol{A}$, where $\boldsymbol{Y}$ is available and some a-priori information about the matrix $\boldsymbol{A}$ is known.

Network component analysis (NCA), proposed by [34], provides a more accurate model for TF-gene regulation and makes use of the related prior information available. It was shown that provided certain conditions are met, the NCA algorithm produces a unique solution of the aforementioned estimation problem in the absence of noise. The *NCA criteria* require that:

1. The matrix $\boldsymbol{A}$ is full column-rank.

2. If a node is removed from the regulatory layer as well as the output elements connected to it, the updated control strength matrix should still be of full column-rank.

3. The TFA matrix $\boldsymbol{S}$ should have a full row-rank. These criteria ensure that the solution obtained is unique up to a scale ambiguity [24, 34].

When the NCA criteria are satisfied, the optimization problem reduces to:

$$\min_{\mathbf{A},\mathbf{S}} \ ||\mathbf{Y} - \mathbf{AS}||_F^2 \qquad \text{s.t. } \ \mathbf{A}(I) = 0 \ , \tag{1.7}$$

where $\|.\|_F$ denotes the Frobenius norm and I is the set of all indices where the entries of matrix $\boldsymbol{A}$ are known to be zero.

1.3.2 Related Work

The significance of the NCA problem can be judged by its successful and effective application in various scenarios e.g., [66] proposed an algorithm that incorporates the motif information in the NCA algorithm and [62] modified the NCA algorithm by trimming the network connectivity to keep the important TF-gene interactions. The algorithm in [15] allows the recovery of source signals when the microarray data consists of fewer data points.

The first solution to the estimation problem (1.7) was proposed in [34], where alternating least squares (ALS) was employed to estimate both the matrices $\boldsymbol{A}$ and $\boldsymbol{S}$. However, since the ALS solution requires solving a high dimensional matrix optimization problem at each iteration, it entails prohibitive computational complexity for large data sets, which often need to be handled in gene networks. Moreover, the solution to ALS can result in multiple local minima and ill-conditioned matrices. To alleviate the latter problems [63] proposed the use of Tikhonov regularization on the TFA matrix $\boldsymbol{S}$. In order to lower the computational burden, FastNCA was proposed which provides a much faster solution by making use of singular value decomposition (SVD) techniques [10], and is several tens of times faster than the ALS algorithm.

The authors in [24] propose a non-iterative version of NCA, herein referred to as NINCA, which offers greater consistency in terms of TFA estimation at the cost of much higher computational complexity than FastNCA.

1.4 Main Contributions of This Research

The main contributions in this research are as follows:

- Non-iterative network component analysis (NINCA), proposed by [24], employs convex optimization methods to estimate the transcription factor control strengths and transcription factor activities. While NINCA provides good estimation accuracy and higher consistency, the costly optimization routine used therein renders a high computational complexity. Section 2 presents a closed form solution to estimate the connectivity matrix which is tens of times faster, and provides similar accuracy and consistency, thus making the closed form NINCA (CFNINCA) algorithm useful for large data sets encountered in practice. The proposed solution is assessed for accuracy and consistency using synthetic and yeast cell cycle data sets by comparing with the existing state-of-the-art algorithms. The robustness of the algorithm to the possible inaccuracies in prior information is also analyzed and it is observed that CFNINCA and NINCA are much more robust to erroneous prior information as compared to FastNCA.

- The algorithms currently available for network component analysis crucially depend on the completeness of this prior information. However, inaccuracies in the measurement process may render incompleteness in the available knowledge about the connectivity matrix. Hence, computationally efficient algorithms are needed to overcome the possible incompleteness in the available data. We present a sparse network component analysis algorithm (sparseNCA) [51] in

Section 3, which incorporates the effect of incompleteness in the estimation of TRNs by imposing an additional sparsity constraint using the ℓ_1 norm, which results in a greater estimation accuracy. In order to improve the computational efficiency, an iterative re-weighted ℓ_2 method is proposed for the NCA problem which not only promotes sparsity but is hundreds of times faster than the ℓ_1 norm based solution. The performance of sparseNCA is rigorously compared to that of FastNCA and NINCA using synthetic data as well as real data. It is shown that sparseNCA outperforms the existing state-of-the-art algorithms both in terms of estimation accuracy and consistency with the added advantage of low computational complexity. The performance of sparseNCA compared to its predecessors is particularly pronounced in case of incomplete prior information about the sparsity of the network. Subnetwork analysis is performed on the *E. coli* data which reiterates the superior consistency of the proposed algorithm.

- Most of the contemporary algorithms for NCA either exhibit the drawback of inconsistency and poor reliability, or suffer from prohibitive computational complexity. In addition, the existing algorithms do not possess the ability to counteract the presence of outliers in the microarray data, which degrades the accuracy of the estimates. Hence, there is a need for algorithms that are not only robust in the presence of outliers, but also lower the computational burden to enable practical applications. In Section 4, we present ROBust Network Component Analysis (ROBNCA) [45], a novel iterative algorithm that explicitly models the possible outliers in the microarray data. An attractive feature of the ROBNCA algorithm is the derivation of a closed form solution for estimating the connectivity matrix, which was not available in prior contributions. The ROBNCA algorithm is compared to FastNCA and NINCA. Since it is well equipped to handle the presence of outliers by estimating them at each step, ROBNCA estimates the TF activity profiles as well as the TF-gene control strength matrix with a much higher degree of accuracy than FastNCA and NINCA, irrespective of varying noise, correlation and/or amount of outliers in case of synthetic data. The ROBNCA algorithm is also tested on *Saccharomyces cerevisiae* data and it is observed to outperform the existing algorithms. A similar analysis is also performed for *Escherichia coli* data which further corroborates the superior performance of the ROBNCA algorithm. The run time of the ROBNCA algorithm is comparable to that of FastNCA, and is hundreds of times faster than NINCA, which makes ROBNCA a suitable candidate for gene network reconstruction.

- Section 5 proposes a novel algorithm for inferring gene regulatory networks which makes use of cubature Kalman filter (CKF) and Kalman filter (KF) techniques in conjunction with compressed sensing methods [50]. The gene network is described using a state-space model. A non-linear model for the evolution of gene expression is considered, while the gene expression data is assumed to follow a linear Gaussian model. The hidden states are estimated using CKF. The system parameters are modeled as a Gauss-Markov process and are estimated using compressed sensing based KF. These parameters provide insight into the regulatory relations among the genes. The Cramer-Rao lower bound of the parameter estimates is calculated for the system model and used as a benchmark to assess the estimation accuracy. The proposed algorithm is evaluated rigorously using synthetic data in different scenarios which include different number of genes and varying number of sample points. In addition, the algorithm is tested on the DREAM4 *in silico* data sets as well as the *in vivo* data sets from IRMA network. The proposed algorithm shows superior performance in terms of accuracy, robustness and scalability.

2. A CLOSED-FORM SOLUTION FOR TRANSCRIPTION FACTOR ACTIVITY ESTIMATION USING NETWORK COMPONENT ANALYSIS*

2.1 Introduction

A convex optimization based non-iterative NCA method: NINCA was proposed in [24] to estimate the signals with higher consistency even in the presence of high correlation. However, this algorithm estimates the connectivity matrix $\boldsymbol{A}$ by resorting to a costly optimization routine, and the resulting high computational complexity may limit its usefulness for large data sets encountered in practice. In order to alleviate the computational load, this section presents a closed-form solution to the optimization problem, herein referred to as Closed-Form NINCA (CFNICA), exhibiting a significantly reduced complexity. Simulations are performed over synthetic as well as real data to test the performance of the proposed CFNICA solution. It is observed that the CFNICA solution for the estimation of connectivity matrix $\boldsymbol{A}$ presents the same superior estimation performance as that offered by NINCA and leads to a significant reduction in computational complexity.

2.2 CFNINCA: NINCA with Closed Form Solutions

CFNINCA is a two step algorithm which first estimates the matrix $\boldsymbol{A}$ and once it is available, the problem of estimating $\boldsymbol{S}$ is reduced to a simple least-squares algorithm. The following subsections explain the estimation of the two matrices.

*Part of this section is reprinted with persmission from "ROBNCA: Robust Network Component Analysis for Recovering Transcription Factor Activities," Amina Noor, Aitzaz Ahmad, Erchin Serpedin, Mohamed Nounou and Hazem Nounou, Bioinformatics, 29(19):2410-2418, Aug 2013. Copyright 2013 by Oxford University Press.

2.2.1 Estimating Connectivity Matrix $\boldsymbol{A}$

In order to estimate $\boldsymbol{A}$, the signal and noise subspaces are separated from $\boldsymbol{Y}$. Rewriting (1.6) in the form of columns as

$$\boldsymbol{y}_k = \boldsymbol{A}\boldsymbol{s}_k + \boldsymbol{\gamma}_k \quad k = 1, 2, ..., K \ , \tag{2.1}$$

where $\boldsymbol{\gamma}_k$ denotes the k^{th} measurement noise vector consisting of $i.i.d$ Gaussian random variables with zero mean and variance σ_γ^2. Let $\boldsymbol{R_s}$ denote the autocorrelation of the TFA vector $\boldsymbol{s}_k$. Then the autocorrelation of the microarray data can be written as

$$\boldsymbol{R_y} = \mathbb{E}\{\boldsymbol{y}_k\boldsymbol{y}_k^T\} = \boldsymbol{A}\boldsymbol{R}_s\boldsymbol{A}^T + \sigma_\gamma^2\boldsymbol{I} \ . \tag{2.2}$$

This autocorrelation matrix $\boldsymbol{R_y}$ is factorized using eigenvalue decomposition to represent it in terms of its eigenvalues and eigenvectors as

$$\boldsymbol{R_y} = \boldsymbol{U}(\boldsymbol{\Lambda} + \sigma_\gamma^2\boldsymbol{I})\boldsymbol{U}^T \ , \tag{2.3}$$

where $\boldsymbol{U} = [\boldsymbol{U_s} \ \ \boldsymbol{U_0}]$ is the unitary matrix consisting of eigenvectors of $\boldsymbol{R_y}$, and $\boldsymbol{\Lambda}$ denotes a diagonal matrix with eigenvalues of $\boldsymbol{R_y}$ as the diagonal entries. The matrices $\boldsymbol{U_s}$ and $\boldsymbol{U_0}$ denote the eigenvectors spanning the signal and noise subspaces, respectively, and are orthogonal to each other by virtue of principle of subspace separation [10, 24]. Using the NCA criteria, it follows that

$$\boldsymbol{U}_0^T\boldsymbol{A}\boldsymbol{R}_s\boldsymbol{A}^T = \boldsymbol{0} \ \Rightarrow \ \boldsymbol{U}_0^T\boldsymbol{A} = \boldsymbol{0} \ , \tag{2.4}$$

which can be expressed column wise as

$$\boldsymbol{U}_0^T \boldsymbol{a}_m = \boldsymbol{0} \ , \quad m = 1, ..., M \ . \tag{2.5}$$

Any $\boldsymbol{A}$ that lies in the signal subspace will satisfy (2.4). However, NCA makes use of the prior information about the connectivity matrix $\boldsymbol{A}$. Suppose that the set of indices where the entries of the connectivity matrix are known to be zero is denoted by I, then it was shown in [24], that the solution to the following constrained subspace problem is unique up to a scale ambiguity

$$\boldsymbol{U}_0^T \boldsymbol{A} = \boldsymbol{0}, \qquad \boldsymbol{A}(I) = \boldsymbol{0} \ . \tag{2.6}$$

Let $\boldsymbol{L}_m$ denote the number of known zeros in a column of $\boldsymbol{a}_m$, then since the rows of $\boldsymbol{Y}$ and $\boldsymbol{A}$ can be interchanged, each column of $\boldsymbol{A}$ can be rearranged as

$$\boldsymbol{a}_m = \begin{bmatrix} \bar{\boldsymbol{a}}_m \\ \\ \boldsymbol{0}_{L_m \times 1} \end{bmatrix} \ . \tag{2.7}$$

It was shown in [24], that under NCA conditions (1) and (2), the subspace constrained solution to $\boldsymbol{U}_0^T \boldsymbol{a}_m = 0$ is unique subject to $\boldsymbol{a}_m$ given in (2.7). Moreover, the solution can be obtained by estimating the columns individually rather than the complete matrix, thereby reducing the complexity. In order to avoid the trivial solution $\boldsymbol{a}_m = \boldsymbol{0}$, a normalization constant is also added in the optimization problem.

This subspace approach requires the ensemble average of the correlation matrix $\boldsymbol{R_y}$ which is difficult to obtain because of limited gene expression data. Therefore, the signal and noise subspaces are determined by factorizing the matrix $\boldsymbol{Y}$ using

SVD as

$$\boldsymbol{Y} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T,$$

where $\boldsymbol{U}$ can be partitioned as previously, into the signal and noise subspaces. Then, the constrained optimization problem is given by [24]

$$\min_{\boldsymbol{a}_m} \ ||\boldsymbol{U}_0^T \boldsymbol{a}_m||_p \quad \text{s.t. } \boldsymbol{a}_m = \begin{bmatrix} \bar{\boldsymbol{a}}_m \\ \boldsymbol{0}_{L_m \times 1} \end{bmatrix}, \ \ \boldsymbol{1}^T.\boldsymbol{a}_m = 1 \ , \tag{2.8}$$

where $p \in (1, 2, \infty)$ denotes a parameter used to choose the norm.

Remark 1. *The optimization problem in* (2.8) *was solved using convex optimization algorithms for $p = 1, 2$ in [24]. However, for real data sets, the vector $\boldsymbol{a}_m$ is usually large and its optimization entails significant computational complexity. Hence, a closed form solution is desired to improve the complexity and efficiency of the subspace based approach.*

In this correspondence, we derive a closed form solution for $p = 2$ using convex optimization techniques. Define an $L_m \times N$ matrix $\boldsymbol{C}_m$ such that

$$\boldsymbol{C}_m = \begin{bmatrix} \boldsymbol{0}_{L_m \times (N - L_m)} & \boldsymbol{I}_{L_m} \end{bmatrix} . \tag{2.9}$$

Using the above definition, the optimization problem (2.8) can be equivalently written as

$$\begin{aligned} \hat{\boldsymbol{a}}_m = \arg\min_{\boldsymbol{a}_m} & \ \ ||\boldsymbol{U}_0^T \boldsymbol{a}_m||_2^2 \\ \text{such that} & \quad \boldsymbol{C}_m \boldsymbol{a}_m = \boldsymbol{0}, \quad \boldsymbol{1}^T \boldsymbol{a}_m = 1 \end{aligned} \tag{2.10}$$

Define now the substitute vector $\bar{\boldsymbol{a}}_m$ via the following equation:

$$\boldsymbol{a}_m = \boldsymbol{D}_m \bar{\boldsymbol{a}}_m \ , \tag{2.11}$$

where the $N \times L_m$ matrix $\boldsymbol{D}_m$ is constructed such that it lies in the null space of the matrix $\boldsymbol{C}_m$, i.e., $\boldsymbol{C}_m \boldsymbol{D}_m = \boldsymbol{0}$. The matrix $\boldsymbol{D}_m$ is, therefore, given by

$$\boldsymbol{D}_m = \begin{bmatrix} \boldsymbol{I}_{(N-L_m)} \\ \boldsymbol{0}_{L_m \times (N-L_m)} \end{bmatrix} . \tag{2.12}$$

Upon substituting $\boldsymbol{a}_m$ from (2.11) in (2.10), we note that the first constraint is always satisfied by virtue of the construction of matrix $\boldsymbol{D}_m$. The resulting optimization problem can be rewritten as

$$\begin{aligned} \hat{\bar{\boldsymbol{a}}}_m = \arg\min_{\bar{\boldsymbol{a}}_m} & \ \ \frac{1}{2} \bar{\boldsymbol{a}}_m^T \boldsymbol{D}_m^T \boldsymbol{Q} \boldsymbol{D}_m \bar{\boldsymbol{a}}_m \\ \text{such that} & \quad \boldsymbol{1}^T \bar{\boldsymbol{a}}_m = 1, \end{aligned} \tag{2.13}$$

where $\boldsymbol{Q} = \boldsymbol{U}_0 \boldsymbol{U}_0^T$. The Lagrangian function can be expressed as

$$\mathcal{L} = \frac{1}{2} \bar{\boldsymbol{a}}_m^T \boldsymbol{D}_m^T \boldsymbol{Q} \boldsymbol{D}_m \bar{\boldsymbol{a}}_m - \mu \left(\boldsymbol{1}^T \bar{\boldsymbol{a}} - 1 \right) . \tag{2.14}$$

The Karush-Kuhn-Tucker (KKT) conditions can be written as

$$\begin{aligned} \boldsymbol{D}_m^T \boldsymbol{Q} \boldsymbol{D}_m \bar{\boldsymbol{a}}_m - \mu \boldsymbol{1} = \boldsymbol{0} \\ \boldsymbol{1}^T \bar{\boldsymbol{a}}_m = 1. \end{aligned} \tag{2.15}$$

It can be shown that the KKT conditions are necessary and sufficient [4]. It follows from the first condition that

$$\bar{\boldsymbol{a}}_m = \mu \left(\boldsymbol{D}_m^T \boldsymbol{Q} \boldsymbol{D}_m\right)^{-1} \mathbf{1} \tag{2.16}$$

where the matrix $\boldsymbol{D}_m^T \boldsymbol{Q} \boldsymbol{D}_m$ is indeed invertible since $\boldsymbol{D}_m$ has full column rank and $\boldsymbol{Q}$ is a product of unitary matrices. Substituting (2.16) into (2.15), the Lagrange multiplier can be expressed as

$$\mu = \frac{1}{\mathbf{1}^T \left(\boldsymbol{D}_m^T \boldsymbol{Q} \boldsymbol{D}_m\right)^{-1} \mathbf{1}}. \tag{2.17}$$

The symmetric invertible matrix $\boldsymbol{Q}$ is partitioned as follows

$$\boldsymbol{Q} = \begin{bmatrix} \boldsymbol{Q}_{11} & \boldsymbol{Q}_{12} \\ \boldsymbol{Q}_{21} & \boldsymbol{Q}_{22} \end{bmatrix} ,$$

where the invertible matrix $\boldsymbol{Q}_{11}$ stands for the upper left-corner $(N - L_m) \times (N - L_m)$ submatrix of $\boldsymbol{Q}$. From the structure of $\boldsymbol{D}_m$, the matrix $\boldsymbol{D}_n^T \boldsymbol{Q} \boldsymbol{D}_m$ can be reduced to

$$\begin{aligned} & \boldsymbol{D}_m^T \boldsymbol{Q} \boldsymbol{D}_m \\ &= \begin{bmatrix} \boldsymbol{I}_{(N-L_m)} & \mathbf{0}_{(N-L_m)\times L_m} \end{bmatrix} \begin{bmatrix} \boldsymbol{Q}_{11} & \boldsymbol{Q}_{12} \\ \boldsymbol{Q}_{21} & \boldsymbol{Q}_{22} \end{bmatrix} \begin{bmatrix} \boldsymbol{I}_{(N-L_m)} \\ \mathbf{0}_{L_m \times (N-L_m)} \end{bmatrix} \\ &= \boldsymbol{Q}_{11} \ . \end{aligned} \tag{2.18}$$

The constrained solution for $\bar{\boldsymbol{a}}_m$ is therefore given by

$$\bar{\boldsymbol{a}}_m = \frac{\boldsymbol{Q}_{11}^{-1} \mathbf{1}}{\mathbf{1}^T \boldsymbol{Q}_{11}^{-1} \mathbf{1}} \ . \tag{2.19}$$

Remark 2. *The closed form solution* (2.19) *only requires the inversion of the* $(N - L_m) \times (N - L_m)$ *submatrix* $\boldsymbol{Q}_{11}$*, which is typically a much smaller matrix, since there are a few non-zero entries in* $\boldsymbol{a}_m$*. The matrix inversion requires* $O\left((N - L_m)\right)^3$ *operations. The numerator in* (2.19) *requires* $O\left((N - L_m)\right)^2$ *operations. The denominator requires* $O\left((N - L_m)\right)^2 + O\left((N - L_m)\right)$ *operations. Hence, the complexity of the closed form solution is approximately* $O\left((N - L_m)\right)^3$ *for large* $(N - L_m)$*.*

2.2.2 Estimating the TFA Matrix $\boldsymbol{S}$

Once an estimate $\hat{\boldsymbol{A}}$ is available, $\boldsymbol{S}$ can now be estimated using a least squares criterion. The optimization problem can be expressed as

$$\boldsymbol{S} = \arg \min_{\boldsymbol{S}} \|\boldsymbol{X} - \hat{\boldsymbol{A}}\boldsymbol{S}\|_F^2 \ . \tag{2.20}$$

By setting the derivative of (4.4) equal to zero and solving for $\boldsymbol{S}$, the estimate is obtained as

$$\boldsymbol{S} = \left(\hat{\boldsymbol{A}}^T \hat{\boldsymbol{A}}\right)^{-1} \hat{\boldsymbol{A}}^T \boldsymbol{X} \ . \tag{2.21}$$

Since closed form solutions are available for the estimates of both $\boldsymbol{A}$ and $\boldsymbol{S}$, CFNINCA exhibits much lower computational complexity than NINCA.

2.3 Simulation Results

In this section, the performance of the proposed algorithm is evaluated in comparison with the existing state-of-the-art algorithms ALS [34], FastNCA [10] and NINCA [24] for synthetic as well as real yeast cell cycle data set.

2.3.1 Synthetic and Hemoglobin Test Data

The algorithm is first investigated for the Hemoglobin test data set used by the original NCA paper [34] and modified in [24], which assumed spectroscopy data ob-

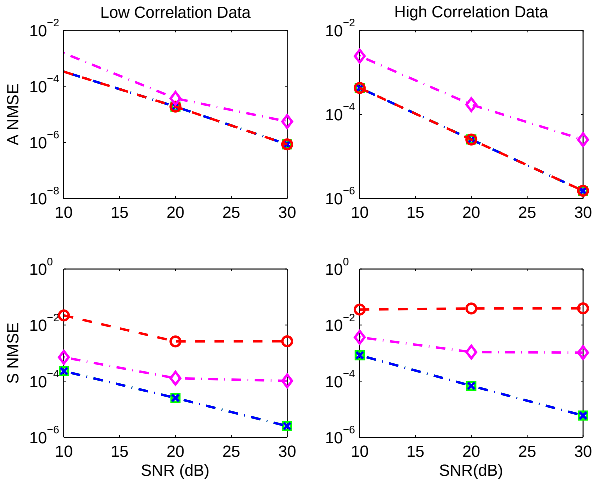


Figure 2.1: Normalized mean square error for the estimation of $\boldsymbol{A}$ and $\boldsymbol{S}$ using CFNINCA (green), NINCA (blue), FastNCA (red), and ALS (magenta).

tained by mixing Hemoglobin solutions. This data set is used because the underlying network structure follows the gene network very closely. Moreover, knowledge of the original source solutions aids in the performance evaluation of the algorithms. The data set consists of $M = 3$ source solutions which result into $N = 7$ mixtures where the spectra are measured for $K = 321$ data points. The presence of a source solution in the mixture solutions indicates the presence of the respective connection in the network connectivity matrix $\boldsymbol{A}$.

This data set is used to evaluate the estimation performance of the algorithms with mean square error (MSE) as the fidelity criterion for $\boldsymbol{A}$ and $\boldsymbol{S}$ matrices. Experiments are performed for low and high correlated data over varying signal-to-noise ratio (SNR), and the Normalized MSEs are depicted in Fig. 2.1. The noise is assumed to be additive white Gaussian (AWGN). For the estimation of $\boldsymbol{A}$, CFNINCA, NINCA and FastNCA perform comparably and provide lower NMSE than the ALS algorithm. CFNINCA and NINCA yield the lowest NMSE for the estimation of $\boldsymbol{S}$

as well, however, the performance of FastNCA and ALS deteriorates significantly. For the estimation of $\boldsymbol{A}$, the MSE decreases with the increase in SNR for all the algorithms, however, FastNCA and ALS exhibit an error floor for estimation of $\boldsymbol{S}$. The better performance of these algorithms in estimating $\boldsymbol{A}$ can be attributed to the availability of prior information.

2.3.2 Subnetwork Analysis

Subnetwork analysis is performed here to assess the consistency of the algorithms, where the data set is divided into four overlapping subsets similar to [74], [10], and [24]. The core idea behind this analysis is to divide the set of transcription factors into a number of smaller subsets, which are not mutually disjoint, where the intersection of these subsets contain the TFs of interest. The subnetworks were constructed to satisfy the gNCA criteria [63] which require that the number of TFAs M should be less than the number of sample points K. These sub-networks are used to estimate the transcription factor activities independent of each other. These TFA estimates are then compared and a smaller disagreement between these estimates is a measure of consistency of the algorithm. This indicates that the results obtained are reliable despite of the presence or absence of certain genes or TFs from the experiment. The disagreement can be quantified as:

$$\text{disagreement}(i) = \frac{1}{K} \sum_{i} \left[\max_{n} s_{n,i}(k) - \min_{n} s_{n,i}(k) \right] \tag{2.22}$$

where s indicates the rows of matrix $\boldsymbol{S}$, i is the TF index and n is the sub-network index.

2.3.3 Results Using S. cerevisiae *Cell Cycle Data*

This section compares the previously mentioned algorithms in the presence of Yeast cell cycle data [33] and [60]. The Yeast cell-cycle data set consists of results from three different synchronization experiments. The first experiment is the synchronization by elutriation which is composed of one cell cycle from 0 to 390 mins. The data consist of 14 points sampled at 30 min intervals. The second experiment performs the synchronization by $\alpha-$factor arrest and contains two cell cycles from 0 to 119 mins. A total of 18 samples are taken every 7 mins. The synchronization in the third set is the result of cdc15 temperature sensitive mutant with samples taken every 20 min from 0 to 300 mins. The data from the three experiments are concatenated to form one large dataset. The Yeast cell cycle study has eleven TFs of interest [10] which are Ace2, Fkh1, Fkh2, Mbp1, Mcm1, Ndd1, Skn7, Stb1, Swi4, Swi5, and Swi6. This section compares the performance of the NCA algorithms for these TFs and the related genes.

Subnetwork analysis is performed here to assess the consistency of the algorithms, where the data set is divided into four overlapping subsets similar to [74]. Each subset consists of 40 TFs, while the 11 TFs under consideration are present in all of them. The number of genes is set to be between 921 to 1247. TFAs are estimated using the four subsets and the difference in their estimation indicates higher degree of inconsistency. This enables us to analyze the robustness of the algorithm to minor modifications in the TFs and genes under consideration [74]. The average of the TFAs estimated using the four subsets is plotted in Fig. 2.2. The rows depict the results of the three synchronization experiments. It is observed that CFNINCA and NINCA result in estimating the same TFA profiles and recovering one, two and three cycles for the three cycles, respectively. FastNCA yields estimates that are either

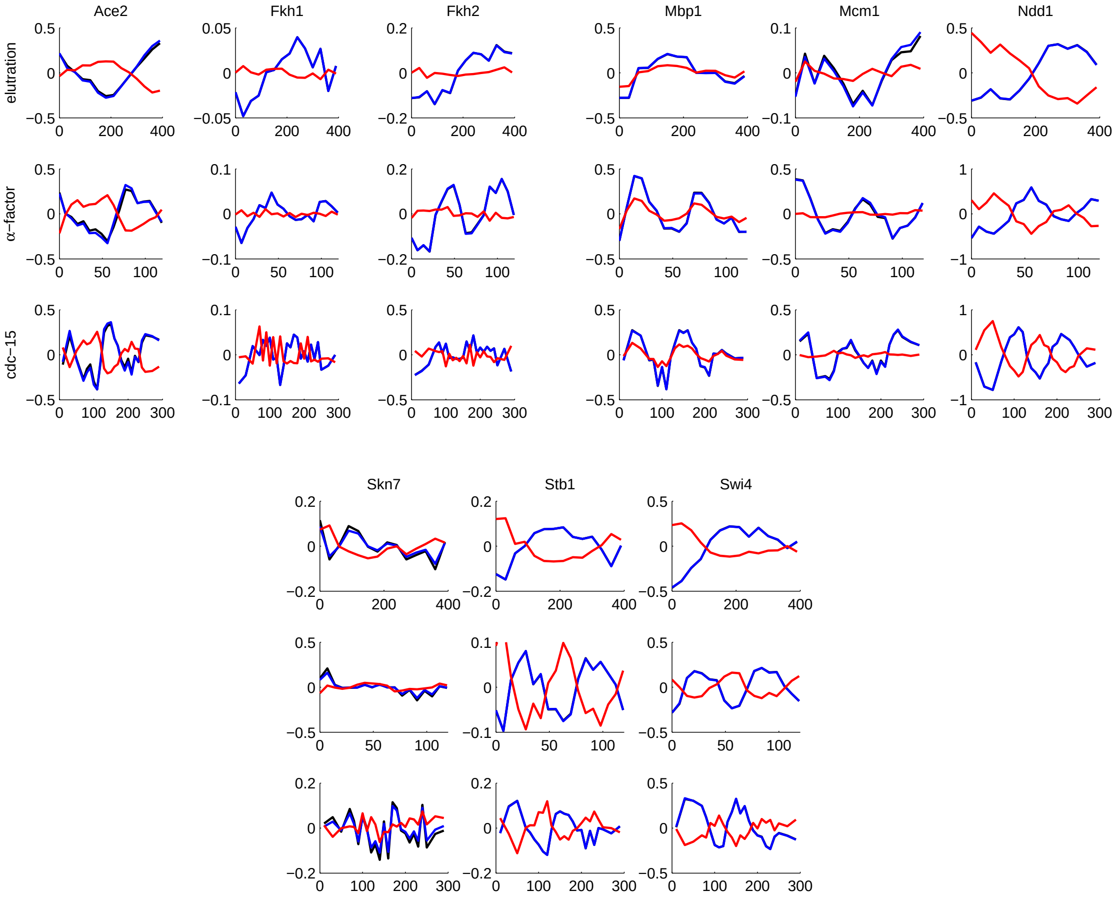


Figure 2.2: TFAs reconstruction: Estimation of 11 TFAs (9 shown) of cell-cycle regulated yeast TFs. Average values of the TFs are shown for the four subnetworks. The results offered by CFNINCA (black), FastNCA (red) and NINCA (blue) are displayed.

opposite to the other algorithms for most TFAs or it does not reveal their periodicity.

In order to further corroborate the results, a consistency comparison study is performed and the disagreement between the subset estimates is shown in Fig. 2.3. It is observed that FastNCA yields much larger disagreement, and therefore, it is less consistent than CFNINCA and NINCA. Therefore, it can be stated that CFNINCA is able to estimate the TFAs with a higher degree of accuracy and consistency. It

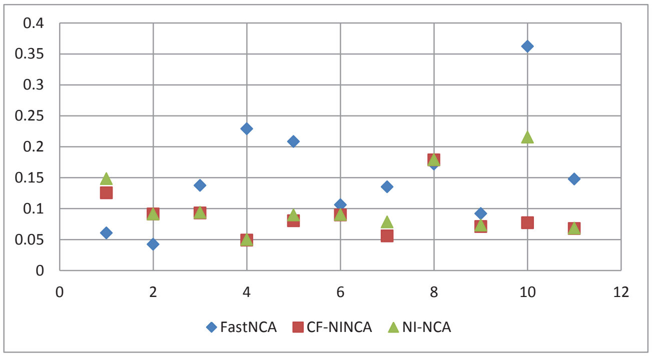


Figure 2.3: Consistency evaluation for *S. cerevisiae* data: Average disagreement from the subsets for TFA estimation.

should also be mentioned that the large size of data set in this experiment prohibits the use of ALS for comparison due to its high computational complexity.

2.3.4 Robustness to Errors in Prior Information

The prior information about connectivity matrix $\boldsymbol{A}$ helps in obtaining a unique solution. However, it is important to study the reliability of the results in case of inaccuracies present in prior knowledge which is a possible scenario [67]. In this analysis, we consider the missed connections only. Suppose that the prior for connectivity matrix erroneously misses some of the true connections and is denoted by $\boldsymbol{A}^{*}$. Then, the m^{th} column of this matrix is given by

$$\boldsymbol{a}_m^{*} = \begin{bmatrix} \bar{\boldsymbol{a}}_m^{*} \\ \mathbf{0}_{L_m^{*}\times 1} \end{bmatrix}, \tag{2.23}$$

where L_m^* denotes the number of zeros in $\boldsymbol{a}_m^*$. The constrained optimization solution can now be stated as

$$\min_{\boldsymbol{a}_m^*} \; ||\boldsymbol{U}_0^T \boldsymbol{a}_m^*||_p \quad \text{s.t. } \boldsymbol{a}_m^* = \begin{bmatrix} \bar{\boldsymbol{a}}_m^* \\ \boldsymbol{0}_{L_m^* \times 1} \end{bmatrix} \;, \;\; \boldsymbol{1}^T.\boldsymbol{a}_m^* = 1, \tag{2.24}$$

Following the same steps as in Section 2.2.1, the solution for this problem is obtained as

$$\bar{\boldsymbol{a}}_m^* = \frac{\boldsymbol{Q}_{11}^{-1*}\boldsymbol{1}}{\boldsymbol{1}^T\boldsymbol{Q}_{11}^{-1*}\boldsymbol{1}} \;, \tag{2.25}$$

where $\boldsymbol{Q}_{11}^*$ is $(N-L_m^*)\times(N-L_m^*)$. Let the error in the m^{th} column be $\boldsymbol{e}_m = \boldsymbol{a}_m - \boldsymbol{a}_m^*$. Then the errors in estimating $\boldsymbol{A}$ and $\boldsymbol{S}$ are calculated as

$$E_A = \sum_{m=1}^{M} ||\boldsymbol{e}_m||_2^2 \;, \tag{2.26}$$

and

$$E_S = ||\boldsymbol{S} - \boldsymbol{S}^*||_F^2 \;, \tag{2.27}$$

respectively. The SNR for this experiment is kept at 30dB. The estimation performance for FastNCA, NINCA and CFNINCA is evaluated using the same Hemoglobin data used in the previous subsection. It is noted in Fig. 2.4, that as the probability of error in the prior increases, the MSEs of all the algorithms increase for $\boldsymbol{A}$. However, NINCA and CFNINCA give much lower MSE than FastNCA for estimation of TFA matrix $\boldsymbol{S}$. NINCA and CFNINCA, therefore, show more robustness to imperfect knowledge of prior. However, CFNINCA offers these advantages at a much lower computational cost.

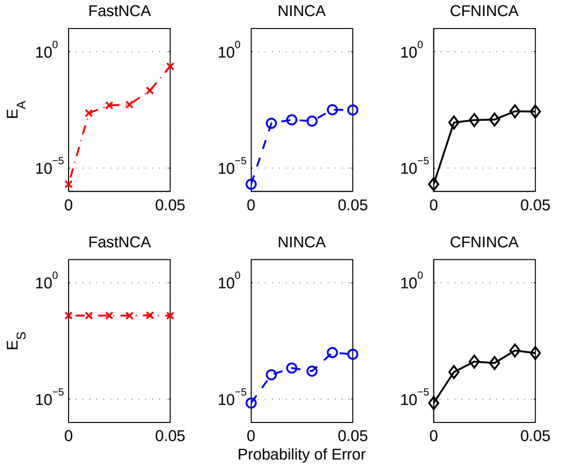


Figure 2.4: Robustness to imperfect prior: Error in the estimation of $\boldsymbol{A}$ and $\boldsymbol{S}$ matrices with missed connections in prior.

Table 2.1: Average computational time in seconds for *S. cerevisiae*.

Subset	1	2	3	4
FastNCA	0.2	0.2	0.24	0.2
CFNINCA	6	3	3	6
NINCA	71	30	125	97
ALS	Exceeds memory limit			

2.3.5 Run Time Comparison

Gene regulatory networks require working with large data sets and therefore a lower computational time for the algorithms is a very appealing feature. We compare the average run time for the algorithms discussed previously for the four subsets of the real data set and the results are given in Table 2.1. These simulations were carried out using Matlab 7.10.0 on a Windows 7 system with a 1.90 GHz Intel Core i7 processor. It is observed that CFNINCA is tens of times faster than NINCA. FastNCA exhibits a clear advantage in terms of lower complexity. However, as noted in the previous simulations, FastNCA suffers from poor estimation accuracy and consistency. The complexity of ALS is known to be prohibitive and is added here only for comparison. Hence, the CFNINCA algorithm avoids the drawback of high computational complexity of the NINCA algorithm by providing a closed form solution to estimate $\boldsymbol{A}$, while maintaining the same estimation accuracy and consistency. This makes CFNINCA well suited for TFA estimation using the large data sets encountered in practice.

2.4 Summary

This section presented a closed form solution to a non-iterative network component analysis algorithm which uses convex optimization techniques to estimate the control strength matrix [24]. The NINCA algorithm exhibits superior consistency in terms of TFA estimation but suffers from high computational complexity. The proposed closed form CFNINCA solution considerably speeds up the algorithm while offering comparable estimation accuracy and consistency to NINCA. The performance of CFNINCA is compared to NINCA, FastNCA, and ALS over synthetic data and yeast cell cycle data. The conducted simulations confirm CFNINCA's advantages in terms of lower run time, robustness to imperfect prior and comparable or better estimation accuracy with respect to the existing state-of-the-art algorithms.

3. SPARSE NETWORK COMPONENT ANALYSIS FOR RECOVERING TRANSCRIPTION FACTOR ACTIVITIES WITH INCOMPLETE PRIOR INFORMATION

3.1 Introduction

A crucial underlying assumption of all the NCA algorithms is the availability of complete prior knowledge of the connectivity matrix. Due to potential errors introduced during the measurement process, Chip-chip data are also highly dependent on the environmental factors [5], and therefore, may not provide the complete information about the TF-gene interactions [84]. The algorithms proposed thus far assume that the zero indices, signifying the absence of a connection, are completely known and the rest of the indices are considered non-zero. However, it is possible that due to the incomplete information, the number of zeros is larger than those provided by the Chip-chip data, as it is known that the TRNs are highly sparse [41, 48]. Therefore, it is imperative to incorporate the incompleteness of the prior information about the connectivity matrix in deriving the NCA solution. It is this very avenue that is investigated in this section.

3.2 Main Contributions

The main contributions in this section can be summarized as follows:

1. We propose sparse network component analysis (sparseNCA), which incorporates the effect of incomplete prior information about the connectivity matrix by introducing additional sparsity constraints in the underlying NCA problem.

2. Sparsity is enforced by making use of an ℓ_1 norm for the estimation of $\boldsymbol{A}$ matrix. In order to derive a closed form expression, an iterative re-weighted

ℓ_2 minimization algorithm is used to estimate the sparse connectivity matrix which is hundreds of times faster than the ℓ_1 norm based solution.

3. SparseNCA algorithm is compared with the state-of-the-art algorithms, and it is shown to outperform NINCA [24] and FastNCA [10] for Hemoglobin synthetic and test data in terms of accuracy and consistency for varying noise, concentration of outliers, correlation of data and amount of prior information. Simulations are also performed with *Escherichia Coli* data and it is observed that sparseNCA recovers the TFAs with a high degree of accuracy and reliability.

3.3 SparseNCA

SparseNCA is a two step algorithm which first estimates the connectivity matrix $\boldsymbol{A}$ by making use of a subspace based method [24] while incorporating the effect of incomplete prior information by imposing a sparsity constraint. A closed form solution is derived which considerably reduces the computational complexity. Once the matrix $\boldsymbol{A}$ is available, the TFA matrix $\boldsymbol{S}$ is estimated by minimizing least squares.

3.3.1 Estimating Connectivity Matrix $\boldsymbol{A}$

It is clear from the above formulation that any solution to the aforementioned constrained optimization problem will fundamentally depend on the prior information about the connectivity matrix. We now consider the scenario where this prior information about the zeros may be incomplete due to errors in the measurement process, and there may be additional zeros in the connectivity matrix not known to us [84]. Towards this end, it is imperative to impose an additional constraint on the vector $\boldsymbol{a}_m$ which exploits sparsity. This is achieved by solving the following combinatorial optimization problem:

$$\min_{\boldsymbol{a}_m} \ ||\boldsymbol{U}_0^T \boldsymbol{a}_m||_2^2 + \lambda ||\bar{\boldsymbol{a}}_m||_0 \quad \text{s.t. } \boldsymbol{a}_m = \begin{bmatrix} \bar{\boldsymbol{a}}_m \\ \boldsymbol{0}_{L_m \times 1} \end{bmatrix} \ , \ \boldsymbol{1}^T.\boldsymbol{a}_m = 1 \ .$$

where $||.||_0$ denotes the number of non-zero elements in $\bar{\boldsymbol{a}_m}$ and $\lambda > 0$ is the regularization parameter. However, it is commonly known that this problem is non-convex and requires a combinatorial search which renders it computationally prohibitive [64]. Therefore, the following alternative problem is commonly solved:

$$\min_{\boldsymbol{a}_m} \ ||\boldsymbol{U}_0^T \boldsymbol{a}_m||_2^2 + \lambda ||\bar{\boldsymbol{a}}_m||_1 \quad \text{s.t. } \boldsymbol{a}_m = \begin{bmatrix} \bar{\boldsymbol{a}}_m \\ \boldsymbol{0}_{L_m \times 1} \end{bmatrix} \ , \ \boldsymbol{1}^T.\boldsymbol{a}_m = 1 \ . \tag{3.1}$$

where $||\bar{\boldsymbol{a}}_m||_1 = \sum_{i=1}^{N-L_m} |\bar{a}_{mi}|$ denotes the ℓ_1 norm of $\bar{\boldsymbol{a}}_m$.

Remark 3. *While the introduction of an ℓ_1 norm promotes sparsity, a closed form solution of* (3.1) *does not exist, and optimization algorithms, e.g., interior point methods must be used. Since gene regulatory networks usually deal with high dimensional data and this computation is typically costly, a simpler algorithm is required.*

3.3.2 Iterative Re-weighted ℓ_2 Minimization for SparseNCA

Instead of constraining the vector using ℓ_1 norm, several alternatives have been proposed which provide a more efficient solution than the interior point method needed to solve (3.1) [70]. We employ an iterative re-weighted least squares method [11, 70] which iteratively solves the following optimization problem at the $(j+1)^{th}$

iteration:

$$\begin{aligned} \boldsymbol{a}_m(j+1) \rightarrow \ \min_{\boldsymbol{a}_m} \ & ||\boldsymbol{U}_0^T \boldsymbol{a}_m||_2^2 + \lambda \sum_i w_i(j) \bar{a}_{mi}^2 \\ \text{s.t.} \ \ \boldsymbol{a}_m = & \begin{bmatrix} \bar{\boldsymbol{a}}_m \\ \boldsymbol{0}_{L_m \times 1} \end{bmatrix} \ , \ \ \boldsymbol{1}^T.\boldsymbol{a}_m = 1 \ , \end{aligned} \tag{3.2}$$

where $w_i, i = 1, ..., N - L_m$ are positive weights. The weights in (3.2) are updated as [11]

$$w_i(j+1) = \left[\left(\bar{a}_{mi}^2(j+1) + \epsilon(j+1) \right) \right]^{-1} . \tag{3.3}$$

where $\epsilon(j+1) \in (0,1)$ is a regularizing parameter. Initially a large value is selected for $\epsilon(j+1)$ which is then decreased upon every iteration. It was observed in [11] that the algorithm converges to a unique solution as $\epsilon(j+1) \rightarrow 0$.

In the sequel, we derive a closed form solution to (3.2) by employing convex optimization methods. We construct an $L_m \times N$ matrix $\boldsymbol{G}_m$ such that

$$\boldsymbol{G}_m = \begin{bmatrix} \boldsymbol{0}_{L_m \times (N-L_m)} & \boldsymbol{I}_{L_m} \end{bmatrix} \ . \tag{3.4}$$

Then the optimization problem (3.2) can be equivalently expressed as

$$\begin{aligned} \boldsymbol{a}_m(j+1) = \arg\min_{\boldsymbol{a}_m} \ & ||\boldsymbol{U}_0^T \boldsymbol{a}_m||_2^2 + \lambda \sum_i w_i(j) \bar{\boldsymbol{a}}_{mi}^2 \\ \text{such that} \quad & \boldsymbol{G}_m \boldsymbol{a}_m = \boldsymbol{0}, \quad \boldsymbol{1}^T \boldsymbol{a}_m = 1 \ . \end{aligned} \tag{3.5}$$

We further construct an $N \times L_m$ matrix $\boldsymbol{H}_m$ which lies in the null space of $\boldsymbol{G}_m$, given by

$$\boldsymbol{H}_m = \begin{bmatrix} \boldsymbol{I}_{(N-L_m)} \\ \boldsymbol{0}_{L_m \times (N-L_m)} \end{bmatrix} . \tag{3.6}$$

Defining the substitution

$$\boldsymbol{a}_m = \boldsymbol{H}_m \bar{\boldsymbol{a}}_m \ , \tag{3.7}$$

and $\boldsymbol{Q} = \boldsymbol{U}_0 \boldsymbol{U}_0^T$, then the problem (3.5) can be expressed as

$$\begin{aligned} \hat{\bar{\boldsymbol{a}}}_m(j+1) = \arg\min_{\bar{\boldsymbol{a}}_m} \ & \frac{1}{2}\bar{\boldsymbol{a}}_m^T \boldsymbol{H}_m^T \boldsymbol{Q} \boldsymbol{H}_m \bar{\boldsymbol{a}}_m + \lambda \frac{1}{2} \bar{\boldsymbol{a}}_m^T \boldsymbol{W}_m(j) \bar{\boldsymbol{a}}_m \\ \text{such that} \quad & \boldsymbol{1}^T \bar{\boldsymbol{a}}_m = 1 \ , \end{aligned} \tag{3.8}$$

where $\boldsymbol{W}_m(j)$ is a diagonal matrix for jth iteration with the vector $w \triangleq [w_1, \ldots, w_{N-L_m}]$ on the diagonal. The Lagrange dual function for (3.8) can be written as

$$\mathcal{L}(\bar{\boldsymbol{a}}_m, \mu) = \frac{1}{2}\bar{\boldsymbol{a}}_m^T \boldsymbol{H}_m^T \boldsymbol{Q} \boldsymbol{H}_m \bar{\boldsymbol{a}}_m + \lambda \frac{1}{2} \bar{\boldsymbol{a}}_m^T \boldsymbol{W}_m(j) \bar{\boldsymbol{a}}_m + \mu \left(\boldsymbol{1}^T \bar{\boldsymbol{a}} - 1 \right) \ ,$$

where $\mu \geq 0$ is the Lagrange multiplier. The Karush-Kuhn Tucker (KKT) conditions can be expressed as [4]

$$\boldsymbol{H}_m^T \boldsymbol{Q} \boldsymbol{H}_m \bar{\boldsymbol{a}}_m + \lambda \boldsymbol{W}_m(j) \bar{\boldsymbol{a}}_m + \mu \boldsymbol{1} = \boldsymbol{0} \tag{3.9}$$

$$\boldsymbol{1}^T \bar{\boldsymbol{a}}_m = 1. \tag{3.10}$$

The KKT conditions can be easily shown to be necessary and sufficient for the optimization problem (3.8) due to the presence of equality constraints and a convex objective function. Solving (3.9), it follows that

$$\bar{\boldsymbol{a}}_m = -\mu \left(\boldsymbol{H}_m^T \boldsymbol{Q} \boldsymbol{H}_m + \lambda \boldsymbol{W}_m(j) \right)^{-1} \boldsymbol{1} \tag{3.11}$$

Plugging (3.11) in (3.10), the Lagrange multiplier μ can be obtained as

$$\mu = -\frac{1}{\mathbf{1}^T \left(\boldsymbol{H}_m^T \boldsymbol{Q} \boldsymbol{H}_m\right)^{-1} \mathbf{1}} \ . \tag{3.12}$$

Substituting (3.12) back in (3.11), the solution for $\bar{\boldsymbol{a}}_m$ can now be expressed as

$$\bar{\boldsymbol{a}}_m(j+1) = \frac{(\boldsymbol{H}_m^T \boldsymbol{Q} \boldsymbol{H}_m + \lambda \boldsymbol{W}_m(j))^{-1}\mathbf{1}}{\mathbf{1}^T(\boldsymbol{H}_m^T \boldsymbol{Q} \boldsymbol{H}_m + \lambda \boldsymbol{W}_m(j))^{-1}\mathbf{1}} \ . \tag{3.13}$$

The special structure of the matrix $\boldsymbol{H}_m$ can be used to further simplify the computation of $\boldsymbol{H}_m^T \boldsymbol{Q} \boldsymbol{H}_m$ as follows:

$$\begin{aligned}
&\boldsymbol{H}_m^T \boldsymbol{Q} \boldsymbol{H}_m \\
&= \begin{bmatrix} \boldsymbol{I}_{(N-L_m)} & \mathbf{0}_{(N-L_m)\times L_m} \end{bmatrix} \begin{bmatrix} \boldsymbol{Q}_{11} & \boldsymbol{Q}_{12} \\ \boldsymbol{Q}_{21} & \boldsymbol{Q}_{22} \end{bmatrix} \begin{bmatrix} \boldsymbol{I}_{(N-L_m)} \\ \mathbf{0}_{L_m\times(N-L_m)} \end{bmatrix} \\
&= \boldsymbol{Q}_{11} \ .
\end{aligned} \tag{3.14}$$

The constrained solution for $\bar{\boldsymbol{a}}_m$ at the $(j+1)^{th}$ iteration is therefore given by

$$\bar{\boldsymbol{a}}_m(j+1) = \frac{(\boldsymbol{Q}_{11} + \lambda \boldsymbol{W}_m(j))^{-1}\mathbf{1}}{\mathbf{1}^T(\boldsymbol{Q}_{11} + \lambda \boldsymbol{W}_m(j))^{-1}\mathbf{1}} \ . \tag{3.15}$$

The iterative process is continued until the difference between the estimated value and the estimate at the previous iteration falls below a threshold. The connectivity matrix is hence estimated column wise and is used for estimating the TFA matrix $\boldsymbol{S}$.

Remark 4. *The closed form solution* (3.15) *requires the addition of* $(N - L_m) \times (N - L_m)$ *matrices* $\boldsymbol{Q}_{11}$ *and* $\lambda \boldsymbol{W}$ *and inversion of the* $(N - L_m) \times (N - L_m)$ *matrix*

$(\boldsymbol{Q}_{11} + \lambda \boldsymbol{W})$, which is typically a much smaller matrix, since there are a few non-zero entries in $\boldsymbol{a}_m$. The matrix addition and inversion requires $O\left(N - L_m\right)^2$ nd $O\left(N - L_m\right)^3$ operations, respectively. The numerator in (5.23) *requires $O\left(N - L_m\right)^2$ operations. The denominator requires $O\left(N - L_m\right)^2 + O\left(N - L_m\right)$ operations. Hence, the complexity of the closed form solution is approximately $O\left(N - L_m\right)^3$ for large $(N - L_m)$.*

3.3.3 Estimating the TFA Matrix $\boldsymbol{S}$

With the estimate of $\boldsymbol{A}$ in hand, the TFA matrix $\boldsymbol{S}$ is estimated by minimizing the least squares and the optimization problem is given by

$$\boldsymbol{S} = \arg \min_{\boldsymbol{S}} \|\boldsymbol{Y} - \boldsymbol{A}\boldsymbol{S}\|_F^2 \ . \tag{3.16}$$

The solution is simply obtained by taking the derivative of (3.16) w.r.t $\boldsymbol{S}$ and equating it to zero yielding

$$\boldsymbol{S} = \left(\boldsymbol{A}^T \boldsymbol{A}\right)^{-1} \boldsymbol{A}^T \boldsymbol{Y} \ , \tag{3.17}$$

where the matrix $\boldsymbol{A}^T\boldsymbol{A}$ is indeed invertible by virtue of the linear independence of columns of $\boldsymbol{A}$ (NCA criterion 1).

The sparseNCA algorithm is summarized in Algorithm 1.

3.4 Results and Discussion

This section evaluates the performance of the sparseNCA algorithm in comparison with FastNCA [10] and NINCA [24]. The sparseNCA algorithm has been implemented in MATLAB. The sparseNCA software is available at `http://people.tamu.edu/~amina/sparsenca`. The source codes for FastNCA and NINCA algorithms are downloaded from `http://www.seas.ucla.edu/~liaoj/download.htm` and `http://www.ece.ucdavis.edu/~jacklin/NCA`.

Algorithm 1 sparseNCA

```
Input: $\boldsymbol{Y}$, Initialize: $\boldsymbol{A}(0) = \boldsymbol{I}$.
for $n = 1, 2, ..., N$ do
    for $j = 1, 2, ...,$ do
        Update $\bar{\boldsymbol{a}}_n(j)$ using (5.23).
        Update weight vector using (3.3).
    end for
end for
Form updated matrix $\boldsymbol{A} = [\boldsymbol{a}_1^T \quad \boldsymbol{a}_2^T \ldots \ \boldsymbol{a}_N^T]^T$.
Update $\boldsymbol{S} = \left(\boldsymbol{A}^T\boldsymbol{A}\right)^{-1} \boldsymbol{A}^T\boldsymbol{Y}$.
return
```

3.4.1 Synthetic and Hemoglobin Test Data

The performance of sparseNCA algorithm is first tested using the synthetic data downloaded from `http://www.ece.ucdavis.edu/~jacklin/NCA` which was also described in the previous section. This data was obtained by mixing $M = 3$ pure components of Hemoglobin solutions to form $N = 7$ different mixtures. The connectivity matrix in this case denotes the presence or absence of a particular solution in the mixtures. The absorption spectra consist of K = 321 points which are measured for wavelengths in the range of 380-700 nm. This dataset has been widely used to assess the performance of NCA algorithms including the original work on NCA [34], FastNCA [10], NINCA [24] and ROBNCA [45]. The system model for the spectroscopy data follows the gene regulatory network model very closely and is captured by the Beer-Lambert Law: $\boldsymbol{Abs} = \boldsymbol{C\epsilon}$ where $\boldsymbol{Abs}$ gives the absorbance spectra, $\boldsymbol{C}$ is the connectivity matrix and $\boldsymbol{\epsilon}$ denotes the pure spectra. The prior knowledge of the pure Hemoglobin solutions aids us in evaluating the accuracy of the algorithms.

The robustness of the algorithms is assessed in case of incomplete prior knowledge about the zeros in the connectivity matrix. In addition, the performance is evaluated for varying degrees of correlation in the signals and amounts of outliers present in the data. The outliers are artificially added to the Hemoglobin using the Bernoulli model while assuming a small probability of occurrence as they are expected to be sparse. The noise is assumed to be Gaussian for all the simulations and the normalized mean square error (NMSE) is used as the fidelity criterion for estimating the connectivity matrix, and it is expressed as $NMSE = \frac{||\boldsymbol{A}-\hat{\boldsymbol{A}}||_F^2}{||\boldsymbol{A}||_F^2}$. The results are averaged over 100 iterations. These scenarios provide a very thorough analysis of the performance of the algorithms.

3.4.1.1 Impact of Incomplete Prior Information

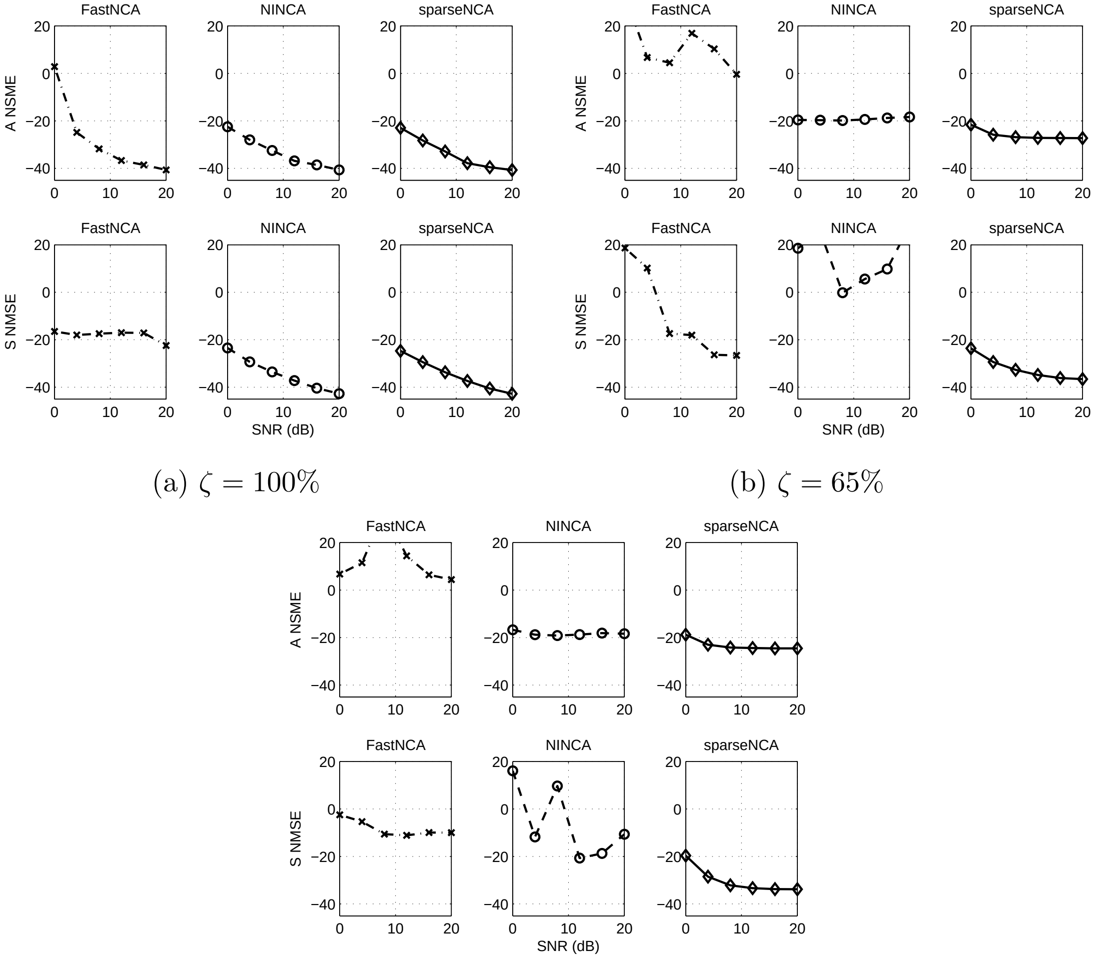
First, a comparison is performed in case of incomplete prior information about the connectivity matrix $\boldsymbol{A}$. While keeping the same matrix structure, simulations are performed by assuming $\zeta = \{100, 65, 35\}$, where ζ denotes the percentage information about the zeros. The results are depicted in Figure 3.1 and 3.2 for different values of ζ. The figures from left to right are plotted for decreasing prior information about the zeros and signal to noise ratio (SNR) is varied from 0 to 20 dBs. For all the simulations, the regularizer parameter ϵ is initialized to 1 and then decreased by a factor of 10 at each iteration following the heuristic approach used in [11]. It is noted from Figures 3.1a and 3.2a that when the prior information is complete, NINCA and sparseNCA give similar performance. However, as the prior information is decreased to 65% and 35%, the performance of NINCA is adversely impacted. On the other hand, sparseNCA results in a much lower NMSE than NINCA due to its incorporation of the incomplete information about the connectivity matrix. FastNCA, however, gives a higher NMSE for all these scenarios. Moreover, the absence of complete prior information makes the estimation by NINCA and FastNCA quite unreliable. It is also observed that the poor estimation of $\boldsymbol{A}$ for both NINCA and FastNCA severely degrades the estimation of $\boldsymbol{S}$, while sparseNCA consistently outperforms both FastNCA and NINCA in estimating the TFA matrix $\boldsymbol{S}$. Hence, sparseNCA has clear advantages over its predecessors when the prior information about the connectivity matrix is incomplete.

3.4.1.2 Impact of Correlation

The simulations are also carried out on the same modified Hemoglobin dataset with low and high correlation. The estimation using low correlation is shown in Figure 3.1 and 3.2, while the results in case of high correlation data are relegated to Figure 1 in Supplementary material. The SNR is varied from 0dB to 20dB for this part of the simulation as well. FastNCA and NINCA are highly inconsistent for the estimation of $\boldsymbol{A}$ and $\boldsymbol{S}$ matrices, respectively, and increasing the SNR does not increase the estimation accuracy. On the other hand, the NMSE for SparseNCA decreases with the increase in SNR. Although, NINCA performs somewhat better than FastNCA, however, the two algorithms perform very poorly when the prior information is incomplete for both low and high correlation datasets compared to the proposed sparseNCA algorithm. A similar trend in the performance of the algorithms is also observed in the case of highly correlated datasets.

3.4.1.3 Impact of Outliers

It is important to test the algorithms in the presence of outliers because real data consist of outlying measurements as well [45]. The results for 0.1% outliers are depicted in Figures 3.1a, 3.1b, and 3.1c. Increasing the outliers to 3% show the sensitivity of FastNCA and NINCA to these inaccuracies as shown in Figures 3.2a, 3.2b, and 3.2c. In general, the algorithms estimate the $\boldsymbol{A}$ matrix better than $\boldsymbol{S}$. The reason for this is that some of the prior information for $\boldsymbol{A}$ is available. As the amount of outliers increases, estimation accuracy for sparseNCA decreases for both $\boldsymbol{A}$ and

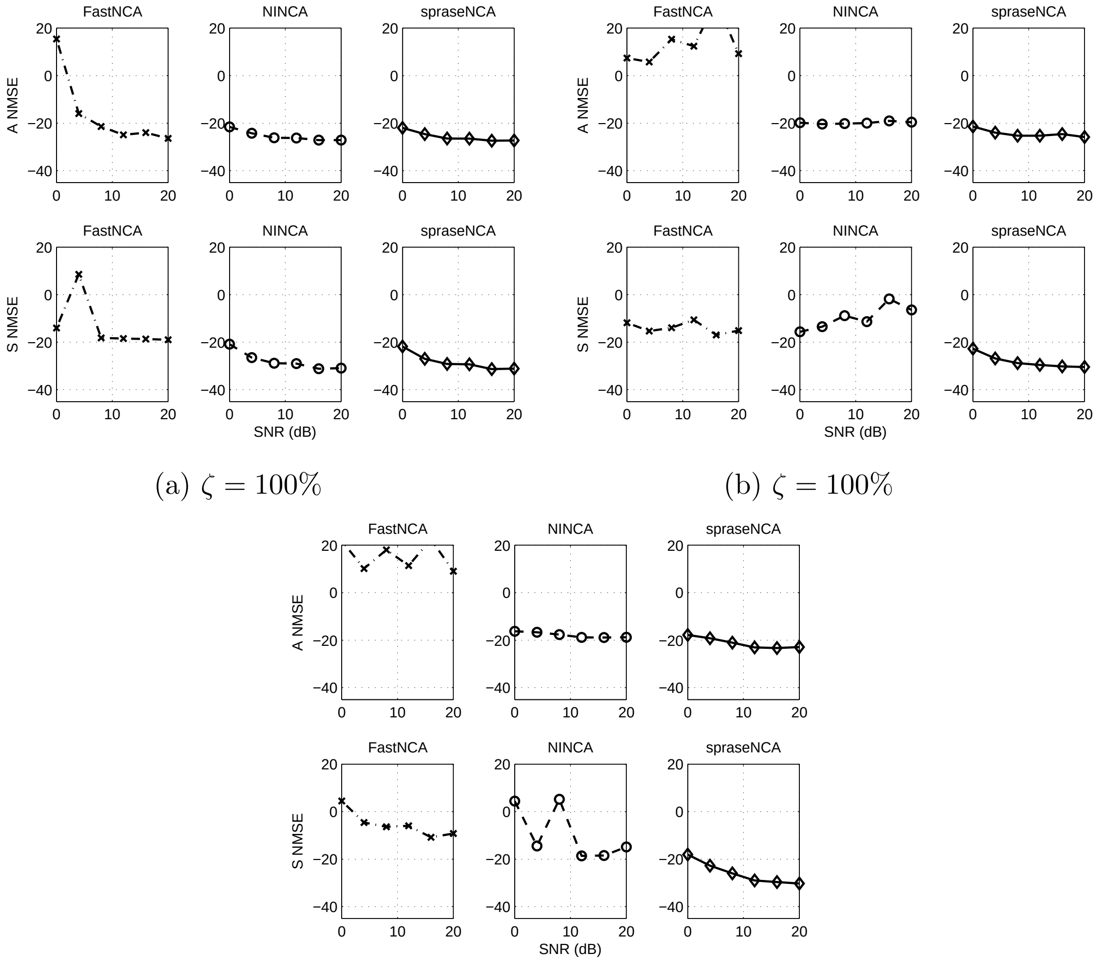


(a) $\zeta = 100\%$

(b) $\zeta = 65\%$

(c) $\zeta = 35\%$

Figure 3.1: Impact of incomplete prior and outliers: Normalized mean square error (NMSE) for FastNCA, NINCA and sparseNCA for different datasets with level of outliers: 0.001 against varying signal to noise ratio (SNR) dB for low correlation data.



(a) $\zeta = 100\%$

(b) $\zeta = 100\%$

(c) $\zeta = 100\%$

Figure 3.2: Impact of incomplete prior and outliers: Normalized mean square error (NMSE) for FastNCA, NINCA and sparseNCA for different datasets with level of outliers: 0.03 against varying signal to noise ratio (SNR) dB for low correlation data.

$\boldsymbol{S}$ matrices, however, the NMSE decreases with the increase in SNR. The other two algorithms, however, do not show consistent behavior and increase in SNR does not help in improving their estimation performance.

3.4.1.4 Comparison between ℓ_1 Norm Solution and sparseNCA Algorithm

The results for estimation of connectivity matrix $\boldsymbol{A}$ using ℓ_1 norm and sparseNCA are shown in Figure 3.3. The simulations are performed for low correlated data by varying the SNR from 0dB to 25dB. The amount of prior information is set to $\zeta = 100, 65, 35$. It is noted that sparseNCA performs as good as or better than ℓ_1 norm solution when the available prior information is 100% and 65%. When the available prior information drops to 35%, there is a small gap between the two algorithms. Therefore, sparseNCA is able to recover the sparsity quite accurately at a much lower computational complexity as will be shown in the next subsection.

By performing these simulations on different degrees of correlation and amount of outliers in the data, it is noted that sparseNCA yields good estimation accuracy if some prior knowledge about the connectivity matrix is missing. Since the transcriptional networks are expected to be sparse, a large number of entries would be zero. Due to inaccuracies in the measurement process, knowledge of these entries may be absent. It can therefore be stated from these simulation results, that sparseNCA yields superior performance than FastNCA and NINCA in case of varying correlation, different degrees of outliers and amounts of available prior information.

3.4.2 Results for E. coli *Data*

SparseNCA algorithm is now used to estimate TFAs for *E. coli* data [28] and the estimates are compared with those obtained from FastNCA and NINCA. In addition, *subnetwork analysis* is performed to ascertain the reliability of the estimates. The dataset for this experiment is downloaded from `http://www.seas.ucla.edu/`

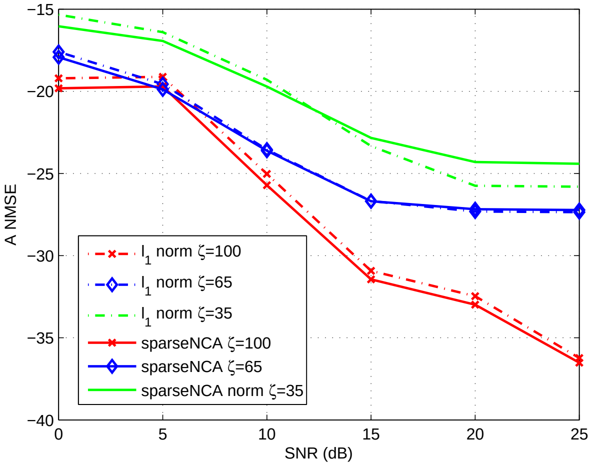


Figure 3.3: Comparison between the ℓ_1 norm solution and the sparseNCA algorithm: Estimation of matrix $\boldsymbol{A}$ for a varying amount of prior information.

`~liaoj/download.htm`.

The gene expression data is measured during the transition of sole carbon source from glucose to acetate and a total of 296 genes were found to be affected during this process. The connectivity matrix $\boldsymbol{A}$ was tested to satisfy the NCA criteria and subsequently, a set of 100 genes and 16 TFs were selected. The TFs consist of ArcA, CRP, CysB, FadR, Cra, GatR, IcIR, LeuO, Lrp, NarL, PhoB, PurR, RPoE, RpoS, TrpR and TyrR. The gene expression data consists of 25 time points which contain repeated measurements as well. The consistency in the TFA estimation can be measured using subnetwork analysis which has been employed previously as well [10, 45, 74]. Subnetwork analysis is based on dividing the dataset into smaller networks, each containing the TFs of interest. TFAs are then estimated using these subnetworks and a smaller disagreement among the estimates indicates a higher consistency of the algorithm. For the *E. coli* data, the data is divided into four subnetworks containing 81 to 88 genes and 20 TFs.

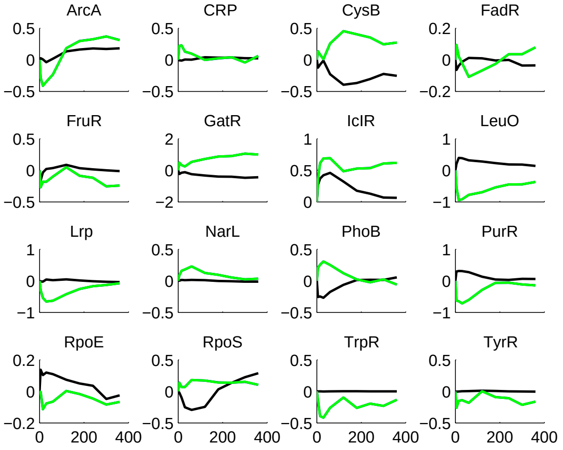


Figure 3.4: TFA estimation for *E. coli* data using FastNCA (black), NINCA (blue), and sparseNCA (green) for the sixteen TFAs of interest. The estimation by sparseNCA and NINCA are in agreement with each other.

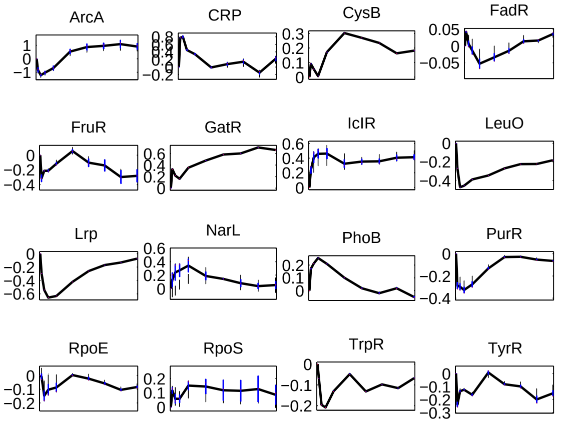


Figure 3.5: Consistency evaluation: Standard deviation in the TFA estimation using sparseNCA. The average values of the four subnetworks formed during subnetwork analysis are shown.

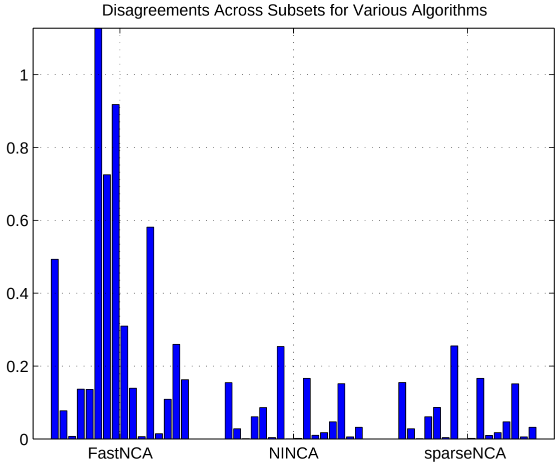


Figure 3.6: Consistency evaluation: Disagreement in the TFA estimation using FastNCA, NINCA, and sparseNCA.

The average of the four subnetworks is plotted in Figure 3.4 and the estimates from FastNCA and NINCA are also shown. It is observed that the estimates of sparseNCA agree with NINCA for all the TFAs. Moreover, they also agree with those reported in [28] except for a few TFAs. In particular, the estimate for CRP has been validated by measuring it experimentally in [28] and sparseNCA provides the same result. The disagreement in the TFA estimation for the three algorithms is calculated using (2.22) and the results are plotted in Figure 3.6. Moreover, the standard deviation observed in estimating the TFAs is also plotted in Figure 3.5 for all the algorithms and it can be seen that sparseNCA results in a very low variability in the estimates. It can therefore be concluded that sparseNCA estimates the TFA with a high degree of reliability and consistency.

SparseNCA is also applied to the yeast cell-cycle data [33] discussed in Section 2 and compared with NINCA and FastNCA. Subnetwork analysis is performed as explained previously and the average of TFA estimates are shown in Figure B.1. These

Table 3.1: Average computational time for various methods in seconds for *E. coli* dataset.

Subset	1	2	3	4
FastNCA	0.014	0.007	0.007	0.008
ℓ_1 norm	5.8	5.2	5.01	5.2
sparseNCA	0.05	0.04	0.03	0.03
NI-NCA LP	0.93	0.76	0.73	0.83
NI-NCA QP	0.59	0.13	0.13	0.13

simulations further validate the high estimation accuracy of sparseNCA compared to its predecessors, FastNCA and NINCA.

3.4.2.1 Computational Complexity Comparison

Gene regulatory network inference often deals with high dimensional biological datasets. Therefore, an important feature of the network inference algorithms is low computational complexity while sacrificing little on estimation accuracy. In order to compare the computational complexity of sparseNCA with FastNCA and NINCA, the run time complexity of the algorithms is calculated for the four subnetworks of *E. coli* data. Matlab 7.10.0 was used to perform these simulations on a Windows 7 system with a 1.90 GHz Intel Core i7 processor and the run times are given in Table 3.1. It is noted that the solution with ℓ_1 has a very high run time but the closed form solution for sparseNCA is many times faster. The sparseNCA is tens of times faster than NINCA and is comparable to FastNCA which has the lowest computational complexity among the existing NCA algorithms.

Therefore, it can be concluded from these experiments on synthetic and real data that sparseNCA not only offers higher estimation accuracy, particularly in case of incomplete prior information, and yields higher consistency but provides these benefits at a very low computational cost.

3.5 Summary

Transcriptional regulatory networks (TRNs) are known to be sparse and the information about the zeros in the connectivity matrix, which signify the absence of connections among genes, may not be completely available. The algorithms proposed thus far crucially rely on the completeness of this information. In order to account for incompleteness in the prior information, this section proposes sparseNCA algorithm which imposes a sparsity constraint on the connectivity matrix using ℓ_1 norm. Additionally, an iterative re-weighted ℓ_2 minimization algorithm is proposed for NCA by deriving a closed form solution for estimation of the connectivity matrix $\boldsymbol{A}$, which not only promotes sparsity but also significantly reduces the computational complexity. The proposed algorithm is compared with the existing algorithms including NINCA and FastNCA for synthetic data using normalized mean square error (NMSE) as the fidelity criterion. The simulations are performed for varying signal to noise ratio (SNR), correlation of data, concentration of outliers and amount of prior information available, and sparseNCA is shown to outperform the current state-of-the-art algorithms. Experiments on *E. coli* data corroborate the superior performance of sparseNCA in terms of higher consistency and estimation accuracy. These benefits make sparseNCA well suited for the inference of gene regulatory networks.

4. ROBNCA: ROBUST NETWORK COMPONENT ANALYSIS FOR RECOVERING TRANSCRIPTION FACTOR ACTIVITIES*

4.1 Introduction

It is commonly known that the microarray data are very noisy and are corrupted with outliers because of erroneous measurements and/or abnormal response of genes, and robust algorithms are required for gene network inference [14]. The decomposition techniques used to derive several NCA algorithms including FastNCA and NINCA are susceptible even to the presence of a small amount of outliers [39] and their performance is expected to deteriorate significantly when the data points are corrupted by outliers. Therefore, it is imperative to develop an NCA algorithm which has an inherent ability to mitigate the effect of outliers, and also entails low computational costs and provides good consistency and accuracy. It is precisely this avenue which is the focus of this section.

4.2 Main Contributions

The main contributions in this section can be summarized as follows:

1. A novel algorithm, ROBust Network Component Analysis (ROBNCA) [46], is proposed which has the inherent ability to counteract the presence of outliers in the data $\boldsymbol{Y}$ by explicitly modeling the outliers as an additional sparse matrix. The iterative algorithm estimates each of the parameters efficiently at each iteration, and delivers superior consistency and greater accuracy for TFA estimation.

*Reprinted with permission from "ROBNCA: Robust Network Component Analysis for Recovering Transcription Factor Activities," Amina Noor, Aitzaz Ahmad, Erchin Serpedin, Mohamed Nounou and Hazem Nounou, 2013, Bioinformatics, 29(19):2410-2418, Aug 2013., Copyright 2013 by Oxford University Press.

2. A particularly attractive feature of the ROBNCA algorithm is the derivation of a closed form solution for the estimation of the connectivity matrix $\boldsymbol{A}$, a major source of high computational complexity in contemporary algorithms. In order to further lower the computational burden, a still faster closed form solution is derived that requires matrix inversion of much smaller size. The resulting algorithm is comparable to FastNCA in terms of computational complexity, and is hundreds of times faster than NINCA.

3. The performance of ROBNCA is tested on Hemoglobin test data from [24] for both low and highly correlated source signals. ROBNCA is seen to outperform the state-of-the-art algorithms for estimating both $\boldsymbol{A}$ and $\boldsymbol{S}$ in terms of mean square error (MSE). In addition, ROBNCA is applied to yeast cell cycle data [33] and *E. coli* data [28] and by plotting the standard deviation of estimates, it is observed that ROBNCA offers better consistency than FastNCA and NINCA.

4.3 NCA with Outliers

Most of the contemporary algorithms have studied the gene network construction problem using NCA with Gaussian noise models. However, inaccuracies in measurement procedures and abnormal gene responses often render heavier tails to the gene expression data, and Gaussian noise models may no longer be a natural fit in these cases. The decomposition techniques employed in the available algorithms are highly sensitive to the presence of outliers, i.e., the samples that do not conform to the Gaussian noise model, and their estimation capabilities are extremely susceptible to outliers. As a consequence, the gene network inference becomes unreliable for practical purposes. Therefore, we focus on deriving computationally efficient NCA algorithms which are robust to the presence of outliers.

Towards this end, we take the approach of explicitly modeling the outliers as an additional matrix that corrupts the data points. From (1.6), it follows that the complete system model that accounts for the presence of outliers as well as noise can be expressed as

$$\boldsymbol{Y} = \boldsymbol{A}\boldsymbol{S} + \boldsymbol{O} + \boldsymbol{\Gamma} \ , \tag{4.1}$$

where the matrix $\boldsymbol{O}$ denotes the outliers. The outlier matrix $\boldsymbol{O}$ is a column sparse matrix since there are typically a few outliers. The joint optimization problem for the estimation of the three parameters, that also allow for controlling outlier sparsity, can be formulated as

$$\begin{aligned} \left\{\hat{\boldsymbol{A}}, \hat{\boldsymbol{S}}, \hat{\boldsymbol{O}}\right\} = \arg \ \min_{\boldsymbol{A},\boldsymbol{S},\boldsymbol{O}} \|\boldsymbol{Y} - \boldsymbol{A}\boldsymbol{S} - \boldsymbol{O}\|_F^2 + \lambda_0 \|\boldsymbol{O}\|_0 \\ \text{such that} \quad \boldsymbol{A}(I) = 0 \ , \end{aligned} \tag{4.2}$$

where the non-convex l_0 norm $\|\boldsymbol{O}\|_0$ denotes the number of nonzero columns in $\boldsymbol{O}$, and the extent of sparsity in the columns of $\boldsymbol{O}$ is controlled by the tuning parameter λ_0. The optimization problem in (4.2) is reminiscent of compressive sampling techniques based on the l_0 norm, and are known to be NP-hard [64]. Therefore, some relaxation is needed in order to solve the joint optimization problem without incurring exponentially increasing computational complexity. A viable alternative is the column-wise l_2 sum, i.e., $\|\boldsymbol{O}\|_{2,c} = \sum_{k=1}^{K} \|\boldsymbol{o}_k\|_2$, which is the closest convex approximation of $\|\boldsymbol{O}\|_0$ [64]. With this relaxation, the resulting joint optimization problem can be expressed as

$$\begin{aligned} \left\{\hat{\boldsymbol{A}}, \hat{\boldsymbol{S}}, \hat{\boldsymbol{O}}\right\} = \arg \ \min_{\boldsymbol{A},\boldsymbol{S},\boldsymbol{O}} \|\boldsymbol{Y} - \boldsymbol{A}\boldsymbol{S} - \boldsymbol{O}\|_F^2 + \lambda_2 \|\boldsymbol{O}\|_{2,c} \\ \text{such that} \quad \boldsymbol{A}(I) = 0 \ . \end{aligned} \tag{4.3}$$

Our goal is to estimate the three parameters $\boldsymbol{A}$, $\boldsymbol{S}$ and $\boldsymbol{O}$ by solving the optimization problem (4.3). However, it can be noticed that the optimization problem is not jointly convex with respect to (w.r.t) $\{\boldsymbol{A}, \boldsymbol{S}, \boldsymbol{O}\}$. Therefore, we resort to an iterative algorithm that alternately optimizes (4.3) w.r.t one parameter at a time.

4.4 ROBNCA Algorithm

The update of each of the parameters, $\boldsymbol{S}(j)$, $\boldsymbol{A}(j)$ and $\boldsymbol{O}(j)$, at an iteration j is next presented.

4.4.1 Update of the TFA Matrix

At iteration j, the value of the parameter $\boldsymbol{S}(j)$ is updated by minimizing the objective function (4.3) w.r.t $\boldsymbol{S}$, while fixing the parameters $\boldsymbol{A}$ and $\boldsymbol{O}$ to their respective values at iteration $(j-1)$. By defining the matrix $\boldsymbol{X}(j) = \boldsymbol{Y} - \boldsymbol{O}(j-1)$, the optimization problem can be written as

$$\boldsymbol{S}(j) = \arg \min_{\boldsymbol{S}} \|\boldsymbol{X}(j) - \boldsymbol{A}(j-1)\boldsymbol{S}\|_F^2 \, . \tag{4.4}$$

Since the connectivity matrix $\boldsymbol{A}(j-1)$ has full column rank (by virtue of NCA criterion 1), the matrix $\boldsymbol{A}^T(j-1)\boldsymbol{A}(j-1)$ is invertible. Therefore, an estimate of the TFA matrix $\boldsymbol{S}$ at the j^{th} iteration can be readily expressed as

$$\boldsymbol{S}(j) = \left(\boldsymbol{A}^T(j-1)\boldsymbol{A}(j-1)\right)^{-1} \boldsymbol{A}^T(j-1)\boldsymbol{X}(j) \, . \tag{4.5}$$

The estimate $\boldsymbol{S}(j)$, so obtained, is used in the upcoming steps to determine $\boldsymbol{A}$ and $\boldsymbol{O}$.

4.4.2 Update of the Connectivity Matrix

The next step in the iterative algorithm is to solve the optimization problem (4.3) w.r.t the matrix $\boldsymbol{A}$, while fixing the values of the parameters $\boldsymbol{S}$ and $\boldsymbol{O}$ to $\boldsymbol{S}(j)$ and $\boldsymbol{O}(j-1)$, respectively. The resulting optimization problem can be written as

$$\begin{aligned} \boldsymbol{A}(j) = \arg\ &\min_{\boldsymbol{A}} \|\boldsymbol{X}(j) - \boldsymbol{A}\boldsymbol{S}(j)\|_F^2 \\ &\text{such that} \quad \boldsymbol{A}(I) = 0\ . \end{aligned} \tag{4.6}$$

Remark 5. *The optimization problem* (4.6) *was also considered in the original work on NCA by Liao et. al. [34]. However, a closed form solution was not provided and the proposed algorithm relied on costly optimization techniques to update the matrix* $\boldsymbol{A}$*. Since this minimization needs to be performed at each iteration until convergence, the ALS algorithm is known to be extremely slow for large networks, and the required computational resources may be prohibitive [24]. Hence, it is imperative that a closed form solution is obtained for the optimization problem in* (4.6) *so that the algorithm is faster and efficient.*

Without loss of generality, we can consider the transposed system

$$\tilde{\boldsymbol{X}} = \tilde{\boldsymbol{S}}\tilde{\boldsymbol{A}} + \tilde{\boldsymbol{\Gamma}}\ , \tag{4.7}$$

where $\tilde{\boldsymbol{X}}$, $\tilde{\boldsymbol{S}}$, $\tilde{\boldsymbol{A}}$, and $\tilde{\boldsymbol{\Gamma}}$ denote the transpose of the original matrices, respectively. The resulting equivalent optimization problem can now be stated as

$$\begin{aligned} \tilde{\boldsymbol{A}}(j) = \arg\ &\min_{\tilde{\boldsymbol{A}}} \|\tilde{\boldsymbol{X}}(j) - \tilde{\boldsymbol{S}}(j)\tilde{\boldsymbol{A}}\|_F^2 \\ &\text{such that} \quad \tilde{\boldsymbol{A}}(\tilde{I}) = 0\ , \end{aligned} \tag{4.8}$$

where $\tilde{I}$ is the set of all indices where the entries of the matrix $\tilde{\boldsymbol{A}}$ are known to be zero. The following theorem presents a closed form solution for the optimization problem (4.8), herein referred to as ROBNCA 1.

Theorem 1. *The solution of* (4.8) *at the* j^{th} *iteration is given by*

$$\tilde{\boldsymbol{a}}_n(j) = \boldsymbol{Q}^{-1}(j)\left[\tilde{\boldsymbol{w}}_n(j) - \boldsymbol{C}_n^T\boldsymbol{\Psi}^{-1}(j)\boldsymbol{C}_n\boldsymbol{Q}^{-1}(j)\tilde{\boldsymbol{w}}_n(j)\right] , \tag{4.9}$$

where $\boldsymbol{\Psi}(j) = \boldsymbol{C}_n\boldsymbol{Q}^{-1}(j)\boldsymbol{C}_n^T$, $\tilde{\boldsymbol{w}}_n(j) = \tilde{\boldsymbol{S}}^T(j)\tilde{\boldsymbol{x}}_n(j)$, *the symmetric matrix* $\boldsymbol{Q}(j) = \tilde{\boldsymbol{S}}^T(j)\tilde{\boldsymbol{S}}(j)$, *and* $\tilde{\boldsymbol{a}}_n$ *and* $\tilde{\boldsymbol{x}}_n$ *represent the* n^{th} *columns of matrices* $\tilde{\boldsymbol{A}}$ *and* $\tilde{\boldsymbol{X}}$, *respectively. The* $L_n \times M$ *matrix* $\boldsymbol{C}_n$ *is a matrix of zeros except* $\boldsymbol{C}_n(\tilde{I}_n) = 1$, *where* $\tilde{I}_n$ *is the set of indices where the entries of* $\tilde{\boldsymbol{a}}_n$ *are zero, and* L_n *denotes the number of zero entries in* $\tilde{\boldsymbol{a}}_n$.

Proof. We begin the proof by first noting that the objective function is separable in terms of the columns of the optimization variable $\tilde{\boldsymbol{A}}$. Using its definition, the Frobenius norm of an $M \times N$ matrix $\boldsymbol{Z}$ can be written as

$$\begin{aligned}\|\boldsymbol{Z}\|_F^2 &= \mathrm{Tr}\left(\boldsymbol{Z}^T\boldsymbol{Z}\right)\\ &= \sum_{n=1}^{N}\|\boldsymbol{z}_n\|^2\end{aligned} \tag{4.10}$$

where $\boldsymbol{z}_n$ is the n^{th} column of $\boldsymbol{Z}$. The n^{th} column of (4.7) can be written as

$$\tilde{\boldsymbol{x}}_n = \tilde{\boldsymbol{S}}\tilde{\boldsymbol{a}}_n + \tilde{\boldsymbol{\gamma}}_n . \tag{4.11}$$

The objective function in (4.8) can be equivalently expressed as

$$\|\tilde{\boldsymbol{X}}(j) - \tilde{\boldsymbol{S}}(j)\tilde{\boldsymbol{A}}\|_F^2 = \sum_{n=1}^{N}\|\tilde{\boldsymbol{x}}_n(j) - \tilde{\boldsymbol{S}}(j)\tilde{\boldsymbol{a}}_n\|^2 . \tag{4.12}$$

The constraint $\tilde{\boldsymbol{A}}(\tilde{I}) = 0$ can be written as a set of n constraints $\boldsymbol{C}_n \tilde{\boldsymbol{a}}_n = \boldsymbol{0}$ for $n = 1, \ldots, N$. The $L_n \times M$ matrix $\boldsymbol{C}_n$ is constructed such that it consists of all zeroes except $\boldsymbol{C}_n(\tilde{I}_n) = 1$. For instance, if $M = 6$, and $\tilde{\boldsymbol{a}}_n = [a_{n_1}, a_{n_2}, 0, a_{n_4}, 0, a_{n_6}]^T$, the 2×6 matrix $\boldsymbol{C}_n$ consists of all zeroes except $\boldsymbol{C}_n(1,3) = \boldsymbol{C}_n(2,5) = 1$. It can be easily verified that the matrix $\boldsymbol{C}_n$ so constructed has full row rank.

The optimization problem in (4.8) can now be written as

$$\begin{aligned} \tilde{\boldsymbol{A}}(j) &= \arg \min_{\tilde{\boldsymbol{A}}} \sum_{n=1}^{N} \|\tilde{\boldsymbol{x}}_n(j) - \tilde{\boldsymbol{S}}(j)\tilde{\boldsymbol{a}}_n\|^2 \\ &\text{such that} \quad \boldsymbol{C}_n \tilde{\boldsymbol{a}}_n = \boldsymbol{0}, \quad \forall n = 1, \ldots, N \ . \end{aligned} \tag{4.13}$$

The optimization problem is, therefore, separable in terms of columns of $\tilde{\boldsymbol{A}}$, and can be equivalently solved by considering one column at a time. This also reduces the computational complexity of estimating the connectivity matrix $\tilde{\boldsymbol{A}}$. Henceforth, we will employ convex optimization techniques to derive a closed form solution of the separable optimization problem. For the n^{th} column, we have

$$\begin{aligned} \tilde{\boldsymbol{a}}_n(j) &= \arg \min_{\tilde{\boldsymbol{a}}_n} \frac{1}{2}\tilde{\boldsymbol{a}}_n^T \boldsymbol{Q}(j)\tilde{\boldsymbol{a}}_n - \tilde{\boldsymbol{w}}_n^T(j)\tilde{\boldsymbol{a}}_n \\ &\text{such that} \quad \boldsymbol{C}_n \tilde{\boldsymbol{a}}_n = \boldsymbol{0} \ , \end{aligned} \tag{4.14}$$

where the objective function is re-scaled and terms independent of $\tilde{\boldsymbol{a}}_n$ are neglected. The Lagrangian dual function can be expressed as

$$\mathcal{L}(\tilde{\boldsymbol{a}}_n, \boldsymbol{\mu}) = \frac{1}{2}\tilde{\boldsymbol{a}}_n^T \boldsymbol{Q}(j)\tilde{\boldsymbol{a}}_n - \tilde{\boldsymbol{w}}_n^T(j)\tilde{\boldsymbol{a}}_n + \boldsymbol{\mu}^T \boldsymbol{C}_n \tilde{\boldsymbol{a}}_n \ .$$

The Karush-Kuhn-Tucker (KKT) conditions can be written as [4]

$$\boldsymbol{Q}(j)\tilde{\boldsymbol{a}}_n - \tilde{\boldsymbol{w}}_n(j) + \boldsymbol{C}_n^T \boldsymbol{\mu} = \boldsymbol{0} \tag{4.15}$$

$$\boldsymbol{C}_n \tilde{\boldsymbol{a}}_n = \boldsymbol{0} \ . \tag{4.16}$$

Lemma 1. *The KKT conditions are necessary and sufficient for the optimization problem* (4.14)*.*

Proof. Since the optimization problem (4.14) contains linear equality constraints, the KKT conditions are necessary for optimality [4]. Let any $\tilde{\boldsymbol{a}}_n^*$ be a local minimum. Then, since the KKT conditions are necessary, there exists a Lagrange multiplier $\boldsymbol{\mu}^*$ such that $(\tilde{\boldsymbol{a}}_n^*, \boldsymbol{\mu}^*)$ is the solution to the system of equations in (4.15) and (4.16). Now since the objective function is convex, it follows that $\tilde{\boldsymbol{a}}_n^*$ is also a global minimum [4]. This implies that the KKT conditions are also sufficient for optimality. □

Hence, a solution to (4.14) can be obtained by solving the KKT system of equations. Using (4.15), it follows that

$$\tilde{\boldsymbol{a}}_n = \boldsymbol{Q}^{-1}(j)\left(\tilde{\boldsymbol{w}}(j) - \boldsymbol{C}_n^T \boldsymbol{\mu}\right) \ , \tag{4.17}$$

where the matrix $\boldsymbol{Q}(j)$ is indeed invertible by virtue of the linear independence of the rows of $\boldsymbol{S}$ (NCA criterion 3). Substituting (4.17) into (4.16), we have

$$\boldsymbol{C}_n \boldsymbol{Q}^{-1}(j) \boldsymbol{C}_n^T \boldsymbol{\mu} = \boldsymbol{C}_n \boldsymbol{Q}^{-1}(j) \tilde{\boldsymbol{w}}(j) \ .$$

Since the matrix $\boldsymbol{C}_n$ has full row rank, the matrix $\boldsymbol{\Psi}(j) \triangleq \boldsymbol{C}_n \boldsymbol{Q}^{-1}(j) \boldsymbol{C}_n^T$ is invertible.

The Lagrange multiplier can, therefore, be expressed as

$$\boldsymbol{\mu} = \boldsymbol{\Psi}^{-1}(j)\boldsymbol{C}_n\boldsymbol{Q}^{-1}(j)\tilde{\boldsymbol{w}}(j)\ . \tag{4.18}$$

Upon substituting (4.18) into (4.17), the solution $\tilde{\boldsymbol{a}}_n$ in Theorem 1 readily follows. □

Therefore, using Theorem 1, an estimate of $\tilde{\boldsymbol{A}}(j)$ can be efficiently obtained and this approach results in substantial reduction in computation complexity compared to the ALS algorithm.

Remark 6. *While the aforementioned closed form solution provides a significant advantage in terms of computational complexity over the ALS algorithm, we note that the solution requires inverting the matrix $\boldsymbol{Q}$. For large networks, this can potentially be a large matrix, whose inverse incurs computational load, and may lead to inaccuracies as well. In the following discussion, we derive a still faster algorithm, ROBNCA 2, that takes advantage of the special structure of the column vector $\tilde{\boldsymbol{a}}_n$ and provides added savings over the closed form solution derived in Theorem 1.*

We begin by noting that the rows of $\tilde{\boldsymbol{X}}$ and $\tilde{\boldsymbol{A}}$ can always be reordered in (4.7). Hence, without loss of generality, the vector $\tilde{\boldsymbol{a}}_n$ can be partitioned as

$$\tilde{\boldsymbol{a}}_n = \begin{bmatrix} \bar{\boldsymbol{a}}_n \\ \boldsymbol{0}_{L_n \times 1} \end{bmatrix}\ , \tag{4.19}$$

where $\bar{\boldsymbol{a}}_n \in \mathcal{R}^{(M-L_n)\times 1}$ is a vector consisting of the non-zero entries in $\tilde{\boldsymbol{a}}_n$. Construct an $L_n \times M$ matrix $\boldsymbol{U}_n$ such that

$$\boldsymbol{U}_n = \begin{bmatrix} \boldsymbol{0}_{L_n \times (M-L_n)} & \boldsymbol{I}_{L_n} \end{bmatrix}\ . \tag{4.20}$$

With the above definition, the optimization problem (4.14) can be equivalently represented as

$$\tilde{\boldsymbol{a}}_n(j) = \arg \min_{\tilde{\boldsymbol{a}}_n} \frac{1}{2}\tilde{\boldsymbol{a}}_n^T \boldsymbol{Q}(j)\tilde{\boldsymbol{a}}_n - \tilde{\boldsymbol{w}}_n^T(j)\tilde{\boldsymbol{a}}_n$$

$$\text{such that} \quad \boldsymbol{U}_n\tilde{\boldsymbol{a}}_n = \mathbf{0} \ . \tag{4.21}$$

Define the substitution

$$\tilde{\boldsymbol{a}}_n = \boldsymbol{V}_n\bar{\boldsymbol{a}}_n \ , \tag{4.22}$$

where the $M \times L_n$ matrix $\boldsymbol{V}_n$ is constructed such that it lies in the null space of the matrix $\boldsymbol{U}_n$, i.e., $\boldsymbol{U}_n\boldsymbol{V}_n = \mathbf{0}$. The matrix $\boldsymbol{V}_n$ is, therefore, given by

$$\boldsymbol{V}_n = \begin{bmatrix} \boldsymbol{I}_{(M-L_n)} \\ \mathbf{0}_{L_n\times(M-L_n)} \end{bmatrix} \ . \tag{4.23}$$

By substituting $\tilde{\boldsymbol{a}}_n$ from (4.22) into (4.21), and noting that the constraint is always satisfied due to the construction of $\boldsymbol{V}_n$, we have an unconstrained optimization problem in the variable $\bar{\boldsymbol{a}}_n$ given by

$$\bar{\boldsymbol{a}}_n(j) = \arg \min_{\bar{\boldsymbol{a}}_n} \frac{1}{2}\bar{\boldsymbol{a}}_n^T \boldsymbol{V}_N^T \boldsymbol{Q}(j)\boldsymbol{V}_N\bar{\boldsymbol{a}}_n - \tilde{\boldsymbol{w}}_n^T(j)\boldsymbol{V}_N\bar{\boldsymbol{a}}_n \ . \tag{4.24}$$

The solution of the aforementioned unconstrained quadratic optimization problem can be easily obtained as

$$\bar{\boldsymbol{a}}_n(j) = \left(\boldsymbol{V}_n^T\boldsymbol{Q}(j)\boldsymbol{V}_n\right)^{-1}\boldsymbol{V}_n^T\tilde{\boldsymbol{w}}_n(j) \ , \tag{4.25}$$

where the matrix $\boldsymbol{V}_n^T\boldsymbol{Q}(j)\boldsymbol{V}_n$ is invertible since $\boldsymbol{V}_n$ has full column rank.

The symmetric invertible matrix $\boldsymbol{Q}(j)$ can be partitioned as

$$\boldsymbol{Q}(j) = \begin{bmatrix} \boldsymbol{Q}_{11}(j) & \boldsymbol{Q}_{12}(j) \\ \boldsymbol{Q}_{21}(j) & \boldsymbol{Q}_{22}(j) \end{bmatrix} ,$$

where the invertible matrix $\boldsymbol{Q}_{11}(j)$ is the upper $(M - L_n) \times (M - L_n)$ submatrix of $\boldsymbol{Q}(j)$. From the structure of $\boldsymbol{V}_n$, the matrix $\boldsymbol{V}_n^T \boldsymbol{Q}(j) \boldsymbol{V}_n$ can be reduced as

$$\begin{aligned} & \boldsymbol{V}_n^T \boldsymbol{Q}(j) \boldsymbol{V}_n \\ &= \begin{bmatrix} \boldsymbol{I}_{(M-L_n)} & \mathbf{0}_{(M-L_n)\times L_n} \end{bmatrix} \begin{bmatrix} \boldsymbol{Q}_{11}(j) & \boldsymbol{Q}_{12}(j) \\ \boldsymbol{Q}_{21}(j) & \boldsymbol{Q}_{22}(j) \end{bmatrix} \begin{bmatrix} \boldsymbol{I}_{(M-L_n)} \\ \mathbf{0}_{L_n\times(M-L_n)} \end{bmatrix} \\ &= \boldsymbol{Q}_{11}(j) \; . \end{aligned} \tag{4.26}$$

Similarly, by partitioning $\tilde{\boldsymbol{w}}_n(j)$ as

$$\tilde{\boldsymbol{w}}_n(j) = \begin{bmatrix} \bar{\boldsymbol{w}}_n(j) \\ \hat{\boldsymbol{w}}_n(j) \end{bmatrix} ,$$

it follows that

$$\boldsymbol{V}_n^T \tilde{\boldsymbol{w}}_n(j) = \bar{\boldsymbol{w}}_n(j) \; , \tag{4.27}$$

where $\bar{\boldsymbol{w}}_n(j)$ is the upper $(M - L_n) \times 1$ vector of $\tilde{\boldsymbol{w}}_n(j)$. Collecting all the terms, the solution $\bar{\boldsymbol{a}}_n$ can now be compactly represented as

$$\bar{\boldsymbol{a}}_n(j) = \boldsymbol{Q}_{11}^{-1}(j) \bar{\boldsymbol{w}}_n(j) \; . \tag{4.28}$$

Once all columns $\tilde{\boldsymbol{a}}_n(j)$ are determined, the connectivity matrix $\boldsymbol{A}(j)$ can be easily updated.

Remark 7. *By comparing the closed form solution derived in* (4.9) *with* (4.28)*, it is clear that the latter only requires inverting the submatrix* $\boldsymbol{Q}_{11}(j)$ *of* $\boldsymbol{Q}(j)$*. Since the connectivity matrix is usually sparse and the number of non zero entries* $(M - L_n)$ *in the* n^{th} *column is usually very small, inverting the* $(M - L_n) \times (M - L_n)$ *matrix* $\boldsymbol{Q}_{11}(j)$ *results in a considerable reduction in computational complexity and ensures a much faster implementation of the iterative algorithm.*

The respective computational times incurred in calculating (4.9) and (4.28) will be quantified in Section 4.5 to emphasize the usefulness of deriving (4.28).

4.4.3 Update of the Outlier Matrix

The last step in the iterative algorithm pertains to the estimation of the outlier matrix $\boldsymbol{O}$ by using the values $\boldsymbol{S}(j)$ and $\boldsymbol{A}(j)$ obtained in the preceding steps. It is straightforward to notice that the optimization problem (4.3) w.r.t $\boldsymbol{O}$ decouples across the columns and results in K subproblems, each of which being expressed as follows:

$$\boldsymbol{o}_k(j) = \arg\min_{\boldsymbol{o}_k} \|\boldsymbol{b}_k(j) - \boldsymbol{o}_k\|_2^2 + \lambda_2 \|\boldsymbol{o}_k\|_2 \ , \tag{4.29}$$

where $\boldsymbol{b}_k(j) = \boldsymbol{y}_k - \boldsymbol{A}(j)\boldsymbol{s}_k(j)$. The solution to (4.29) is given by [31]

$$\boldsymbol{o}_k(j) = \frac{\boldsymbol{b}_k(j)\left(\|\boldsymbol{b}_k(j)\|_2 - \frac{\lambda_2}{2}\right)_+}{\|\boldsymbol{b}_k(j)\|_2}, \quad k = 1, \ldots, K \tag{4.30}$$

where $(g)_+ \triangleq \max(0, g)$. The solution (4.30) is intuitively satisfying since it sets the outlier $\boldsymbol{o}_k(j)$ to zero whenever $\|\boldsymbol{b}_k(j)\|_2$ fails to exceed the threshold $\lambda_2/2$, where λ_2 is the sparsity-controlling parameter. Several approaches have been identified in the literature for selecting λ_2 which depend on any a-priori information available about the extent of sparsity [17]. If the concentration of outliers is unknown, a typical rule of thumb is to take $\lambda_2 = 0.7$ where this value has been determined to provide 95%

asymptotic efficiency of the estimator [31]. If a rough estimate of the concentration of outliers is available, (4.29) can be solved for a grid of values and selecting the λ_2 giving the expected number of outliers which can be performed efficiently using the Group-LARS algorithm [76]. It is noted, that the performance of the algorithm is insensitive to minor variations in the value of the parameter. Since the subproblems at each iteration have unique minimizers, and the non-differentiable regularization affects only the outlier matrix $\boldsymbol{O}$, the convergence of the ROBNCA algorithm is established using the results in [65].

Proposition 2. *As $j \to \infty$, the iterates generated by the ROBNCA algorithm converge to a stationary point of* (4.3).

It is important to point out that ROBNCA is significantly different from NINCA algorithm. NINCA, as the name suggests, is a non-iterative algorithm which uses a subspace based method for the estimation of the connectivity matrix $\boldsymbol{A}$ using eigen-decomposition and relies on solving a constrained quadratic optimization problem, which has high computational cost. On the other hand, in ROBNCA, we propose two closed form solutions for the estimation of the connectivity matrix $\boldsymbol{A}$ which result in considerable reduction in computational complexity. The steps of the ROBNCA algorithm are summarized in Algorithm 2. The iterations of the ROBNCA algorithm are stopped when the update in the objective function in (4.3) is less than $\epsilon\%$ of its value at the previous iteration.

4.5 Results and Discussion

This section investigates the observed performance of ROBNCA, in comparison with the state-of-the-art algorithms including FastNCA, NINCA, and ALS in terms of MSE using both synthetic and real data. The efficiency and consistency of ROBNCA in estimating the TFAs under various scenarios is also illustrated. The data sets for all of the experiments as well as the MATLAB implementation of FastNCA and

Algorithm 2 `ROBNCA`

```
Initialize $\boldsymbol{A}(0) = \boldsymbol{I}$ and $\boldsymbol{O}(0) = \boldsymbol{0}$.
Set $\lambda_2$ and $\epsilon$.
for $j = 1, 2, ...,$ do
    Update $\boldsymbol{S}(j) = \left(\boldsymbol{A}^T(j-1)\boldsymbol{A}(j-1)\right)^{-1} \boldsymbol{A}^T(j-1)\boldsymbol{X}(j)$.
    for $n = 1, 2, ..., N$ do
        Update $\bar{\boldsymbol{a}}_n(j) = \boldsymbol{Q}_{11}^{-1}(j)\bar{\boldsymbol{w}}_n(j)$ using (4.28).
    end for
    Form updated matrix $\boldsymbol{A}(j) = [\tilde{\boldsymbol{a}}_1^T(j) \quad \tilde{\boldsymbol{a}}_2^T(j) \ldots \tilde{\boldsymbol{a}}_N^T(j)]^T$.
    for $k = 1, ..., K$ do
        Update the outlier column $\boldsymbol{o}_k(j) = \frac{\boldsymbol{b}_k(j)\left(\|\boldsymbol{b}_k(j)\|_2 - \frac{\lambda_2}{2}\right)_+}{\|\boldsymbol{b}_k(j)\|_2}$
    end for
end for
return
```

NINCA are downloaded from `http://www.seas.ucla.edu/~liaoj/download.htm` and `http://www.ece.ucdavis.edu/~jacklin/NCA`, respectively.

4.5.1 Synthetic and Hemoglobin Test Data

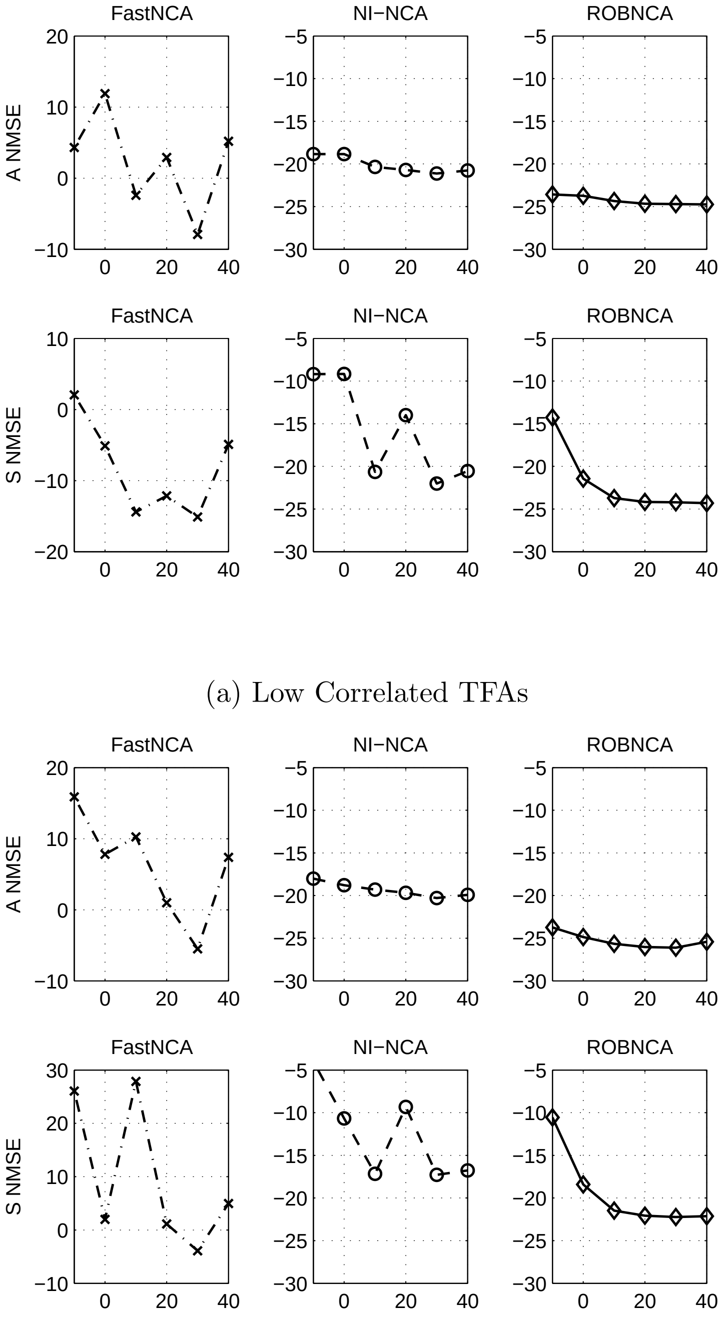
First, in order to evaluate the performance of various algorithms, test data from [34] is used. The spectroscopy data consists of $M = 7$ hemoglobin solutions formed by mixing up $N = 3$ pure hemoglobin components. The connectivity matrix in this case represents the concentration and presence or absence of each component in the mixture. In addition, the structure of this matrix is validated to comply with the NCA criteria. The absorbance spectra are taken for wavelengths in the range of 380nm to 700nm with 1nm increments to get $K = 7$ observation points which is defined as $\boldsymbol{Abs} = \boldsymbol{C\epsilon}$ [34], where the rows of $\boldsymbol{Abs}$ give the absorbance spectra for the range of wavelengths, $\boldsymbol{C}$ denotes the connectivity matrix and $\boldsymbol{\epsilon}$ gives the spectra of the pure components. The importance of using this dataset is that this experiment mimics the gene regulatory network very closely and contains all of its key properties. Knowledge of the pure spectra helps us to effectively evaluate the performance of various NCA algorithms. In addition, using the data from [34] and [24] ensures a fair comparison.

The proposed algorithm is tested against varying noise for two very important scenarios: (a) when the source signals are correlated, and (b) the observed data is corrupted with outliers. Using the same connectivity matrix, source signals were generated with low, moderate and high correlation [24]. The outliers were artificially added to the data by modeling them as a Bernoulli process. The success probability indicates the concentration of outliers present and is assumed to be the same for all the genes. Since only a few points are expected to be corrupted in the real data, the outliers are assumed to be sparse and therefore the success probability for the presence of outliers is kept small.

The performance of ROBNCA, FastNCA, and NINCA is evaluated in the aforementioned scenarios. ROBNCA algorithm is implemented in MATLAB. Since the observed data matrix $\boldsymbol{Y}$ is expected to contain outlying points, the algorithms are assessed by computing the MSE incurred in estimating the matrices $\boldsymbol{A}$ and $\boldsymbol{S}$, instead of fitting error for $\boldsymbol{Y}$. The comparison with ALS is omitted here because it takes much longer to run as will be shown in the next subsection.

4.5.1.1 Impact of Correlation

The algorithms are first tested for low and highly correlated source signals by varying the signal-to-noise ratio (SNR). The noise is modeled as Gaussian in all the experiments. The results are averaged over 100 iterations and are depicted in Figure 4.1. It is observed that the presence of a small amount of outliers makes the estimation using FastNCA very unreliable and inconsistent for both low and highly correlated signals. On the other hand, NINCA is able to estimate $\boldsymbol{S}$ better than FastNCA, and the estimation of $\boldsymbol{A}$ is quite accurate and consistent as well. It can



(a) Low Correlated TFAs

(b) High Correlated TFAs

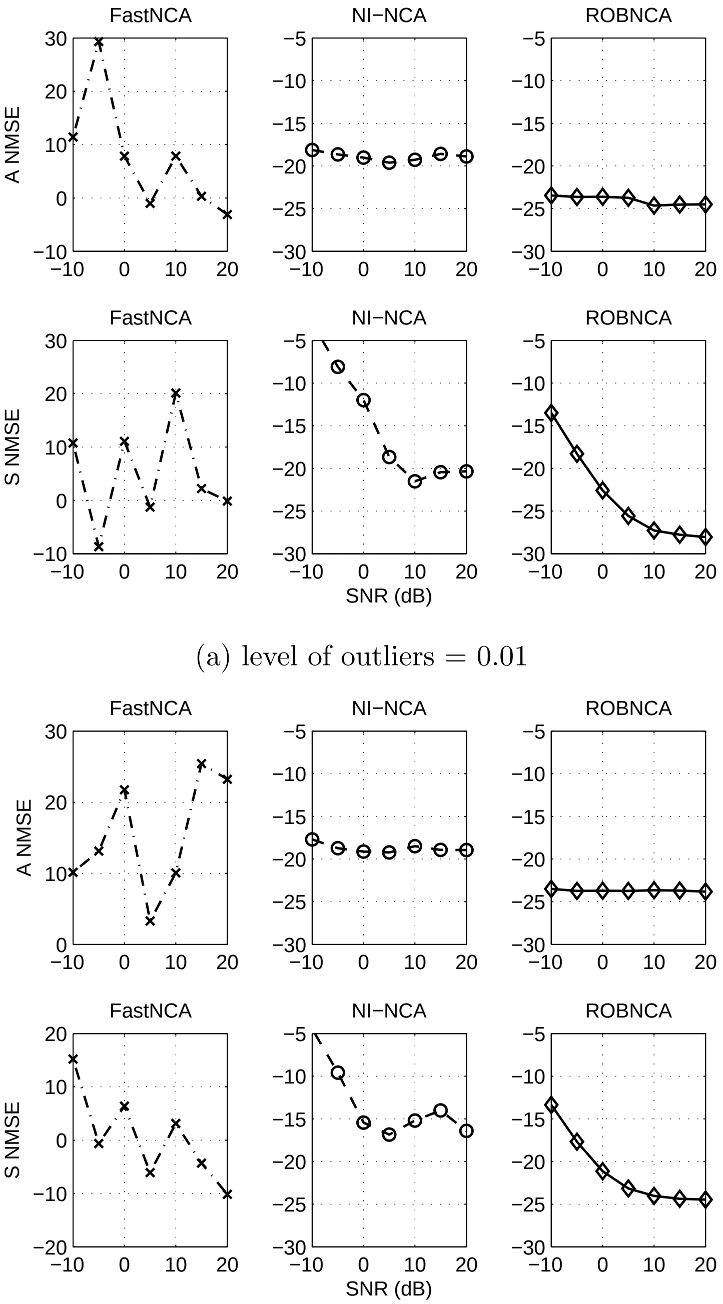
Figure 4.1: Impact of correlation: Normalized mean square error (NMSE) (dB) for different algorithms and different data sets against varying signal-to-noise (SNR) ratio(dB) with the level of outliers set to 0.05.

be observed that the overall estimation performance for $\boldsymbol{A}$ is much better and more consistent than that of $\boldsymbol{S}$. The reason for this could be attributed to the availability of some prior information for the former. Since ROBNCA takes into account the presence of outliers in the observed data, it outperforms the other two algorithms for estimating both $\boldsymbol{A}$ and $\boldsymbol{S}$ and its consistent performance should be contrasted with the unboundedness and unpredictability exhibited by the other two algorithms. In general, the performance of all the algorithms improves with the increase in SNR and degrades with the increase in correlation of the source signals.

4.5.1.2 Impact of Outliers

As noted earlier, the presence of outliers can severely affect the performance of algorithms. It is therefore, important to investigate the impact of the presence of outlying points in the observation matrix $\boldsymbol{Y}$. Comparison performed for low and high concentration of outliers is depicted in Figure 4.2. It is observed from Figure 4.2a that in the case of low concentration of outliers, NINCA provides good accuracy for $\boldsymbol{A}$ and estimates it quite consistently. The estimation of $\boldsymbol{S}$ gives a small MSE as well and generally performs consistently. FastNCA, however, is not able to estimate both the matrices even for high SNRs. This indicates its high vulnerability to the presence of even a small number of outliers. In case of a higher concentration of outliers, the performance of NINCA degrades a little bit as depicted in Figure 4.2b. It is observed that ROBNCA is able to estimate the two matrices for both low and high outliers, and outperforms the other two algorithms.

The estimation of $\boldsymbol{O}$ matrix is shown in Figure 4.3, which depicts the outliers present in the synthetic data and their estimates using ROBNCA algorithm. It is noted that ROBNCA is able to identify the outliers very well. Figure 4.4 shows the recovered signal $\boldsymbol{AS}$ after subtracting the outlier matrix O from the data matrix $\boldsymbol{X}$.



(a) level of outliers = 0.01

(b) level of outliers = 0.1

Figure 4.2: Impact of outliers: Normalized mean square error (NMSE) (dB) for different algorithms and different data sets against varying signal-to-noise (SNR) ratio(dB) for a highly correlated data set.

It can be observed that the recovered signal is a good match with the original signal.

These experiments indicate that ROBNCA solves the estimation problem with much more accuracy than NINCA and FastNCA. It is important to emphasize here that the MSE for NINCA is always higher than that of ROBNCA and its computational complexity is many times greater than the latter which can prove to be a bottle-neck in case of large data sets.

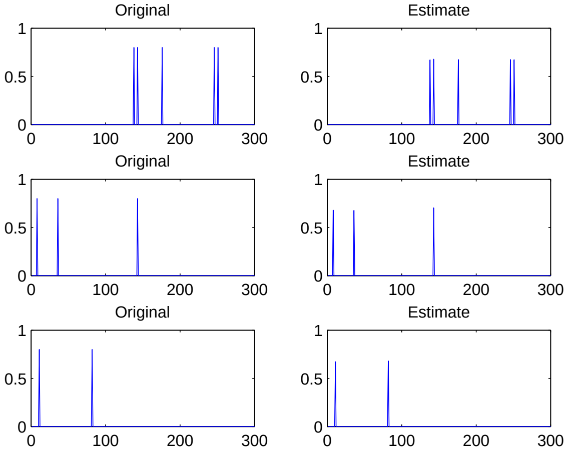


Figure 4.3: Estimation of outlier matrix O: ROBNCA estimates for three out of the seven signals in the synthetic data are shown here. It is noted that ROBNCA is able to capture the outliers well.

4.5.2 Results for Real Data

We now turn our attention to the comparison of these algorithms on real data. Two datasets are considered for this purpose which are the *S. cerevisiae* cell cycle data [33] and *E. coli* data [28]. The transcription factor activities are estimated for the TFs of interest in each experiment and the results are compared for different algorithms. In addition, the variability of the estimates is evaluated using the sub-

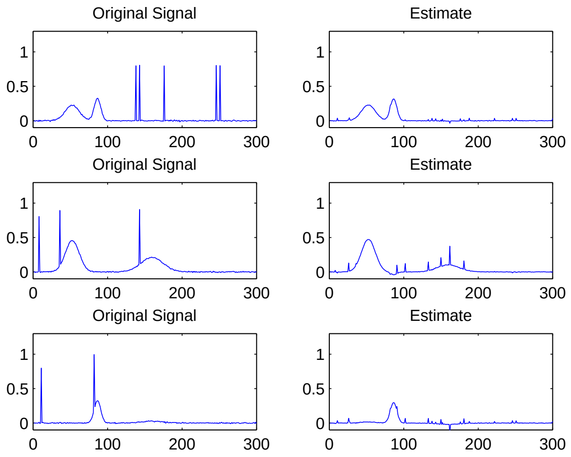


Figure 4.4: Outlier removal from the signal using ROBNCA: Three out of the seven signals in the synthetic data are shown here. The resulting $\boldsymbol{AS}$ after subtracting the $\boldsymbol{O}$ matrix from gene expression data matrix $\boldsymbol{X}$ is shown here.

network analysis [74] which will be explained in the following subsections.

4.5.2.1 S. cerevisiae *Cell Cycle Data*

The algorithms discussed in this paper are applied to the yeast cell cycle data from [33] and [60]. In order to assess the performance and variability of the various NCA algorithms, *sub-network analysis* is performed which has also been used previously in [10], [74] and [24] and the details of which have been given in Section 2.

The original network is divided into four subnetworks each consisting of 40 TFs and the number of genes varies from 921 to 1247. The aforementioned 11 TFs are included in each of the subsets. The structure of $\boldsymbol{A}$ is verified to satisfy the NCA criterion (2) for all of the sub-networks. The reconstruction of the eleven TFAs, which is the average of the four sub-networks, using ROBNCA, FastNCA, and NINCA is depicted in Figure 4.5. The TFA estimation using ALS algorithm is skipped here because the algorithm takes very long to run for this large data set. The results

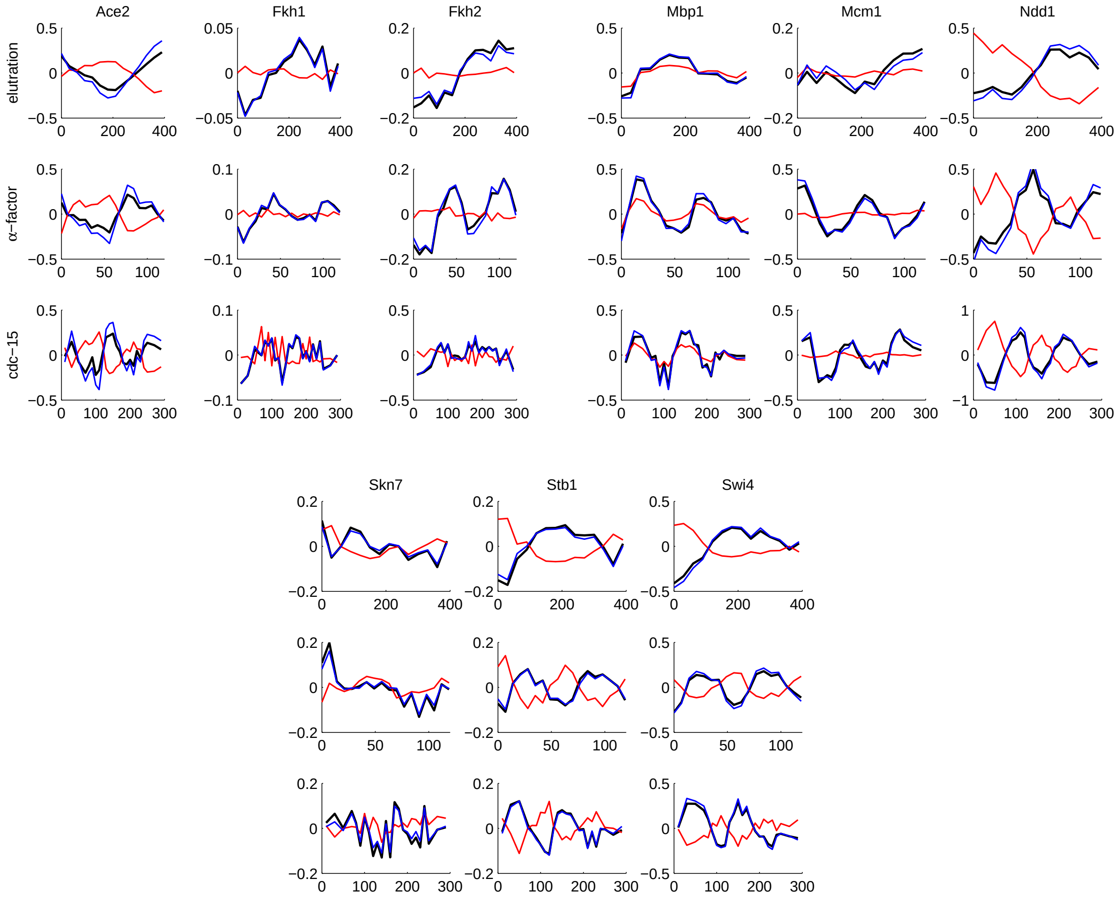


Figure 4.5: TFAs reconstruction: Estimation of 11 TFAs (9 shown) of cell-cycle regulated yeast TFs. Average values of the TFs are shown for the four subnetworks. The results of ROBNCA (black), FastNCA (red) and NINCA (blue) are given.

for the three experiments are shown separately in the three columns. The TFAs for these experiments are expected to have a periodic behavior with one, two and three cycles in elutriation, $\alpha-$factor and cdc-15, respectively, which can easily be corroborated from the figure. The results from ROBNCA differ from FastNCA in some of the instances. On the other hand, NINCA provides very similar estimates to that of ROBNCA. It can be inferred that the results of these two algorithms are more reliable as compared to FastNCA because the former reveal the periodic behavior in almost all of the TFs.

We now look to investigate the consistency of the algorithms. The disagreement between the TFA estimates of the four sub-networks is calculated using (2.22) and the results are shown in Figure 4.6. Out of the three algorithms considered, ROBNCA incurs the smallest disagreement. The performance of NINCA is somewhat comparable, however, FastNCA shows a high degree of inconsistency.

The simulations for standard deviation for TFAs are presented in Figures C.1, C.2 and C.3 for ROBNCA, NINCA and FastNCA, respectively. It is noted that ROBNCA yields the lowest variation whereas FastNCA shows much higher variation in the TFA estimates than both the other algorithms. It can therefore be concluded that ROBNCA outperforms NINCA both in terms of estimating the TFAs as well as in terms of consistency for Yeast cell-cycle data.

4.5.2.2 E. coli *Data*

The performance of NCA algorithms is now tested for *E. coli* data. This dataset contains the gene expression profiles obtained during transition of the sole carbon source from glucose to acetate [28]. Out of 296 genes found to be of relevance during the carbon source transition, 100 genes were separated so that the resulting network satisfies the NCA criteria. A total of 16 TFs were identified to be related to this

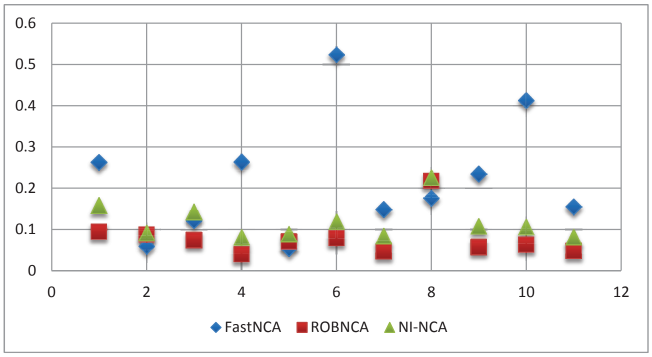


Figure 4.6: Average disagreement for different algorithms across the subsets for TFAs. X-axis indicates the TFA index. Consistency comparison for *S. cerevisiae* data.

experiment which are ArcA,CRP, CysB, FadR, FruR, GatR, IcIR, LeuO, Lrp, Narl, PhoB, PurR, RpoE, RpoS, TrpR, TyrR. We perform sub-network analysis onto this dataset in order to estimate the transcription factor activities for the 16 TFs of interest. The downloaded network is divided into four subnetworks containing 81, 82, 85 and 88 genes, respectively. The number of TFs in each subnetwork is fixed to 20, where the aforementioned 16 TFs are included in all of them. The samples are taken at 5, 15, 30, 60 mins and then every hour until 6 hours. Multiple samples are taken at these instances which make a total of 25 time points. The advantage of using this data is that the ALS algorithm can be added to the performance evaluation because of its smaller subnetworks. ALS is known to have prohibitive computational complexity [24] and is included here only for the comparison of estimation accuracy. The reconstruction of TFAs is performed using the four algorithms and the average of the TFA estimates from four subnetworks is depicted in Figure 4.7. The results from ROBNCA, NINCA and ALS are in agreement for almost all of the TFAs. In addition, these estimates are also similar to those found in [28] except for a few TFAs. The reason for this small dissimilarity could be that, in this paper the estimates are

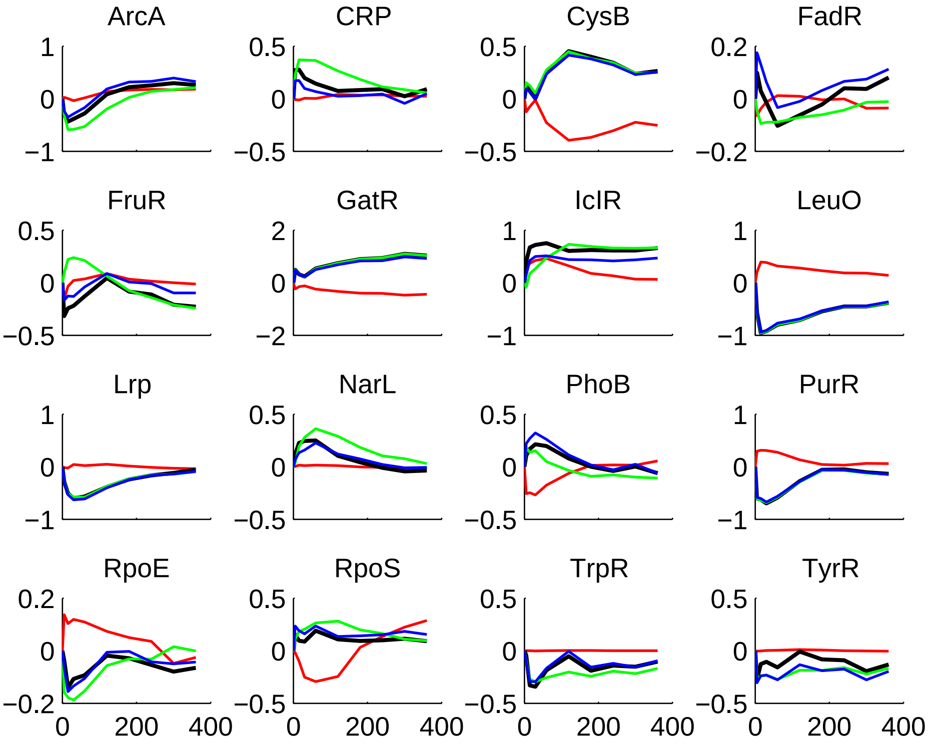


Figure 4.7: TFAs Reconstruction: Estimation of 16 TFAs of *E. coli.* Average values of TFs are shown. The results of ROBNCA (black), FastNCA (red), NINCA (blue) and ALS (green) are given.

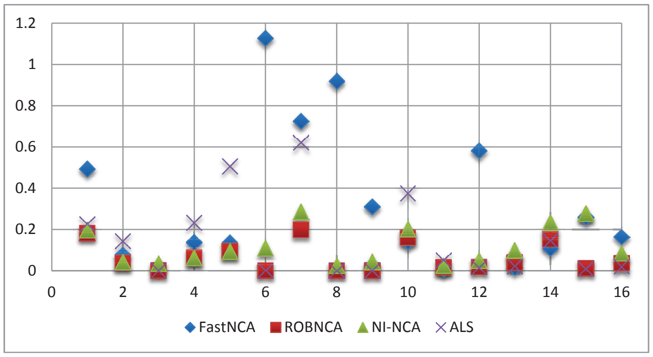


Figure 4.8: Average disagreement for different algorithms across the subsets for TFAs. X-axis indicates the TFA index. Consistency comparison for *E. coli* data.

obtained using the subnetworks whereas [28] use the complete network of 100 genes. For 5 out of the 16 TFs, namely GatR, Lrp, NarL, TrpR and TyrR, FastNCA is not able to recover the TFAs. Moreover, the TFAs predicted by ROBNCA are similar to those predicted by ALS which is the original solution as shown in Figure 5. It can therefore be inferred that ROBNCA estimates the TFAs more accurately than FastNCA.

The consistency of the algorithms is assessed for this experiment as well and the respective disagreement for each of the four algorithms is shown in Figure 4.8.

FastNCA is again seen to incur the maximum disagreement. NINCA and ALS perform better than FastNCA, however, ROBNCA yields the least disagreement for the four estimates of TFAs and performs the most consistently out of all the algorithms.

4.5.2.3 Computational Complexity Comparison

An important feature of all gene network reconstruction algorithms is the computational complexity incurred in their implementation. The computational complexity of estimating $\boldsymbol{A}$ in (4.28) at a particular iteration is approximately $O(\sum_{n=1}^{N}(M-L_n)^3$

+ $(M - L_n)^2$), where $(M - L_n)$ is the number of non-zero unknowns in the n^{th} column, which is usually very small. We now compare the computational complexity of the four algorithms using the subnetwork data from Yeast and *E. coli*. Average runtime calculated in seconds is summarized for the four subnetworks of each data in Table 4.1. These experiments were performed on a Windows 7 system with a 1.90 GHz Intel Core i7 processor on a Matlab 7.10.0. It is noted that the run time of ROBNCA is comparable to that of FastNCA and is hundreds of times faster than NINCA algorithms for both of its implementations, i.e., involving linear programming and quadratic programming. Moreover, the run time for ROBNCA is far superior to that of the ALS, a direct consequence of the closed form solution derived for estimating the connectivity matrix. It can also be observed that the faster closed form solution for estimating $\boldsymbol{A}$ (4.28) provides additional savings over its predecessor (4.9).

Therefore, it can be inferred from these experiments on synthetic and real data sets that ROBNCA renders superior performance than the contemporary algorithms not only on the yardsticks of accuracy and reliability, but also in terms of computational complexity. The high computational complexity of NINCA far outweighs the benefits it offers in terms of consistency. FastNCA has the smallest run time out of all the algorithms but has poor reliability and is the least robust to the presence of outliers in the data.

4.6 Summary

In this section, we presented ROBNCA, an algorithm for robust network component analysis for estimating the TFAs. The ROBNCA algorithm accounts for the presence of outliers by modeling them as an additional sparse matrix. A closed form solution available at each step of the iterative ROBNCA algorithm ensures faster and reliable performance. The performance of the proposed ROBNCA algorithm

Table 4.1: Average computational time for various methods in seconds.

	S. cerevisiae				*E. coli*			
Subset	1	2	3	4	1	2	3	4
FastNCA	0.2	0.2	0.24	0.2	0.014	0.007	0.007	0.008
ROBNCA 2	0.2	0.2	0.25	0.2	0.016	0.010	0.008	0.008
ROBNCA 1	1.0	0.8	0.99	0.8	0.020	0.018	0.016	0.023
NINCA LP	67	36	56.2	33	0.93	0.76	0.73	0.83
NINCA QP	71	30	125	97	0.59	0.13	0.13	0.13
ALS	Exceeds memory limit				5.3	6.0	7.1	3.5

is compared with NINCA and FastNCA for synthetic as well as real data sets by varying SNR, degrees of correlation and outlier concentration. It is observed that while FastNCA is computationally simpler, yet the TFA recovery is inaccurate and unreliable, a direct consequence of the sensitivity of its decomposition approach to the presence of outliers. The NINCA algorithm offers performance somewhat comparable to the ROBNCA algorithm, however, the ROBNCA algorithm is much more computationally efficient and does not require solving costly optimization problems. Therefore, the cumulative benefits of robustness to the presence of outliers, higher consistency and accuracy compared to the existing state-of-the-art algorithms, and much lower computational complexity make ROBNCA well-suited to the analysis of gene regulatory networks which invariably requires working with large data sets.

5. REVERSE ENGINEERING SPARSE GENE REGULATORY NETWORKS USING CUBATURE KALMAN FILTER AND COMPRESSED SENSING*

5.1 Introduction

Various methods for gene network modeling have been proposed recently in the literature with varying degrees of sophistication [20, 27, 44, 59]. These techniques can be broadly classified as static and dynamic modeling schemes. Static modeling includes the use of correlation and statistical independence for clustering [6, 18, 19], and information theoretic criteria [13, 79, 81]. On the other hand, dynamic models provide an insight into the temporal evolution of gene expressions and hence, yield a more quantitative prediction on gene network behavior [40, 56, 78, 80]. In order to incorporate the stochasticity of gene expressions, statistical techniques have been applied [6]. A rich literature is also available on the Bayesian modeling of gene networks [3, 23, 35, 43, 55, 77]. Promoted in part by the Bayesian methods, the state-space approach is a popular technique to model the gene networks [1, 21, 48, 57, 58, 71, 73], whereby the hidden states can be estimated using the Kalman filter. In case of nonlinear functions, the extended Kalman filter (EKF) and particle filter represent feasible approaches [48, 49, 68]. However, the EKF relies on the first order linear approximations of nonlinearities, while the particle filter may be computationally too complex. A comprehensive review of these methods can be found in [53].

In this work, the gene network is modeled using a state-space approach and the cubature Kalman filter (CKF) is used to estimate the hidden states of the nonlinear model [2, 47]. The gene expressions are assumed to evolve following a sigmoid

*Reprinted from “Reverse Engineering Sparse Gene Regulatory Networks using Cubature Kalman Filter and Compressed Sensing,” Amina Noor, Erchin Serpedin, Mohamed Nounou and Hazem Nounou, Advances in Bioinformatics, 2013, doi:10.1155/2013/205763, 2013. Copyright 2013, Amina Noor et al.

squash function whereas a linear function is considered for the expression data. The noise is assumed to be Gaussian for both the state evolution and gene expression measurements. As the gene network is assumed sparse, any simple mean square error minimization technique will not suffice for the estimation of static parameters. Therefore, a compressed sensing based Kalman filter (CSKF) [29] is used in conjunction with CKF for reliable estimation of parameters. In case of statistical inference, it is essential to obtain some guarantees on the performance of estimators. In this regard, the Cramer-Rao lower bound (CRB) of the parameter estimates is used as a benchmarking index to assess the mean square error (MSE) performance of the proposed estimator which is evaluated here for a parameter vector. The performance of the proposed algorithm is tested on synthetically generated random Boolean networks in various scenarios. The algorithm is also tested using DREAM4 data sets and IRMA networks [9, 54]

5.2 Main Contributions

The main contributions of this section can be summarized as follows.

1. CKF is proposed for the estimation of states and a compressed sensing based Kalman filter is used for the estimation of system parameters. The genes are known to interact with few other genes only necessitating the use of sparsity constraint for more accurate estimation. The proposed algorithm carries out online estimation of parameters and is therefore computationally efficient and is particularly suitable for large gene networks.

2. The Cramer-Rao lower bound is calculated for the estimation of unknown parameters of the system. The performance of the proposed algorithm is compared to CRB. This comparison is significant as it shows room for improvement in the estimation of parameters.

3. The proposed algorithm is compared with the EKF algorithm. Using the false alarm errors, true connections and Hamming distance as fidelity criteria, rigorous simulations are carried out to assess the performance of the algorithm with the increase in the number of samples. In addition, receiver operating characteristic (ROC) curves are plotted to evaluate the algorithms for different network sizes. It is observed that the proposed algorithm outperforms EKF in terms of accuracy and precision. The proposed algorithm is then applied to the DREAM4 10-gene and 100-gene data sets to assess the algorithm accuracy. The underlying gene network for the IRMA data sets is also inferred.

5.3 System Model

Gene regulatory networks can be modeled as static or dynamical systems. In this work, state-space modeling is considered which is an instance of a dynamic modeling approach, and can effectively cope with time variations. The states represent gene expressions and their evolution in time, in general, can be expressed as

$$\boldsymbol{x}_k = g(\boldsymbol{x}_{k-1}) + \boldsymbol{w}_k \qquad k = 1, ..., K, \tag{5.1}$$

where K is the total number of data points available, $\boldsymbol{w}_k$ is assumed to be a zero-mean Gaussian random variable with covariance $\boldsymbol{Q}_k = \sigma_w^2 \boldsymbol{I}$, and the function $g(.)$ represents the regulatory relationship between the genes and is generally non-linear. The microarray data is a set of noisy observations and is commonly expressed as a linear Gaussian model [22]

$$\boldsymbol{y}_k = h(\boldsymbol{x}_k) + \boldsymbol{v}_k, \tag{5.2}$$

where $\boldsymbol{v}_k$ is Gaussian distributed random variable with zero mean and covariance $\boldsymbol{S}_k = \sigma_v^2 \boldsymbol{I}$ and incorporates the uncertainty in the microarray experiments. In order to capture the gene interactions effectively, the following non-linear state evolution model is assumed [48, 68]

$$x_{k,n} = \sum_{m=1}^{N} b_{nm} f(x_{k-1,m}) + w_{k,n}$$

$$k = 1, ..., K, \quad n = 1, ..., N, \tag{5.3}$$

where N is the total number of genes in the network and $f(.)$ is the sigmoid squash function

$$f(x_{k-1,m}) = \frac{1}{1 + e^{-x_{k-1,m}}}. \tag{5.4}$$

This particular choice for the non-linear function ensures that the conditional distribution of the states remains Gaussian [22]. The multiplicative constants b_{nm} quantify the positive or negative relations between various genes in the network. A positive value of b_{nm} implies that the m^{th} gene is activating the n^{th} gene, whereas a negative value implies repression [68, 69]. The absolute value of these parameters indicates the strength of interaction.

The model given in (5.3) and (5.4) in the absence of any constraints may be unidentifiable and result into over-fitted solutions [72]. Assumptions on network structures are, therefore, necessary to obtain a connectivity matrix that agrees with the biological knowledge. In a gene regulatory network (GRN), the genes are known to interact with few other genes only. To this end, the coefficients b_{nm}s are estimated using sparsity constraints, as explained in the next section.

A discrete linear Gaussian model for the microarray data is considered which can be expressed at the k^{th} time instant as [22]

$$\boldsymbol{y}_k = \boldsymbol{x}_k + \boldsymbol{v}_k. \tag{5.5}$$

Stacking the unknown parameters together, the parameter vector to be estimated is

$$\boldsymbol{b} \triangleq [\phi_1, \phi_2, \ldots, \phi_N], \tag{5.6}$$

where $\phi_n = [b_{n1}, \ldots, b_{nN}]$. Plugging the values of states from (5.3) into (5.5), it follows that

$$\boldsymbol{y}_k = \boldsymbol{R}_k \boldsymbol{b} + \boldsymbol{e}_k. \tag{5.7}$$

where

$$\boldsymbol{R}_k \triangleq \begin{bmatrix} \tilde{\boldsymbol{f}}_k & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ 0 & \tilde{\boldsymbol{f}}_k & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \ddots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \tilde{\boldsymbol{f}}_k \end{bmatrix} \tag{5.8}$$

and

$$\tilde{\boldsymbol{f}}_k \triangleq [f(x_{k-1,1}) \ldots f(x_{k-1,N})].$$

Thus, the gene network inference problem boils down to the estimation of system parameters $\boldsymbol{b}$ using the observations $\boldsymbol{y}_k$ where the effective noise $\boldsymbol{e}_k$ is the sum of system and observation noises. The next section describes the proposed inference algorithm for sparse networks.

5.4 Method

In this section, the methodology proposed to infer the system parameters in (5.3) is described. The proposed cubature Kalman filter with sparsity constraints (CKFS) approach is succinctly illustrated in Fig. 5.1. The specific details of this algorithm are as next presented.

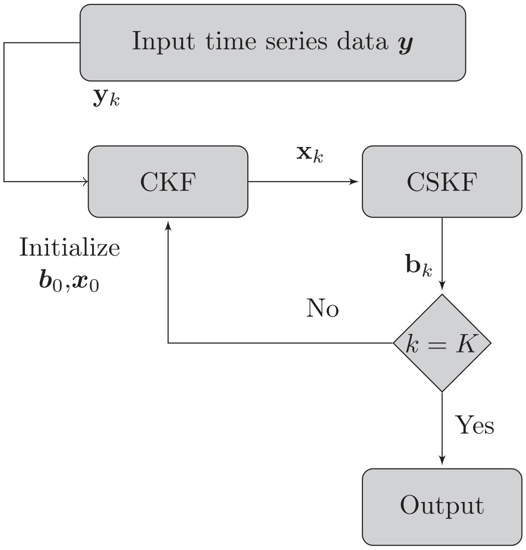


Figure 5.1: Block diagram of network inference methodology CKFS

5.4.1 Cubature Kalman Filter

Kalman filter is a Bayesian filter which provides the optimal solution to a general linear state space inference problem depicted by (5.1) and (5.2), and assumes a recursive *predictive-update* process. The underlying assumption of Gaussianity for the predictive and the likelihood densities simplifies the Kalman filter algorithm to a two step process, consisting of prediction and update of the mean and covariance of the hidden states. However, the presence of nonlinear functions in the state and measurement equations requires calculation of multidimensional integrals of the form *non-linear function* × *Gaussian density* [2], which in general is computationally prohibitive. Several solutions to this problem have been proposed including the EKF, which linearizes the non-linear function by taking its first order Taylor approximation, and the unscented Kalman filter (UKF), which approximates the probability density function (PDF) using a non-linear transformation of the random variable. Recently, a new approach, CKF, has been proposed which evaluates the integrals numerically using spherical-radial cubature rules [2].

The next two subsections briefly explain the working of Bayesian filtering and the CKF solution for the non-linear multidimensional integrals.

5.4.1.1 Time Update

Let the observations up to the time instant k be denoted by $\boldsymbol{d}_k$, i.e., $\boldsymbol{d}_k \triangleq [\boldsymbol{y}_1^T, \ldots, \boldsymbol{y}_k^T]^T$. In the prediction phase, also called the time update of the Bayesian filter, the mean and covariance of the Gaussian posterior density are computed as follows

$$\begin{aligned} \hat{\boldsymbol{x}}_{k|k-1} &= E\left[\boldsymbol{f}(\boldsymbol{x}_{k-1})|\boldsymbol{d}_{k-1}\right] \\ \boldsymbol{P}_{xx,k|k-1} &= E\left[\boldsymbol{f}(\boldsymbol{x}_k)\boldsymbol{f}^T(x_k)\right] - \hat{\boldsymbol{x}}_{k|k-1}\hat{\boldsymbol{x}}_{k|k-1}^T + \boldsymbol{Q}_{k-1}, \end{aligned} \tag{5.9}$$

where E denotes the expectation operator and $\boldsymbol{x}_{k-1}$ is normally distributed with parameters $(\hat{\boldsymbol{x}}_{k-1|k-1}, \boldsymbol{P}_{xx,k-1|k-1})$. The third equality is a consequence of the zero mean nature of Gaussian noise $\boldsymbol{w}$ and its independence from $\boldsymbol{d}_k$. The estimates $\hat{\boldsymbol{x}}_{k-1|k-1}$ and $\boldsymbol{P}_{xx,k-1|k-1}$ are assumed to be available from the previous iteration. Here, $\boldsymbol{P}_{xx,k|k-1}$ is an estimate of the error covariance matrix.

5.4.1.2 Measurement Update

Since the measurement noise is also Gaussian, the likelihood density is given by $\boldsymbol{y}_{k-1}|\boldsymbol{d}_{k-1} \sim \mathcal{N}(\boldsymbol{z}_{k-1}; \hat{\boldsymbol{y}}_{k|k-1}, \boldsymbol{P}_{xx,k|k-1})$. As the measurements become available at the k^{th} time instant, the mean and covariance of the likelihood density are calculated as follows:

$$\begin{aligned} \hat{\boldsymbol{y}}_{k|k-1} &= E\left[\boldsymbol{y}_k|\boldsymbol{d}_{k-1}\right] \\ \boldsymbol{P}_{yy,k|k-1} &= E\left[\boldsymbol{x}_k\boldsymbol{x}_k^T)\right] - \hat{\boldsymbol{y}}_{k|k-1}\hat{\boldsymbol{y}}_{k|k-1}^T + \boldsymbol{S}_{k-1}. \end{aligned} \tag{5.10}$$

The updated posterior density, obtained from the conditional joint density of states and the measurements can be expressed as

$$([\boldsymbol{x}_k^T \boldsymbol{y}_k^T]^T | \boldsymbol{d_{k-1}}) \sim \mathcal{N}\left(\begin{pmatrix} \boldsymbol{x}_{k|\hat{k}-1} \\ \boldsymbol{y}_{k|\hat{k}-1} \end{pmatrix}, \begin{pmatrix} \boldsymbol{P}_{xx,k|k-1} & \boldsymbol{P}_{xy,k|k-1} \\ \boldsymbol{P}_{xy,k|k-1}^T & \boldsymbol{P}_{yy,k|k-1} \end{pmatrix}\right)$$

where

$$P_{xy,k|k-1} = E\left[\boldsymbol{x}_k \boldsymbol{x}_k^T\right] - \hat{\boldsymbol{x}}_{k|k-1} \hat{\boldsymbol{y}}_{k|k-1}^T$$

is the cross-covariance matrix between the states and the measurements. Hence, the states and the corresponding error covariance matrix are updated by calculating the innovation $\boldsymbol{z}_k - \hat{\boldsymbol{z}}_{k|k-1}$ and the Kalman gain $\boldsymbol{K}_{G,i}$:

$$\begin{aligned} \hat{\boldsymbol{x}}_{k|k} &= \hat{\boldsymbol{x}}_{k|k-1} + \boldsymbol{K}_{G,k}(\boldsymbol{y}_k - \hat{\boldsymbol{y}}_{k|k-1}) \\ P_{xx,k|k} &= P_{xx,k|k-1} - \boldsymbol{K}_{G,k} P_{yy,k|k-1} \boldsymbol{K}_{G,k}^T \\ \boldsymbol{K}_{G,k} &= P_{xy,k|k-1} P_{yy,k|k-1}^{-1}. \end{aligned} \tag{5.11}$$

The next subsection briefly describes the computation of high-dimensional integrals present in the equations above.

5.4.1.3 Computation of Integrals Using Spherical-Radial Cubature Points

In order to determine the expectations in (5.9) using a numerical integration method, a spherical-radial cubature rule is applied. This method calculates the cubature points, $\boldsymbol{X}_{j,k-1|k-1}$ as follows [2]

$$\boldsymbol{X}_{j,k-1|k-1} = \boldsymbol{U}_{k-1|k-1} \zeta_j + \hat{\boldsymbol{x}}_{k-1|k-1},$$

where $\zeta_j = \sqrt{\frac{\ell}{2}}[1]_j$, $j = 1, ..., \ell$, $\ell = 2N$ denotes the total number of cubature points, and $\boldsymbol{U}_{k-1|k-1}$ stands for the square-root of the error covariance matrix, i.e.,

$$\boldsymbol{P}_{xx,k-1|k-1} = \boldsymbol{U}_{k-1|k-1}\boldsymbol{U}_{k-1|k-1}^T.$$

The cubature points are updated via the state equation

$$\boldsymbol{X}^*_{j,k|k-1} = g(\boldsymbol{X}_{j,k-1|k-1}). \tag{5.12}$$

The propagated cubature points yield the state and error covariance estimates:

$$\begin{aligned} \hat{\boldsymbol{x}}_{k|k-1} &= \frac{1}{\ell}\sum_{j=1}^{\ell}\boldsymbol{X}^*_{j,k|k-1} \\ \boldsymbol{P}_{xx,k|k-1} &= \frac{1}{\ell}\sum_{j=1}^{\ell}\boldsymbol{X}^*_{j,k|k-1}\boldsymbol{X}^{*T}_{j,k|k-1} - \hat{\boldsymbol{x}}_{k|k-1}\hat{\boldsymbol{x}}^T_{k|k-1} + \boldsymbol{Q}_{k-1}. \end{aligned} \tag{5.13}$$

The integrals in (5.10), (5.11) can be evaluated in a similar manner. The next subsection explains the estimation of parameters in the system.

5.4.2 Estimation of Sparse Parameters Using Kalman Filter

The state estimates are obtained using the CKF as described in the previous subsection. In order to estimate the unknown parameters in the system model, one of the most commonly used methods involves stacking the parameters with the states and estimating them together. The estimation process performed in this manner is called *joint estimation*. Another method for the estimation of parameters consists of a two step recursive process which is termed *dual estimation*. This process estimates the states in the first step and with the assumption that states are known, parameters are estimated in the second step. These steps are repeated until the algorithm converges to the true values or until the amount of available observations is exhausted. Here we make use of the latter technique.

The vector $\boldsymbol{b}$ as defined in (5.6) is assumed to be evolving as a Gauss-Markov model. As discussed previously, the states are assumed known at this step. The system evolution equations can therefore, be expressed as

$$\begin{aligned}\boldsymbol{b}_k &= \boldsymbol{b}_{k-1} + \boldsymbol{\eta}_{k-1} \\ \boldsymbol{y}_k &= \boldsymbol{R}_k \boldsymbol{b}_k + \boldsymbol{e}_k,\end{aligned} \tag{5.14}$$

where $\boldsymbol{\eta}_k$ stands for the i.i.d Gaussian noise and $\boldsymbol{R}_k$ is as defined in (5.8). It is observed that (5.14) is a system of linear equations with additive Gaussian noise, and therefore, the Kalman filter is the optimal choice for the estimation of parameter vector. The standard *predict* and *update* steps involved in Kalman filter are summarized as follows:

$$\begin{aligned}\hat{\boldsymbol{b}}_{k|k-1} &= \hat{\boldsymbol{b}}_{k-1|k-1} + \boldsymbol{\eta}_k \\ \boldsymbol{P}_{bb,k|k-1} &= \boldsymbol{P}_{bb,k-1|k-1} + \boldsymbol{\Sigma}_{\eta_k} \\ \boldsymbol{u}_k &= \boldsymbol{y}_k - \boldsymbol{R}_{f_k} \hat{\boldsymbol{b}}_k \\ \boldsymbol{K}_G &= \boldsymbol{P}_{bb,k|k-1} \boldsymbol{R}_{f_k}^T (\boldsymbol{R}_{f_k} \boldsymbol{P}_{bb,k|k-1} \boldsymbol{R}_{f_k}^T + \sigma_e^2 \boldsymbol{I}^{-1} \\ \hat{\boldsymbol{b}}_{k|k} &= \hat{\boldsymbol{b}}_{k|k-1} + \boldsymbol{K}_G \boldsymbol{u}_k \\ \boldsymbol{P}_{bb,k|k} &= (\boldsymbol{I} - \boldsymbol{K}_G \boldsymbol{R}_{f_k}) \boldsymbol{P}_{bb,k|k-1},\end{aligned} \tag{5.15}$$

where $\boldsymbol{K}_G$ denotes the Kalman gain and $\boldsymbol{P}$ represents the error covariance matrix.

The Kalman filter algorithm is based on a l_2-norm minimization criterion. As the gene networks are known to be highly sparse, the parameter vector is expected to have only a few non-zero values. A more accurate approach for estimating such a vector would be to introduce an additional constraint on its l_1-norm which is the core idea in compressed sensing [29, 61]. Such an l_1-norm constraints provides a unique solution to the under-determined set of equations [8]. Therefore, instead of a simple l_2 norm minimization, the following constrained optimization problem is considered:

$$\min_{\hat{\boldsymbol{b}}_k} ||\hat{\boldsymbol{b}}_k - \boldsymbol{b}_k||_2^2 \text{ s.t. } ||\hat{\boldsymbol{b}}_k|| \leq \epsilon. \tag{5.16}$$

The importance of this constraint can be judged by the fact that without it, the system would be rendered unidentifiable [72].

The problem (5.16) can be solved using a pseudo-measurement (PM) method which incorporates the inequality constraint (5.16) in the filtering process by assuming an artificial measurement $||\boldsymbol{b}_k||_1 - \epsilon = 0$. This is concisely expressed as

$$0 = \bar{\boldsymbol{R}}\hat{\boldsymbol{b}}_k - \epsilon, \ \bar{\boldsymbol{R}}_\tau = [\text{sign}(\hat{\boldsymbol{b}}_\tau(1)), \ldots, \text{sign}(\hat{\boldsymbol{b}}_\tau(N))].$$

The value of the covariance matrix $\boldsymbol{\Sigma_\epsilon} = \sigma_\epsilon^2 \boldsymbol{I}$ of the pseudo-noise ϵ is selected in a similar manner as the process noise covariance in the EKF algorithm. However, it is found that large values of variances, i.e., $\sigma_\epsilon^2 \geq 100$ prove sufficient in most cases [29]. Further details on selecting these parameters can be found in [16, 29]. The PM method solves (5.16) in a recursive manner for K_τ iterations using the following steps:

$$\begin{aligned}
\boldsymbol{K}_G^\tau &= \boldsymbol{P}_\tau \bar{\boldsymbol{R}}_\tau^T (\bar{\boldsymbol{R}}_\tau \boldsymbol{P}_\tau \bar{\boldsymbol{R}}_\tau^T + \boldsymbol{\Sigma_\epsilon})^{-1} \\
\hat{\boldsymbol{b}}_{\tau+1} &= (\boldsymbol{I} - \boldsymbol{K}_G^\tau \bar{\boldsymbol{R}}_\tau)\hat{\boldsymbol{b}}_\tau \\
\boldsymbol{P}_{\tau+1} &= (\boldsymbol{I} - \boldsymbol{K}_G^\tau \bar{\boldsymbol{R}}_\tau)\boldsymbol{P}_\tau.
\end{aligned} \tag{5.17}$$

At each k^{th} time instant, $\boldsymbol{P}_{bb,k|k}$ and $\hat{\boldsymbol{b}}_{k|k}$ obtained from (5.15) are considered as initial values i.e., $\hat{\boldsymbol{b}}^1 = \hat{\boldsymbol{b}}_{k|k}$ and $\boldsymbol{P}_1 = \boldsymbol{P}_{bb,k|k}$ which is the error covariance matrix. The value of K_τ is equal to the number of constraints i.e. the expected number of non-zero $\boldsymbol{b}_{mn}$s in the system. Possible ways for calculating K_τ include minimum description length (MDL) principle and Bayesian information criterion (BIC).

5.4.3 Inference Algorithm

The network inference algorithm is summarized in Algorithm 3. The algorithm consists of a recursive process which repeats itself for the number of observations present in the time series data. For each time sample, the state estimate is obtained using the CKF and the parameter estimate is obtained using the KF. Since the parameters are expected to be sparse, the estimates are then refined further using the CSKF algorithm. This iterative process results in a simple and accurate algorithm for gene network inference while considering a complex non-linear model.

Algorithm 3 `Network Inference - CKFS`

```
Input time series data set y.
Initialize $I, K, \phi_0, \boldsymbol{x}_0$.
for $k = 1, ..., K$ do
    Find the state estimates using CKF following the time and measurement update steps in Section 3.
    Estimate parameters $\hat{\boldsymbol{b}}_k$ from $\boldsymbol{x}_k$ and $\boldsymbol{y}_k$ using (5.15).
    for $\tau = 1, ..., K_\tau$ do
        Update the parameters $\hat{\boldsymbol{b}}_k$ using (5.17).
    end for
end for
return
```

5.5 Cramér-Rao Bound

The performance of an estimator can be judged by comparing it with theoretical lower bounds proposed in parameter estimation theory. The CRB establishes a lower bound on the MSE of an unbiased estimator [30]. In particular, the CRB states that the covariance matrix of the estimator $\hat{\boldsymbol{b}}$ is lower bounded by

$$\mathbb{E}\left[\left(\hat{\boldsymbol{b}}-\boldsymbol{b}\right)\left(\hat{\boldsymbol{b}}-\boldsymbol{b}\right)^T\right] \succeq \left[\boldsymbol{I}\left(\boldsymbol{b}\right)\right]^{-1}, \tag{5.18}$$

where the matrix inequality $\succeq$ is to be interpreted in the semi-definite sense and $\boldsymbol{I}\left(\boldsymbol{b}\right)$ is the Fisher information matrix (FIM):

$$\boldsymbol{I}\left(\boldsymbol{b}\right) = \mathbb{E}\left[\left(\frac{\partial \ln f\left(\boldsymbol{y}|\boldsymbol{b}\right)}{\partial \boldsymbol{b}}\right)\left(\frac{\partial \ln f\left(\boldsymbol{y}|\boldsymbol{b}\right)}{\partial \boldsymbol{b}}\right)^T\right]. \tag{5.19}$$

The CRB for gene network inference can be calculated as follows. By stacking all the observations for $k = 1, \ldots, K$, (5.7) can be written compactly in the matrix form

$$\boldsymbol{y} = \boldsymbol{R}\boldsymbol{b} + \boldsymbol{e}, \tag{5.20}$$

where $\boldsymbol{y} = \left[\boldsymbol{y}_1^T, \ldots, \boldsymbol{y}_K^T\right]^T$, $\boldsymbol{R} = \left[\boldsymbol{R}_1^T, \ldots, \boldsymbol{R}_K^T\right]^T$ and $\boldsymbol{e} = \left[\boldsymbol{e}_1^T, \ldots, \boldsymbol{e}_K^T\right]^T$. The PDF $p\left(\boldsymbol{y}|\boldsymbol{b}\right)$ is expressed as

$$p\left(\boldsymbol{y}|\boldsymbol{b}\right) = C \exp\left(-\frac{\left(\boldsymbol{y}-\boldsymbol{R}\boldsymbol{b}\right)^T\left(\boldsymbol{y}-\boldsymbol{R}\boldsymbol{b}\right)}{2\sigma_e^2}\right), \tag{5.21}$$

where C is a constant. The derivative of $\ln p\left(\boldsymbol{y}|\boldsymbol{b}\right)$ can be expressed as

$$\begin{aligned}\frac{\partial \ln p\left(\boldsymbol{y}|\boldsymbol{b}\right)}{\partial \boldsymbol{b}} &= -\frac{\partial}{\partial \boldsymbol{b}}\left[\frac{\left(\boldsymbol{y}-\boldsymbol{R}\boldsymbol{b}\right)^T\left(\boldsymbol{y}-\boldsymbol{R}\boldsymbol{b}\right)}{\sigma_e^2}\right] \\ &= \frac{\boldsymbol{R}^T\boldsymbol{y}-\boldsymbol{R}^T\boldsymbol{R}\boldsymbol{b}}{\sigma_e^2}.\end{aligned}$$

It now follows that

$$\begin{aligned}\left(\frac{\partial \ln p\left(\boldsymbol{y}|\boldsymbol{b}\right)}{\partial \boldsymbol{b}}\right)\left(\frac{\partial \ln p\left(\boldsymbol{y}|\boldsymbol{b}\right)}{\partial \boldsymbol{b}}\right)^T &= \\ &\frac{\boldsymbol{R}^T\left(\boldsymbol{y}-\boldsymbol{R}\boldsymbol{b}\right)\left(\boldsymbol{y}-\boldsymbol{R}\boldsymbol{b}\right)^T\boldsymbol{R}}{\sigma_e^4}\,.\end{aligned} \tag{5.22}$$

By taking the expectation of (5.22), the FIM in (5.19) is given by

$$\boldsymbol{I}\left(\boldsymbol{b}\right) = \frac{\boldsymbol{R}^T\boldsymbol{R}}{\sigma_e^2}. \tag{5.23}$$

The inverse of the FIM in (5.23) can be used to place a lower bound on the estimation error of the parameter vector $\boldsymbol{b}$. Fig. 5.2 shows the comparison of MSE of CKFS algorithm with CRB as a function of number of samples K for one representative gene from the eight-gene network considered in Section 5.1. It is observed that the MSE of the estimated parameters decreases with increasing number of samples.

5.6 Results and Discussion

The simulation results of the CKFS algorithm are discussed in this section. The performance is first tested on synthetic data obtained from randomly generated Boolean networks under various scenarios and performance metrics. The algorithm is then assessed on the DREAM4 networks and the IRMA network.

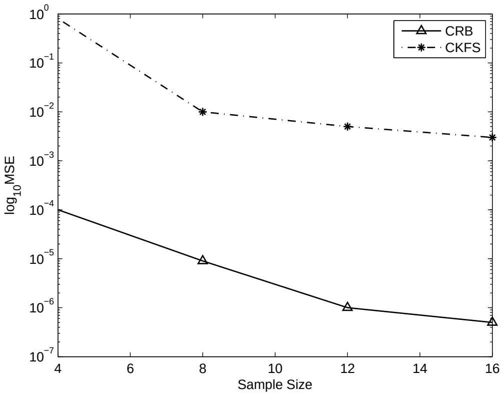


Figure 5.2: Cramer Rao bound on the estimation of parameters. The MSE for one of the representative θ is shown here for a network consisting of 8 vertices.

5.6.1 Synthetic Data

Time series data is produced from randomly generated Boolean networks using the system model (5.3) and (5.5). Two scenarios are considered for this purpose.

First, the comparison is performed by varying the number of sample size while keeping the network size fixed. The gene network consists of 8 genes and 20 vertices. In terms of network estimation, if the algorithm predicts an edge between two nodes which may not be present in reality, an error, referred to as *false alarm error* (F), is said to have occurred. Another situation is the indication of the absence of a vertex in the graph which in fact is present in the real network. This kind of error is termed *missed detection* (M). The summation of these two errors normalized over the total number of vertices in the network yields the *Hamming distance.* It is also important to consider the probability of predicting the true connections correctly which will be assessed by the *true connections* (T) metric. An algorithm with low Hamming distance and small false alarm error is particularly desirable as predicting an edge erroneously can be troublesome for biologists. True connections indicate the

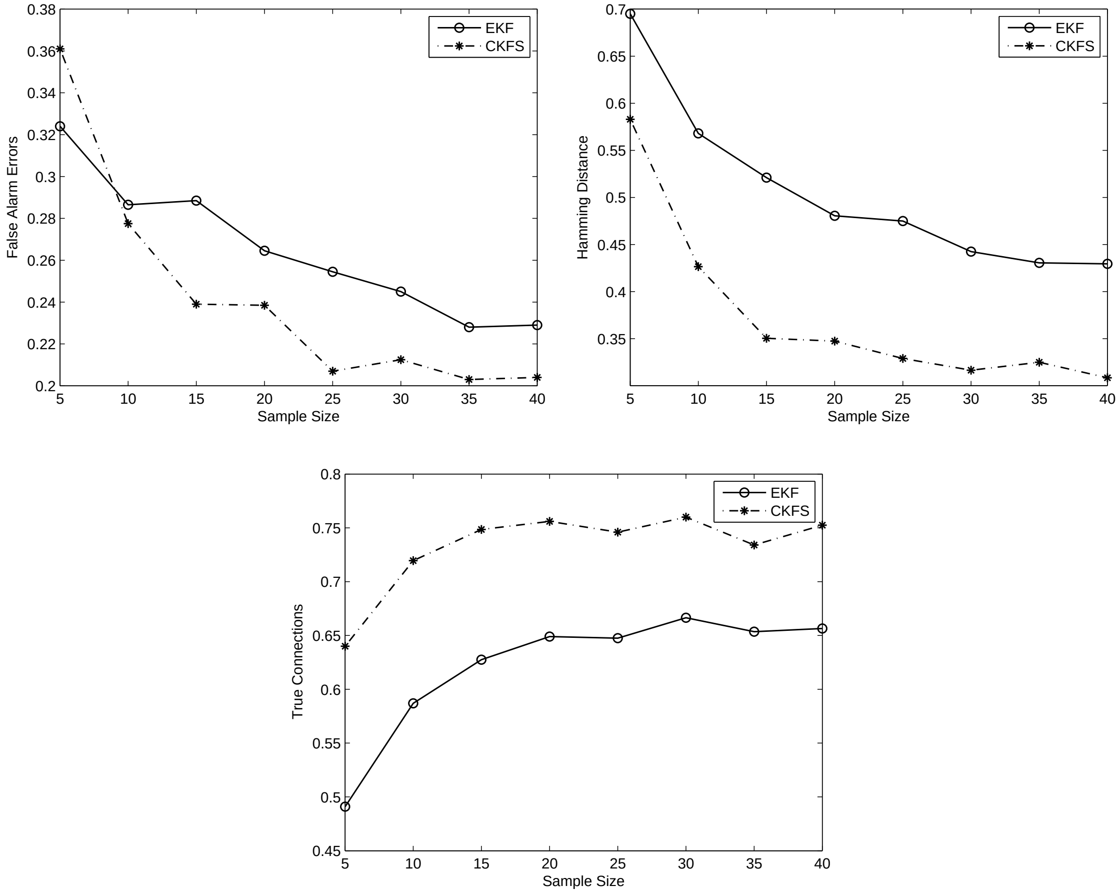


Figure 5.3: Left to right: False alarm errors, Hamming distance and true connections. The synthetic networks consist of 8 vertices and 20 edges. The metric is normalized over the number of edges. CKFS gives lower error and predicts more true connections with the increase in the sample size of data.

reliability of the predictions. Figure. 5.3 illustrate the performance of the CKFS algorithm and that of the EKF algorithm proposed in [68] in terms of the metrics described above. It is important to mention here that the same system model is assumed by both CKFS and EKF algorithms for the purpose of this simulation. These metrics are the same as those used in [79]. The variances of both the system and measurement noises, σ_w^2 and σ_v^2, respectively are taken to be 10^{-5} in all the

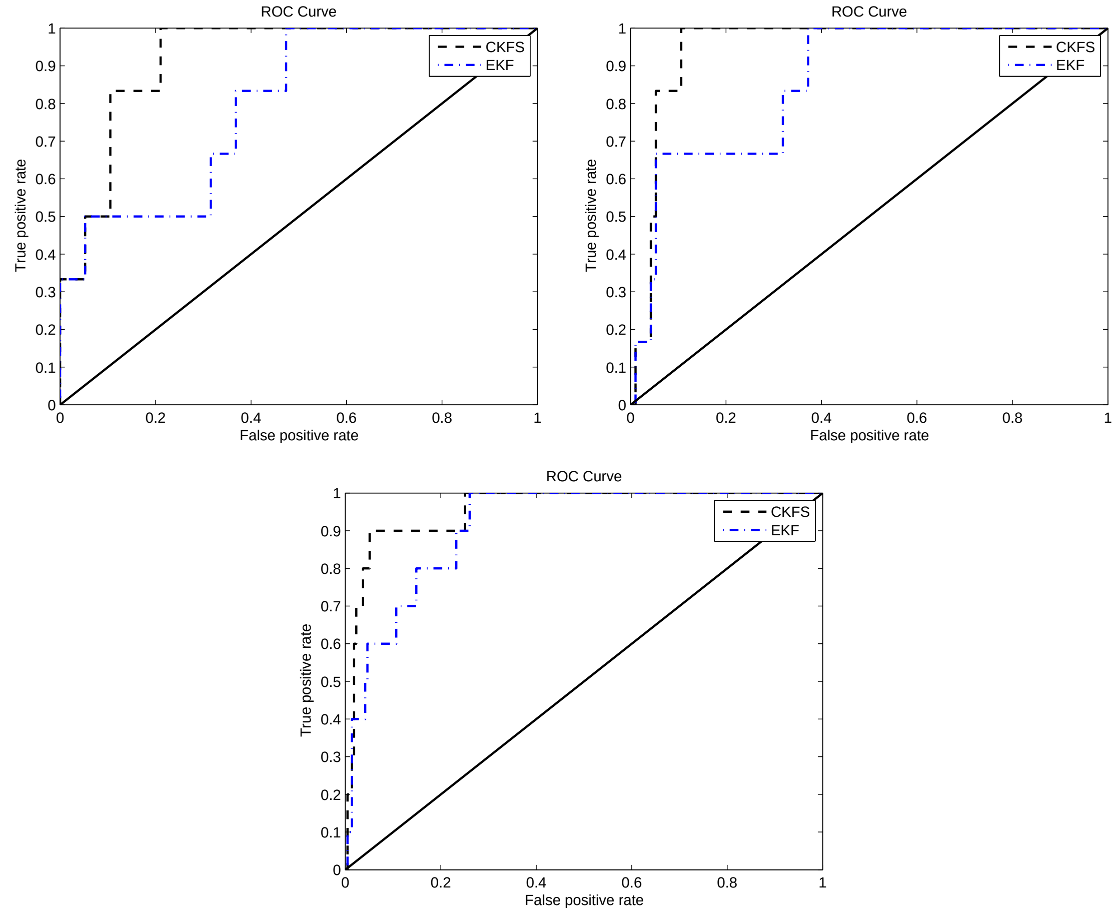


Figure 5.4: ROC curves for the performance of CKFS and EKF using synthetic data. (N,E,K) Left to right: (5,10,20), (10,12,20),(15,19,20). The area under the ROC curve for CKFS is more than that for EKF for various sized networks.

simulations and are assumed to be known. It is noticed that EKF has a slightly lower false alarm rate when the number of samples is small, however, as the number of samples increases, CKFS yields a lower false alarm error. The Hamming distance for CKFS is also smaller than EKF indicating lesser cumulative error. True connections show a consistent behavior for the two algorithms when the number of samples is increased where CKFS is able to predict connections more accurately. These experiments show the superiority of CKFS in terms of lower error rate.

To obtain a more rigorous evaluation, the performance of algorithms is then compared in a scenario which considers the sample size to be fixed and assumes networks of different sizes. The receiver operating characteristic (ROC) curves are plotted as performance measures. A higher area under the ROC curve (AUROC) shows more true-positives for a given false-positive, and therefore, indicates better classification. The performance of CKFS(N, E, K) and EKF$(N, E, K)$is shown in Fig. 5.4, where N stands for the number of nodes, E represents the number of edges and K denotes the time points. It is observed that the CKFS exhibits superior performance than the EKF for networks of different sizes.

The complexity of the two algorithms is compared for synthetically generated networks with number of genes equal to 10, 20, 30 and 40. The sample size is kept to 50 time points for each of these networks and the run time for EKF and CKFS algorithms is calculated as shown in Table 5.1. It is noted that EKF is faster for smaller network sizes but as the network size increases, the run time gets much larger than that for CKFS. The main reason for this is that EKF [68] estimates the states and parameters by stacking them together which requires large sized matrix multiplications at each iteration. The benefit associated with performing dual estimation, as in CKFS, is that the parameters are estimated separately from the states. Since the system is linear and one-to-one for parameters, inversion of much smaller matrices can be performed reducing the computational complexity of CKFS algorithm. CKFS, is therefore, particularly attractive for large sized networks.

5.6.2 DREAM4 Gene Networks

Several *in silico* networks have been produced in order to benchmark the performance of gene network inference algorithms. DREAM (Dialogue on Reverse Engineering Assessment and Methods) *in silico* networks serve as one of the popular

Table 5.1: Run time in seconds for EKF and CKFS algorithms for varying network sizes for synthetically generated data. The number of sample points is fixed to 50.

Number of Genes	10	20	30	40
EKF	0.16	1.9	16.5	84
CKFS	1.2	4.3	11.5	24.1

Table 5.2: Area under the ROC curve (AUROC) and area under the PR curve (AUPR) for DREAM4 10-gene networks for the five different networks.

Algorithm	network 1	network 2	network 3	network 4	network 5
ODE [54]	0.62 (0.27)	0.63 (0.32)	0.58 (0.21)	0.63 (0.23)	0.68 (0.25)
CKFS	0.63 (0.40)	0.67 (0.50)	0.72 (0.50)	0.75 (0.49)	0.81 (0.42)
random [54]	0.55 (0.18)	0.55 (0.19)	0.55 (0.17)	0.57 (0.17)	0.56 (0.16)

Table 5.3: Area under the ROC curve (AUROC) and area under the PR curve (AUPR) for DREAM4 100-gene networks for the five different networks.

Algorithm	network 1	network 2	network 3	network 4	network 5
ODE [54]	0.55 (0.02)	0.55 (0.03)	0.60 (0.03)	0.54 (0.02)	0.59 (0.03)
CKFS	0.67 (0.13)	0.57 (0.08)	0.60 (0.10)	0.62 (0.10)	0.60 (0.07)
random [54]	0.50 (0.002)	0.50 (0.002)	0.50 (0.002)	0.50 (0.002)	0.50 (0.002)

methods used for this purpose [54]. In this section, the performance of the CKFS algorithm is evaluated using the 10-gene and 100-gene networks released online by the DREAM4 challenge. Five networks are produced using the known GRNs of *Escherichia coli* and *Saccharomyces cerevisiae*. The data sets for each of 10-gene network consists of 21 data points for five different perturbations. The inference is performed by using all the perturbations. The 100-gene network consists of data sets for ten perturbations. AUROC and area under the precision-recall curve (AUPR) are calculated for the five networks of both the data sets and shown in Table 5.2 and Table 5.3, respectively. The quantities: *precision* and *recall* are defined as $P = T/(T + F)$ and $R = T/(T + M)$, respectively. For comparison purposes, the values of the two quantities for time-series network identification (TSNI) algorithm that exploits ordinary differential equations are also given [54]. The CKFS algorithm is found to perform significantly better than the TSNI algorithm.

5.6.3 IRMA Gene Network

In addition to synthetic data, it is imperative to test the algorithms using real biological data. In this sub-section, the performance of the CKFS algorithm is assessed using the *in vivo* reverse-engineering and modeling assessment (IRMA) network [9]. This network consists of five genes. Galactose activates the gene expression in the network whereas glucose deactivates it. The cells are grown in the presence of galactose and then switched to glucose to obtain the switch-off data which represents the expressive samples at 21 time points. The switch-on data consisting of 16 sample points and is obtained by growing the cells in a glucose medium and then changing to galactose. The system and measurement noise variances for the CKFS are assumed to be identical as in the previous simulations. Fig. 5.5 shows the inferred network, the gold standard and the network inferred using TSNI. It is observed that

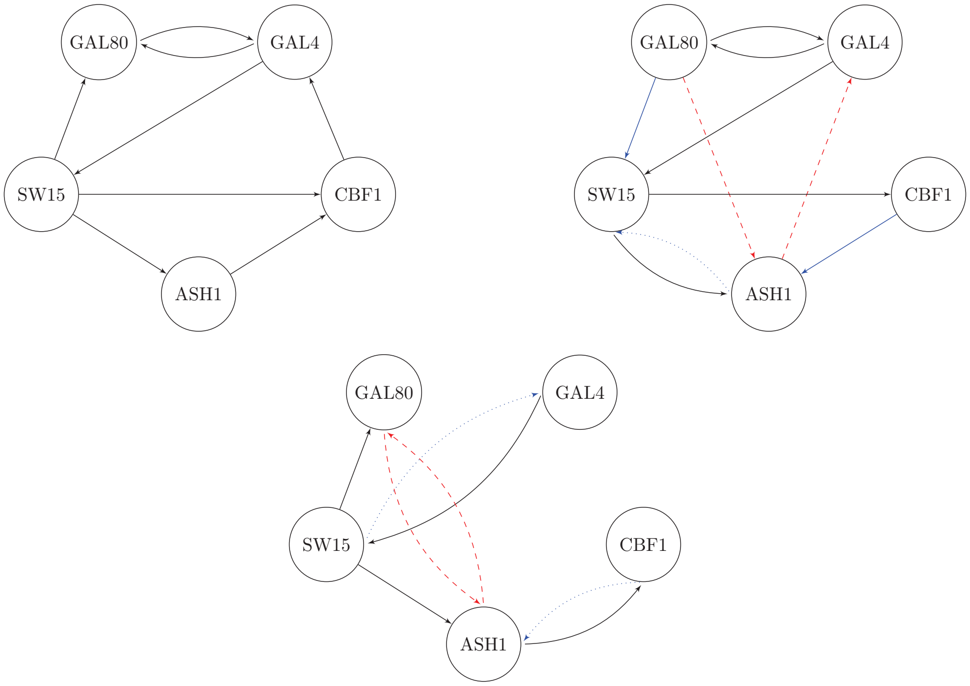


Figure 5.5: The inferred IRMA networks. Left to right: gold standard, inferred network using CKFS, inferred network using ODE [9,54]. Black arrows indicate true connections, blue arrows indicate the edges that are correct but their directions are reversed, and red arrows indicate false positives.

the CKFS algorithm succeeds to predict most of the interactions while giving lower false positives.

5.7 Summary

This section presented a novel algorithm for inferring gene regulatory networks from time series data. Gene regulation is assumed to follow a non-linear state evolution model. The parameters of the system, which indicate the inhibitory or excitatory relationships between the genes, are estimated using compressed sensing based Kalman filtering. The sparsity constraint on the parameters is crucial because the genes are known to interact with few other genes only. The use of CKF and the dual estimation of states and parameters renders the algorithm computationally efficient. The performance of CKFS is evaluated for synthetic data for different network sizes as well as varying sample points. ROC curves, Hamming distance and True positives are used for comparing the accuracy of inferred network with EKF. It is observed that CKFS outperforms the EKF algorithm. In addition, CKFS gives advantages over EKF in terms of smaller run time for large networks. The Cramer-Rao lower bound is also determined for the parameters of the model and compared with the MSE performance of the proposed algorithm. Assessment using DREAM4 10-gene and 100-gene networks and IRMA network data corroborate the superior performance of CKFS. Future research directions include incorporating the estimation of model order in the network inference algorithm.

6. CONCLUSIONS AND FUTURE WORK

Precise and accurate inference of gene regulatory networks is imperative in understanding the roles of complicated biological processes. These networks can aid in understanding which genes cause a particular disease and how its harmful effects can be warded off. Gene expression data available from high-throughput technologies measure the response of genes to various other genes and transcription factors (TFs). In addition, Chip-chip data provide knowledge about TF-gene interactions. This dissertation investigates transcription factor activity estimation using network component analysis (NCA) and the inference of gene regulatory networks.

First, a closed form solution is presented for a NINCA algorithm using convex optimization methods which reduces the computational complexity by tens of times while giving the same estimation accuracy. Next, this dissertation investigated how to overcome the challenge of incomplete prior information about the TF-gene interactions. An iterative reweighted ℓ_2 norm based algorithm was proposed which estimates the connectivity matrix with higher accuracy and lower complexity when the connectivity information could be missing. Another important extension treated in this dissertation was to find computationally efficient algorithms which are robust to the presence of outliers in the gene expression data. An attractive feature of all these algorithms was the derivation of a closed-form solution for the connectivity matrix. Finally, a novel gene regulatory network inference algorithm was presented which makes use of the knowledge that gene networks are known to be sparse.

An interesting scenario that remains open for future investigation is the case of imperfect prior knowledge of the TF-gene interactions, as the biological data is known to have errors. Fast and robust algorithms should be studied which handle the uncertainty in the prior information by employing a Bayesian framework. Using the replicated gene expression data can also provide useful information that can be exploited in the inference problem.

REFERENCES

[1] J. Angus, M.J. Beal, J. Li, C. Rangel, and D.L. Wild. Inferring transcriptional networks using prior biological knowledge and constrained state-space models. In N.D. Lawrence, M. Girolami, M. Rattray, and G. Sanguinetti, editors, *Learning and Inference in Computational Systems Biology*, pages 117–152. MIT Press, Cambridge, MA, 2010.

[2] I. Arasaratnam and S. Haykin. Cubature Kalman filter. *IEEE Transactions on Automatic Control*, 54(6):1254–1269, June 2009.

[3] M. J. Beal, F. Falciani, Z. Ghahramani, C. Rangel, and D. L. Wild. A Bayesian approach to reconstructing genetic regulatory networks with hidden factors. *Bioinformatics*, 21(3):349–356, 2005.

[4] S. Boyd and L. Vandenberghe. *Convex optimization.* Cambridge University Press, Cambridge, UK, 2004.

[5] M. P. Brynildsen, L. M. Tran, and J. C. Liao. A Gibbs sampler for the identification of gene expression and network connectivity consistency. *Bioinformatics*, 22(24):3040–3046, 2006.

[6] X. Cai and G. B. Giannakis. Identifying differentially expressed genes in microarray experiments with model-based variance estimation. *IEEE Transactions on Signal Processing*, 54(6):2418–2426, June 2006.

[7] X. Cai and X. Wang. Stochastic modeling and simulation of gene networks. *IEEE Signal Processing Magazine*, 24(1):27–36, January 2007.

[8] E. J. Candes and T. Tao. Decoding by linear programming. *IEEE Transactions on Information Theory*, 51(12):4203–4215, 2005.

[9] I. Cantone, L. Marucci, F. Iorio, M. A. Ricci, V. Belcastro, M. Bansal, S. Santini, D. di Bernardo, M. di Bernardo, and M. P. Cosma. A yeast synthetic network for in vivo assessment of reverse-engineering and modeling approaches. *Cell*, 137:172–181, 2009.

[10] C. Chang, Z. Ding, Y. S. Hung, and P. C. W. Fung. Fast network component analysis (FastNCA) for gene regulatory network reconstruction from microarray data. *Bioinformatics*, 24(11):1349–1358, 2008.

[11] R. Chartrand and W. Yin. Iteratively reweighted algorithms for compressive sensing. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2008*, pages 3869–3872. IEEE, 2008.

[12] P. Comon. Independent component analysis. In *Proceedings of the International Signal Processing Workshop on Higher-Order Statistics*, pages 111–120, 1991.

[13] J. Dougherty, I. Tabus, and J. Astola. Inference of gene regulatory networks based on a universal minimum description length. *EURASIP Journal on Bioinformatics and Systems Biology*, 2008:5, 2008.

[14] M. Finegold and M. Drton. Robust graphical modeling of gene networks using classical and alternative t-distributions. *The Annals of Applied Statistics*, 5(2A):1057–1080, 2011.

[15] S. J. Galbraith, L. M. Tran, and J. C. Liao. Transcriptome network component analysis with limited microarray data. *Bioinformatics*, 22(15):1886–1894, 2006.

[16] J. D. Geeter, H. V. Brussel, and J. D. Schutter. A smoothly constrained Kalman filter. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(10):1171–1177, October 1997.

[17] G. B. Giannakis, G. Mateos, S. Farahmand, V. Kekatos, and H. Zhu. Uspacor: Universal sparsity-controlling outlier rejection. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2011*, pages 1952–1955. IEEE, 2011.

[18] C. D. Giurcaneanu, I. Tabus, and J. Astola. Clustering time series gene expression data based on sum-of-exponentials fitting. *EURASIP Journal on Advances in Signal Processing*, 2005(8):1159–1173, 2005.

[19] C. D. Giurcaneanu, I. Tabus, J. Astola, J. Ollila, and M. Vihinen. Fast iterative gene clustering based on information theoretic criteria for selecting the cluster structure. *Journal of Computational Biology*, 11(4):660–682, 2004.

[20] H. Hache, H. Lehrach, and R. Herwig. Reverse engineering of gene regulatory networks: a comparative study. *EURASIP Journal on Bioinformatics and Systems Biology*, 2009:8, 2009.

[21] O. Hirose, R. Yoshida, S. Imoto, R. Yamaguchi, T. Higuchi, D. S. Charnock-Jones, S. Miyano, et al. Statistical inference of transcriptional module-based gene networks from time course gene expression profiles by using state space models. *Bioinformatics*, 24(7):932–942, 2008.

[22] Y. Huang, I. Tienda-Luna, and Y. Wang. Reverse engineering gene regulatory networks. *IEEE Signal Processing Magazine*, 26(1):76–97, 2009.

[23] Y. Huang, J. Wang, J. Zhang, M. Sanchez, and Y. Wang. Bayesian inference of genetic regulatory networks from time series microarray data using dynamic Bayesian networks. *Journal of Multimedia*, 2(3):46–56, 2007.

[24] N. Jacklin, Z. Ding, W. Chen, and C. Chang. Noniterative convex optimization methods for network component analysis. *IEEE/ACM Transactions on*

Computational Biology and Bioinformatics, 9(5):1472–1481, 2012.

[25] G. H. Jajamovich, X. Wang, A. P. Arkin, and M. S. Samoilov. Bayesian multiple-instance motif discovery with bambi: inference of recombinase and transcription factor binding sites. *Nucleic Acids Research*, 39(21):e146–e146, 2011.

[26] I. T. Jolliffe. *Principal component analysis*, volume 487. Springer-Verlag New York, 1986.

[27] H. de Jong. Modeling and simulation of genetic regulatoy systems: A literature review. *Journal of Computational Biology*, 9(1):67–103, 2002.

[28] K. C. Kao, Y-L. Yang, R. Boscolo, C. Sabatti, V. Roychowdhury, and J. C. Liao. Transcriptome-based determination of multiple transcription regulator activities in escherichia coli by using network component analysis. *Proceedings of the National Academy of Sciences of the United States of America*, 101(2):641–646, 2004.

[29] A. Karmi, P. Gurfil, and D. Kanevsky. Methods for sparse signal recovery using Kalman filtering with embedded pseudo-measurement norms and quasi-norms. *IEEE Transactions on Signal Processing*, 56(4):2405–2409, April 2010.

[30] S. M. Kay. *Fundamentals of statistical signal processing. Estimation theory.* Prentice-Hall, Upper Saddle River, NJ, 1993.

[31] V. Kekatos and G. B. Giannakis. From sparse signals to sparse residuals for robust sensing. *IEEE Transactions on Signal Processing*, 59(7):3355–3368, July 2011.

[32] H. Kitano. Computational systems biology. *Nature*, 420:206–210, Nov. 2002.

[33] T. I. Lee, N. J. Rinaldi, F. Robert, D. T. Odom, Z.v Bar-Joseph, G. K. Gerber, N. M. Hannett, C. T. Harbison, C. M. Thompson, I. Simon, and et.al. Tran-

scriptional regulatory networks in *Saccharomyces cerevisiae*. *Science Signalling*, 298(5594):799, 2002.

[34] J. C. Liao, R. Boscolo, Y. L. Yang, L. M. Tran, C. Sabatti, and V. P. Roychowdhury. Network component analysis: Reconstruction of regulatory signals in biological systems. *Proceedings of the National Academy of Sciences*, 100(26):15522–15527, 2003.

[35] H. Liu, D. Yue, Y. Chen, S-J. Gao, and Y. Huang. A Bayesian approach for identifying miRNA targets by combining sequence prediction and gene expression profiling. *BMC Genomics*, 11(Suppl 3):S12, 2010.

[36] B. K. Mallick, D. Gold, and V. Baladandayuthapani. *Bayesian analysis of gene expression data*. Wiley, Cornwall, UK, 2009.

[37] A. A. Margolin, I. Nemenman, K. Basso, C. Wiggins, G. Stolovitzky, R. D. Favera, and A. Califano. Aracne: an algorithm for the reconstruction of gene regulatory networks in a mammalian cellular context. *BMC Bioinformatics*, 7(Suppl 1):S7, 2006.

[38] F. Markowetz and R. Spang. Inferring cellular networks–a review. *BMC Bioinformatics*, 8(Suppl 6):S5, 2007.

[39] G. Mateos and G. B. Giannakis. Robust PCA as bilinear decomposition with outlier-sparsity regularization. *IEEE Transactions on Signal Processing*, 60(10):5176–5190, October 2012.

[40] J. Meng, Y. Chen, S-J. Gao, and Y Huang. Robust inference of the context specific structure and temporal dynamics of gene regulatory network. *BMC Genomics*, 11(Suppl 3):S11, 2010.

[41] J. Meng, J. Zhang, Y. Qi, Y. Chen, and Y. Huang. Uncovering transcriptional regulatory networks by sparse Bayesian factor model. *EURASIP Journal on Advances in Signal Processing*, 2010:3, 2010.

[42] A. Mortazavi, B.A. Williams, K. McCue, L. Schaeffer, and B. Wold. Mapping and quantifying mammalian transcriptomes by RNA-seq. *Nature Methods*, 5(7):621–628, 2008.

[43] K. Murphy and S. Mian. Modeling gene expression data using dynamic Bayesian networks. Technical report, University of California, Berkeley, 2001.

[44] I. Nachman, A. Regev, and N. Friedman. Inferring quantitative models of regulatory networks from expression data. *Bioinformatics*, 20(1):248–256, 2004.

[45] A. Noor, A. Ahmad, E. Serpedin, M. Nounou, and H. Nounou. ROBNCA: robust network component analysis for recovering transcription factor activities. *Bioinformatics*, 29(19):2410–2418, 2013.

[46] A. Noor, A. Ahmad, E. Serpedin, M. N. Nounou, and H. N. Nounou. ROBNCA: robust network component analysis for recovering transcription factor activities. In *Proceedings of the IEEE International Workshop on Genomic Signal Processing and Statistics (GENSIPS), 2013*.

[47] A. Noor, E. Serpedin, M. Nounou, and H. Nounou. A cubature Kalman filter approach for inferring gene regulatory networks using time series data. In *Proceedings of the IEEE International Workshop on Genomic Signal Processing and Statistics (GENSIPS), 2011*, pages 25–28. IEEE, 2011.

[48] A. Noor, E. Serpedin, M. Nounou, and H. Nounou. Inferring gene regulatory networks via nonlinear state-space models and exploiting sparsity. *IEEE/ACM*

Transactions on Computational Biology and Bioinformatics (TCBB), 9(4):1203–1211, 2012.

[49] A. Noor, E. Serpedin, M. Nounou, and H. Nounou. Inferring gene regulatory networks with nonlinear models via exploiting sparsity. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2012*, pages 725–728. IEEE, 2012.

[50] A. Noor, E. Serpedin, M. Nounou, and H. Nounou. Reverse engineering sparse gene regulatory networks using cubature Kalman filter and compressed sensing. *Advances in Bioinformatics*, 2013, doi:10.1155/2013/205763, 2013.

[51] A. Noor, E. Serpedin, M. Nounou, and H. Nounou. SparseNCA: Sparse network component analysis for recovering transcription factor activities with incomplete prior information. *submitted to Bioinformatics*, Sept. 2013.

[52] A. Noor, E. Serpedin, M. Nounou, H. Nounou, N. Mohamed, and L. Chouchane. Information theoretic methods for modeling of gene regulatory networks. In *Proceedings of the IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB), 2012*, pages 418–423. IEEE, 2012.

[53] A. Noor, E. Serpedin, M. N. Nounou, H. N. Nounou, N. Mohamed, and L. Chouchane. An overview of the statistical methods used for inferring gene regulatory networks and protein-protein interaction networks. *Advances in Bioinformatics*, 2013, doi:10.1155/2013/953814, 2013.

[54] C. A. Penfold and D. L. Wild. How to infer gene networks from expression profiles, revisited. *Interface Focus*, 1(6):857–870, 2011.

[55] B-E. Perrin, L. Ralaivola, A. Mazurie, S. Bottani, J. Mallet, and F. d'Alche Buc. Gene networks inference using dynamic Bayesian networks. *Bioinformatics*, 19(2):138–148, 2003.

[56] L. Qian, H. Wang, and E. R. Dougherty. Inference of noisy nonlinear differential equation models for gene regulatory networks using genetic programming and Kalman filtering. *IEEE Transactions on Signal Processing*, 56(7-2):3327–3339, July 2008.

[57] M. Quach, N. Brunel, and F. d'Alch Buc. Estimating parameters and hidden variables in non-linear state-space models based on odes for biological networks inference. *Bioinformatics*, 23(23):3209–3216, 2007.

[58] C. Rangel, J. Angus, Z. Ghahramani, M. Lioumi, E. Sotheran, A. Gaiba, D. L. Wild, and F. Falciani. Modeling t-cell activation using gene expression profiling and state-space models. *Bioinformatics*, 20(9):1361–1372, 2004.

[59] T. Schlitt and A. Brazma. Current approaches to gene regulatory network modelling. *BMC Bioinformatics*, 8(Suppl 6):S9, 2007.

[60] P. T. Spellman, G. Sherlock, M. Q. Zhang, V. R. Iyer, K. Anders, M. B. Eisen, P. O. Brown, D. Botstein, and B. Futcher. Comprehensive identification of cell cycle–regulated genes of the yeast *Saccharomyces cerevisiae* by microarray hybridization. *Molecular Biology of the Cell*, 9(12):3273–3297, 1998.

[61] R. Tibshirani. Regression shrinkage and selection via the LASSO. *Journal of the Royal Statistical Society (Series B)*, 58:267–288, 1996.

[62] L. Tran, D. Hyduke, and J. Liao. Trimming of mammalian transcriptional networks using network component analysis. *BMC Bioinformatics*, 11(Suppl 1):S11, 2010.

[63] L. M. Tran, M. P. Brynildsen, K. C. Kao, J. K. Suen, and J. C. Liao. gNCA: a framework for determining transcription factor activity based on transcriptome: identifiability and numerical implementation. *Metabolic Engineering*, 7(2):128–141, 2005.

[64] J. A. Tropp. Just relax: Convex programming methods for identifying sparse signals in noise. *IEEE Transactions on Information Theory*, 52(3):1030–1051, 2006.

[65] P. Tseng. Convergence of a block coordinate descent method for nondifferentiable minimization. *Journal of Optimization Theory and Applications*, 109(3):475–494, 2001.

[66] C. Wang, J. Xuan, L. Chen, P. Zhao, Y. Wang, R. Clarke, and E. Hoffman. Motif-directed network component analysis for regulatory network inference. *BMC Bioinformatics*, 9(Suppl 1):S21, 2008.

[67] C. Wang, J. Xuan, I-M. Shih, R. Clarke, and Y. Wang. Regulatory component analysis: A semi-blind extraction approach to infer gene regulatory networks with imperfect biological knowledge. *Signal Processing*, 92(8):1902–1915, 2012.

[68] Z. Wang, X. Liu, Y. Liu, J. Liang, and V. Vinciotti. An extended Kalman filtering approach to modeling nonlinear dynamic gene regulatory networks via short gene expression time series. *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)*, 6(3):410–419, 2009.

[69] Z. Wang, F. Yang, D. W. C. Ho, S. Swift, A. Tucker, and X. Liu. Stochastic dynamic modeling of short gene expression time-series data. *IEEE Transactions on Nanobioscience*, 7(1):44–55, March 2008.

[70] D. Wipf and S. Nagarajan. Iterative reweighted ℓ_1 and ℓ_2 methods for finding sparse solutions. *IEEE Journal of Selected Topics in Signal Processing*, 4(2):317–329, 2010.

[71] F-X Wu, W-J Zhang, and A. J. Kusalik. Modeling gene expression from microarray expression data with state-space equations. In Russ B. Altman, A. Keith Dunker, Lawrence Hunter, Tiffany A. Jung, and Teri E. Klein, editors, *Pacific Symposium on Biocomputing*, pages 581–592. World Scientific, 2004.

[72] H. Xiong and Y. Choe. Structural systems identification of genetic regulatory networks. *Bioinformatics*, 24(4):553–560, 2008.

[73] R. Yamaguchi, S. Yoshida, S. Imoto, T. Higuchi, and S. Miyano. Finding module-based gene networks with state-space modelling high-dimensional and short time-course gene expression data. *IEEE Signal Processing Magazine*, 24(1):37–46, January 2007.

[74] Y-L. Yang, J. Suen, M. P. Brynildsen, S. J. Galbraith, and J. C. Liao. Inferring yeast cell cycle regulators and interactions using transcription factor activities. *BMC Genomics*, 6:90, 2005.

[75] K.Y. Yip, R.P. Alexander, K.K. Yan, and M. Gerstein. Improved reconstruction of in silico gene regulatory networks by integrating knockout and perturbation data. *PloS one*, 5(1):e8121, 2010.

[76] M. Yuan and Y. Lin. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(1):49–67, 2005.

[77] Y. Zhang, Z. Deng, H. Jiang, and P. Jia. Inferring gene regulatory networks from multiple data sources via a dynamic Bayesian network with structural EM. In

Sarah Cohen Boulakia and Val Tannen, editors, *DILS*, volume 4544 of *Lecture Notes in Computer Science*, pages 204–214. Springer, 2007.

[78] W. Zhao, E. Serpedin, and E. R. Dougherty. Inferring gene regulatory networks from time series data using the minimum description length principle. *Bioinformatics*, 22(17):2129–2135, 2006.

[79] W. Zhao, E. Serpedin, and E. R. Dougherty. Inferring connectivity of genetic regulatory networks using information-theoretic criteria. *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)*, 5(2):262–274, 2008.

[80] X. Zhou, X. Wang, M. Bittner, R. Pal, I. Ivanov, and E. R. Dougherty. A Bayesian connective-based approach to constructing probabilistic gene regulatory networks. *Bioinformatics*, 20(17):2918–2927, 2004.

[81] X. Zhou, X. Wang, and E. R. Dougherty. Gene clustering based on cluster-wide mutual information. *Journal of Computational Biology*, 11(1):151–165, 2004.

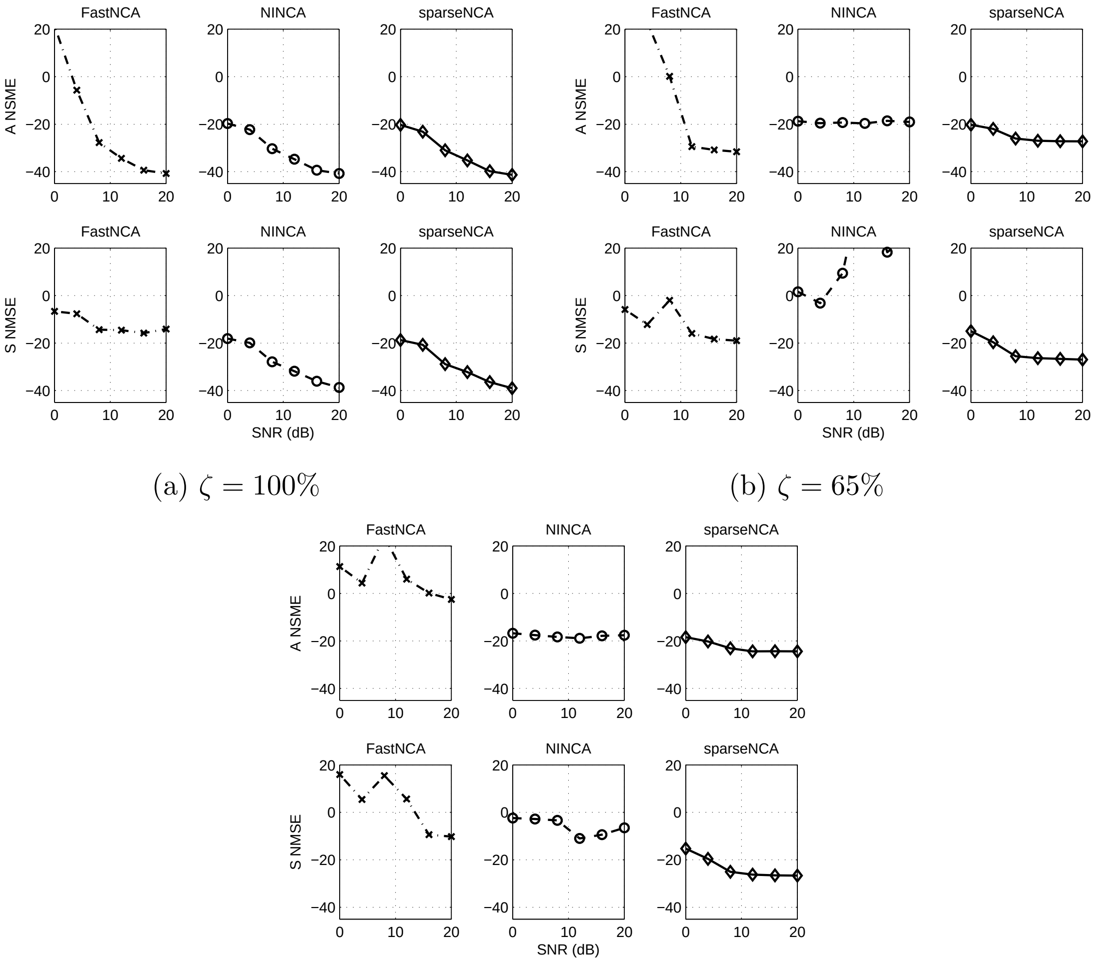
[82] X. Zhou, X. Wang, and E. R. Dougherty. Genomic networks: Statistical inference from microarray data. *Wiley Encyclopedia of Biomedical Engineering*, 2006.

[83] X. Zhou and S. T. C. Wong. *Computational systems bioinformatics.* World Scientific Publishing Co., Singapore, 2008.

[84] X. Zhu, M. Gerstein, and M. Snyder. Getting connected: analysis and principles of biological networks. *Genes & Development*, 21(9):1010–1024, 2007.

APPENDIX A

SPARSENCA IN HIGH CORRELATION DATA

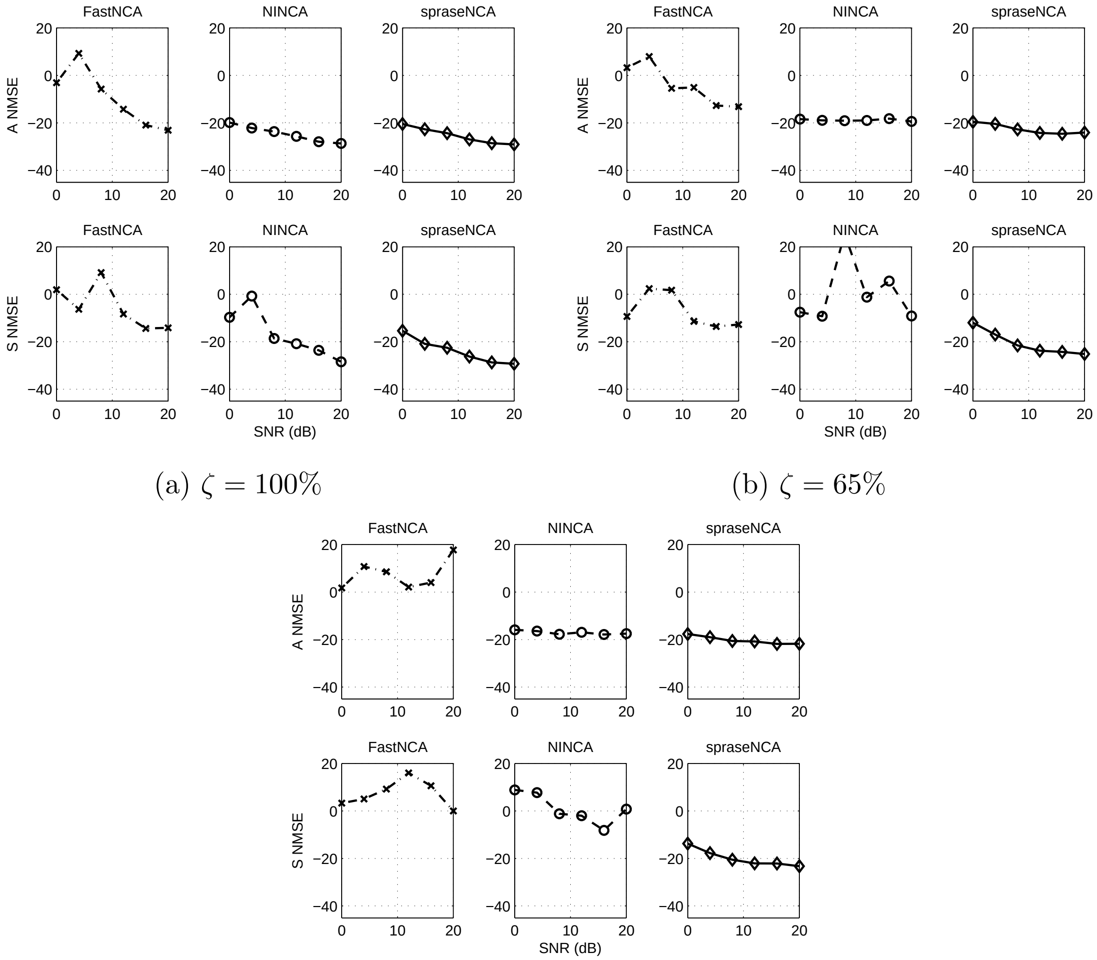


(a) $\zeta = 100\%$

(b) $\zeta = 65\%$

(c) $\zeta = 35\%$

Figure A.1: Impact of incomplete prior and outliers: Normalized mean square error (NMSE) for FastNCA, NINCA and sparseNCA for different datasets with level of outliers: 0.001 against varying signal to noise ratio (SNR) dB for high correlation data.



(a) $\zeta = 100\%$

(b) $\zeta = 65\%$

(c) $\zeta = 35\%$

Figure A.2: Impact of incomplete prior and outliers for high correlation dataset: Normalized mean square error (NMSE) for FastNCA, NINCA and sparseNCA for different datasets with level of outliers: 0.03 against varying signal to noise ratio (SNR) dB for high correlation data.

APPENDIX B

TFA RECONSTRUCTION BY SPARSENCA

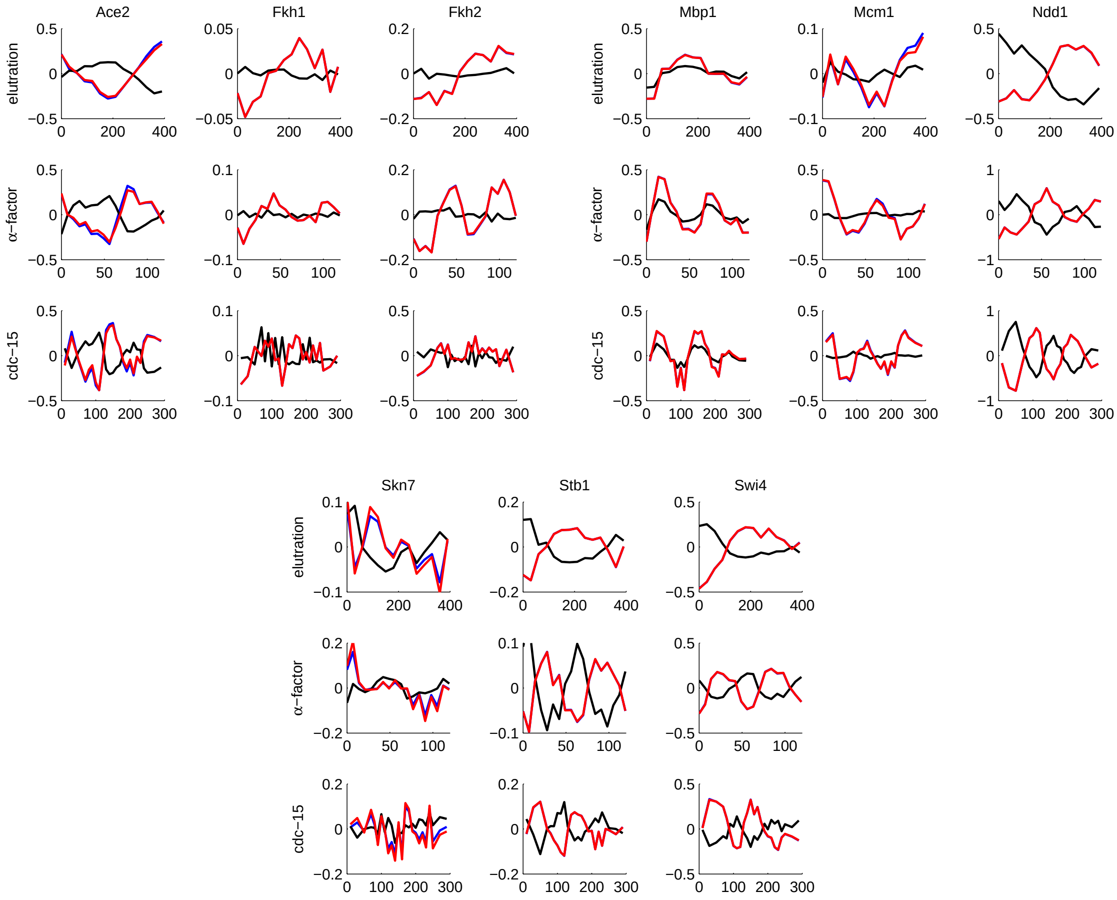


Figure B.1: TFAs reconstruction using Yeast cell-cycle dataset: Estimation of 11 (9 shown) TFAs of cell-cycle regulated yeast TFs. Average values of the TFs are shown for the four subnetworks. The results of FastNCA(black), NINCA (blue) and sparseNCA (red) are given. NINCA and sparseNCA are able to identify the periodicity of almost all the TFs and their results agree with each other.

APPENDIX C

STANDARD DEVIATION FOR TFA ESTIMATION

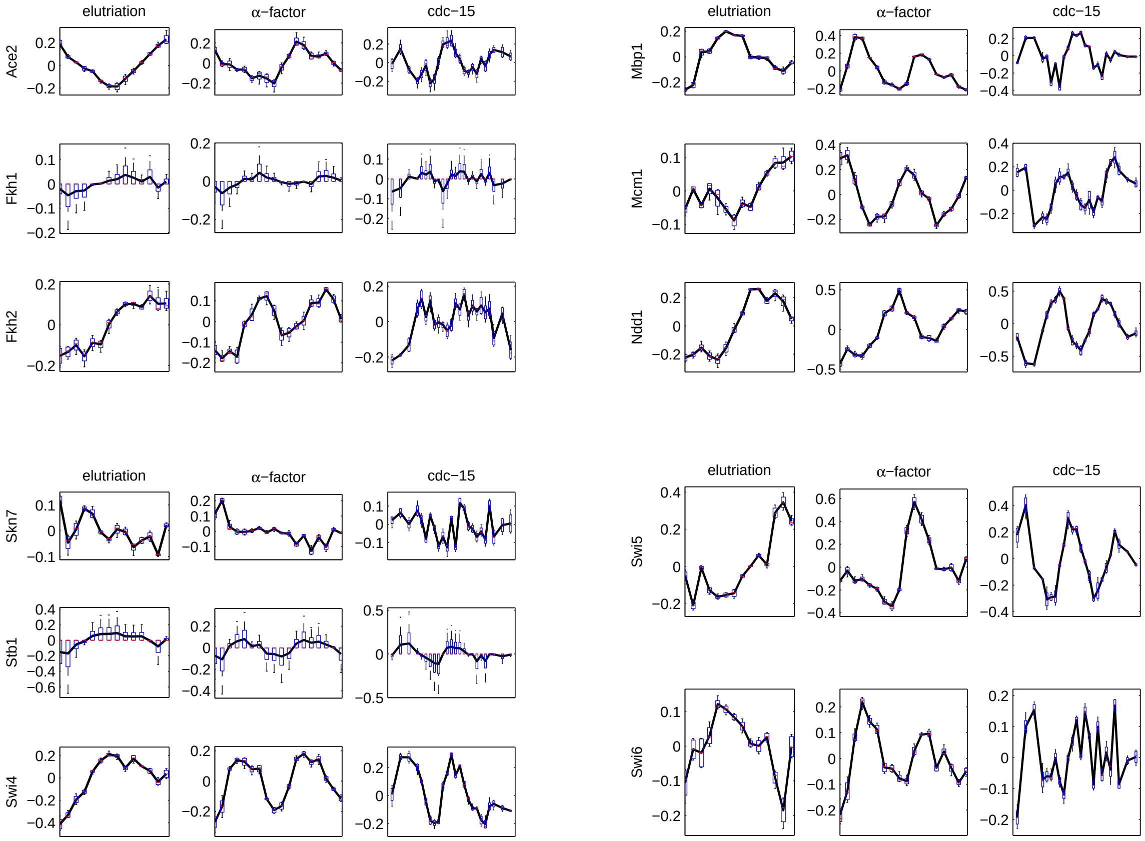


Figure C.1: Standard deviation in TFAs reconstruction for ROBNCA: Estimation cof 11 TFAs of cell-cycle regulated yeast TFs. Average values of the TFs are shown for the four subnetworks.

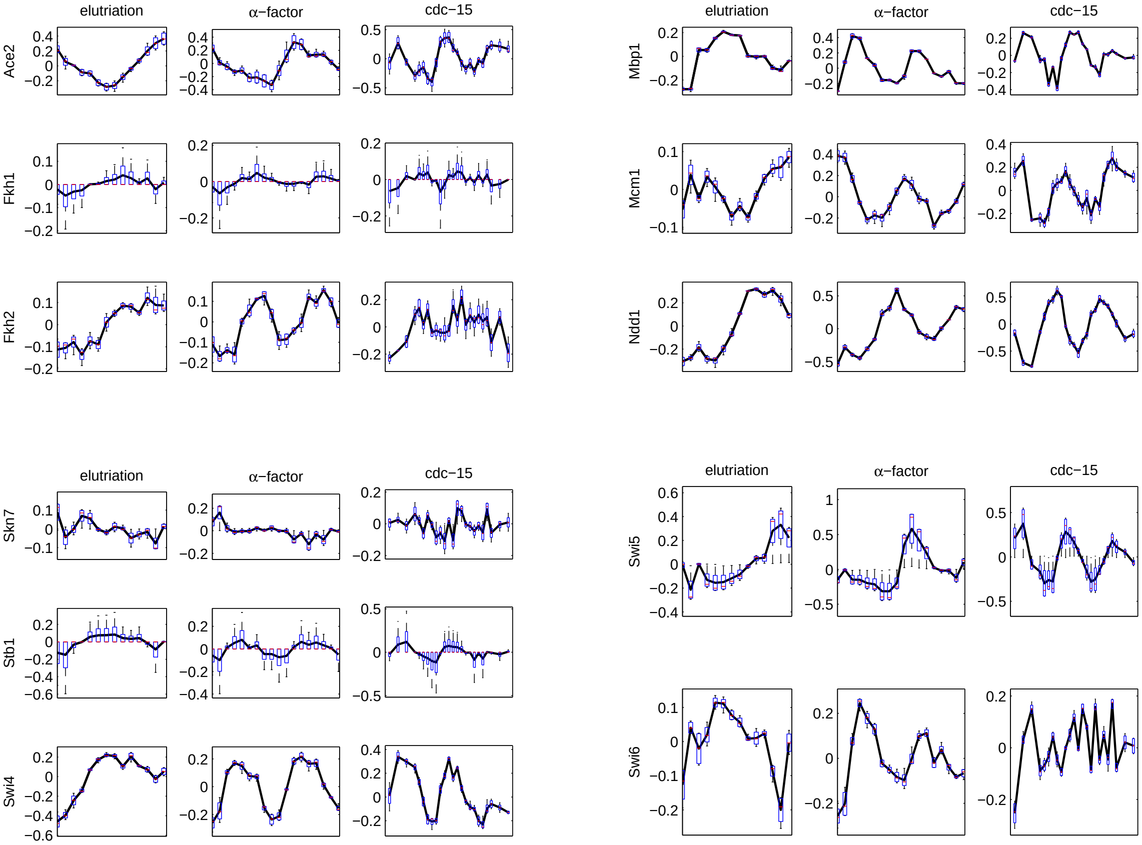


Figure C.2: Standard deviation in TFAs reconstruction for NINCA: Estimation of 11 TFAs of cell-cycle regulated yeast TFs. Average values of the TFs are shown for the four subnetworks.

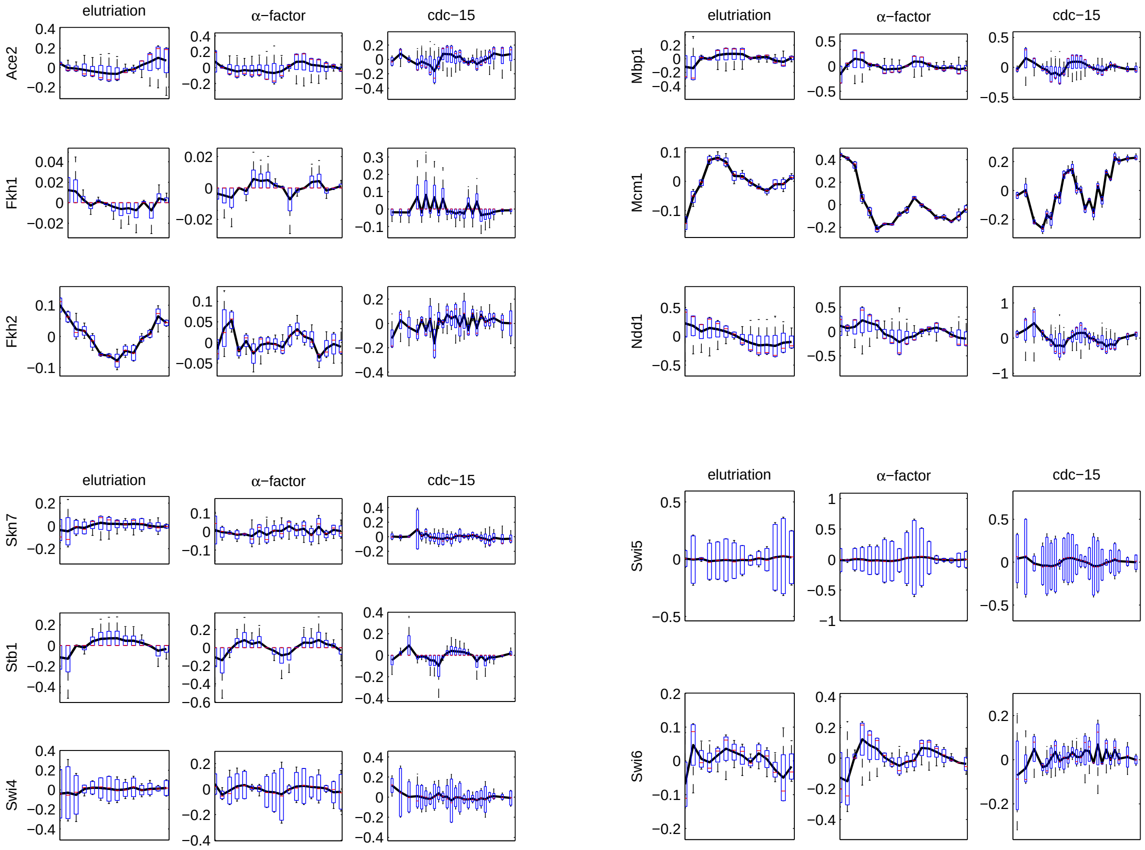


Figure C.3: Standard deviation in TFAs reconstruction for FastNCA: Estimation of 11 TFAs of cell-cycle regulated yeast TFs. Average values of the TFs are shown for the four subnetworks.