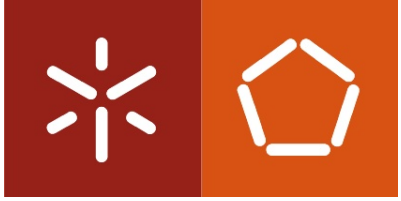
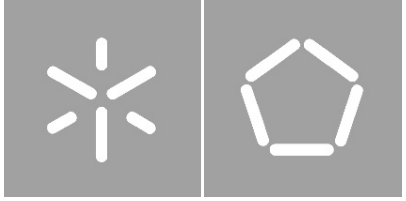


Universidade do Minho
Escola de Engenharia

Marta R. A. Matos

Network Inference: extension of a linear programming model for time-series data

Setembro de 2013



Universidade do Minho
Escola de Engenharia

Marta R. A. Matos

Network Inference: extension of a linear programming model for time-series data

Tese de Mestrado
Mestrado em Engenharia Informática

Trabalho realizado sob orientação de
Professora Doutor Rui Mendes
Doutora Bettina Knapp

Setembro de 2013

Acknowledgements

I want to thank both my supervisors: Rui Mendes for his support, and Bettina Knapp, for all her patience and dedication. In particular, for the huge amount of time she invested on me, and for all the discussions we had. I also want to thank Lars Kaderali for hosting me at his group – at the Institute for Medical Informatics and Biometry (IMB) –, for the excellent conditions provided, and for the insightful discussions. Furthermore, I would like to focus that my stay at IMB was a very enriching experience. Not only did I learn a lot about network inference in general, but I also came into contact with other research topics related to Bioinformatics/Computational Biology/Systems Biology, which helped to broaden my horizons.

Finally, I want to thank Jorge Leitão for asking the right questions, and I acknowledge the Erasmus scholarship 2012-1-PT1-ERA02-12586 (11.3) 217/2012 for financial support during my eight month Erasmus Placement.

Abstract

With the widespread availability of high-throughput technologies, it is now possible to study the behavior of dozens or even hundreds of gene/proteins through a single experiment. Still, these experiments provide only the gene/protein expression values, telling nothing about their interactions with each other. To understand these interactions, network inference methods need to be applied. By understanding such interactions, new light can be shed into biological processes and, in particular, into disease's mechanisms of action, providing new insights for drug design: which genes/proteins should be targeted in order to cure/prevent a specific disease.

In this thesis, we developed and tested two alternative extensions for a previously developed model based on linear programming. Such model infers signal transduction networks from perturbation steady-state data. The extensions now developed take advantage of perturbation time-series data, which further improves the resolution of causal relationships between genes/proteins.

In a first phase, we use artificial networks with simulated data to test the performance of both extensions in different conditions. Additionally, we compare their performance to the original model and to a state-of-the-art model for perturbation time-series data, DDEPN. Overall, our second extension exhibits a better performance, and significantly higher sensitivity. This extension assumes a given gene/protein can only influence its targets if it is in an active form.

In a second phase, we use two experimental datasets related to ERBB signaling and evaluate the resulting networks: 1) by finding literature support for the inferred edges, and 2) by using a network assembled with Ingenuity IPA as true network to do a quantitative assessment. Our results are further compared to DDEPN and the original model in a quantitative way. Quantitatively, our second model extension is shown to perform better than both the original model and DDEPN. Qualitatively, we find literature support for most of the inferred edges in both datasets, while also inferring a few plausible edges for which no literature evidence was found.

Resumo

Com o uso generalizado de tecnologias de alto rendimento como os *microarrays* de ADN, torna-se comum estudar dezenas ou mesmo centenas de genes/proteínas numa única experiência. Contudo, estas experiências apenas nos permitem determinar a expressão dos genes/proteínas e nada nos dizem sobre as interações entre os mesmos. Assim, torna-se necessário o uso de métodos de inferência de redes, de modo a estudar as interações entre genes/proteínas. Ao perceber estas interações, não só é possível perceber melhor os processos biológicos em geral, como também o modo como actuam as doenças, de forma a desenvolver novos medicamentos.

Nesta tese de mestrado, desenvolvemos e testámos duas extensões para um modelo baseado em programação linear. Este modelo infere redes de transdução de sinal a partir de experiências de RNAi em que as medidas são feitas após a perturbação, quando a rede se encontra em estado estacionário. Com as extensões desenvolvidas nesta tese é possível tirar partido de séries temporais de dados provenientes de experiências de RNAi, o que permite distinguir relações de causalidade entre proteínas.

Numa primeira fase, usamos redes artificiais e dados simulados para testar a performance de ambas as extensões em diferentes condições. Além disso, comparamo-las com o modelo original e com um modelo recente, DDEPN, que usa séries temporais de dados de experiências em que a rede a inferir é perturbada. Em geral, a nossa segunda extensão obtém melhores resultados, principalmente em termos de sensibilidade. Esta extensão assume que só proteínas activas podem influenciar outras proteínas.

Numa segunda fase, usamos dois conjuntos de dados experimentais e avaliamos os resultados obtidos: 1) procurando referências na literatura para as ligações inferidas, e 2) usando uma rede de referência para fazer uma avaliação quantitativa e estabelecer comparações com o modelo original e o DDEPN. Quantitativamente, a nossa segunda extensão obtém melhores resultados do que o modelo original e o DDEPN. Qualitativamente, encontrámos suporte na literatura para a maioria das ligações inferidas pela segunda extensão. Inferimos ainda algumas ligações bastante plausíveis, embora não tenhamos encontrado suporte para estas.

Contents

Nomenclature

ACC	Accuracy
BNs	Bayesian Networks
CV	Cross Validation
DBNs	Dynamic Bayesian Networks
DDEPN	Dynamic Deterministic Effects Propagation Networks
DEPN	Deterministic Effects Propagation Networks
FN	False Negatives
FP	False Positives
GA	Genetic Algorithm
K-fold CV	K-fold Cross-Validation
LP	Linear Programming
MAD	Median Absolute Deviation
MSE	Mean Squared Error
NEMs	Nested Effects Models
PR	Precision
RNAi	RNA interference
SN	Sensitivity
SP	Specificity

TFs	Transcription Factors
TN	True Negatives
TP	True Positives
GRNs	Genetic Regulatory Networks
RPPAs	Reverse-Phase Protein Arrays
STNs	Signal Transduction Networks

List of Figures

List of Tables

Chapter 1

Introduction

By modeling biological networks in general it is possible to understand how their different components - genes, proteins, metabolites, etc. - interact with each other. Three of the most important types of biological networks are gene regulatory networks (GRNs), signal transduction networks (STNs), and metabolic networks. GRNs are constituted by genes and proteins, and the objective is to understand how these genes and proteins regulate each other. STNs are a set of pathways in which proteins interact with each other, allowing the cell to respond to external stimuli. Finally, metabolic networks are basically a series of biochemical reactions through which initial molecules are transformed in different products.

Understanding the inner workings of these biological networks is important, not only for the sake of knowledge itself, but also because of its potential applications, for instance, by understanding a disease mechanisms of action, it may be possible to prevent or cure it. The ERBB signaling network is an example of a network whose role on disease onset needs to be better understood. Because the ERBB receptors affect the cell cycle, when overexpressed, these can lead to high levels of cell proliferation [86] and, by allowing defective cells to survive, may result in tumor formation. Therefore, by understanding how this network functions, it may be possible to counteract its negative effects in the cell cycle. Another application of this kind of knowledge is to model metabolic networks and find how to modify the network so that the cells can transform substrates into products at an higher rate, this is important for the production of beer, wine, bread, among other products.

In this thesis we will focus on signal transduction networks.

Signal Transduction Networks

Signaling networks work roughly in the following way: molecules outside of the cell

(ligands) bind to receptors located in the cell membrane and activate them, which in turn activate other molecules inside the cell (messengers), initiating the signal transduction process. This process stops when the signal achieves a transcription factor that will affect gene transcription, and thus influence gene expression and even the cell cycle progression. However, this signaling process does not necessarily evolve in a linear way, one ligand can activate different receptors, the same messenger can pass on the signal to different transcription factors, and so on. Hence, there is signal cross-talk [42], and studying each receptor/messenger/transcription factor individually is not the best approach, there is a need to take a more global approach and study the whole network.

In this thesis we are particularly interested in the ERBB signaling network, as the two experimental datasets used to test our model refer to this network.

ERBB signaling network

The ERBB signaling network "starts" in the ERBB molecules, receptor tyrosine kinases located in the cell's membrane, which trigger different signaling pathways upon stimulation by ligands such as Epidermal Growth Factor (EGF) or neuroregulin (NRG). The ERBB molecules family is composed by ERBB1 (also known as EGFR and HER1), ERBB2 (Neu, HER2), ERBB3 (HER3), and ERBB4 (HER4), and is present in several types of cells and different organs. Both ERBB1 and ERBB4 are activated by extracellular ligands, such as EGF or NRG1-4, and directly transduce the signal into the cell, whereas ERBB2 and ERBB3 need to form heterodimers with other ERBB receptors in order to be activated and transduce the signal into the cell. This is due to the lack of an extracellular ligand domain in the case of ERBB2 and the lack of an intracellular kinase domain in the case of ERBB3. By responding to external stimuli, the ERBB molecules initiate a signaling cascade, and influence a number of cellular functions, namely: cell survival, proliferation, division, migration, cell apoptosis, among others. This signal transduction network is one of the most extensively studied.

When the behavior of the ERBB receptors deviates from the normal, either by overexpression or underexpression of one of the receptors, a number of conditions may occur. In particular, the overexpression of ERBB1 and ERBB2 have been linked to the occurrence of cancer: breast cancer, gastric cancer, among others [38]. However, disruptions in the normal functioning of ERBB receptors are not only related to tumor development, but also to diseases such as Parkinson's disease and schizophrenia. In postmortem studies, protein levels of EGF and ERBB1 were shown to be diminished

in the brains of people suffering from Parkinson's disease [39], while ERBB1 is over-expressed in the forebrain regions of people suffering from schizophrenia [30]. Plus, the correct functioning of ERBB receptors is also critical for the adult heart maintenance under stress conditions [69], and for the development of the heart trabeculae, a structure that assures the correct functioning of the embryonic heart [32, 49, 58].

ERBB signaling network influence on the cell cycle

One process influenced by ERBB signaling is the cell cycle. The cell cycle [59] is constituted by four phases, briefly: i) G1 phase, in which the cell size increases and the G1 checkpoint ensures that it is ready for DNA replication. During this phase several signals intervene to influence decisions such as whether the cell will self-renew, differentiate, or die; ii) Synthesis, in which DNA replication takes place, by the end of it all of the chromosomes have been replicated; iii) G2 phase, this checkpoint ensures that the cell is ready for the next phase and mends replication errors that might have occurred, while the cell continues to grow; iv) Mitosis, in which the cell divides into two daughter cells containing the same genetic material.

Focusing now in the G1 phase, it starts with the association of Cyclin Dependent Kinases (CDK) 4 and 6 with D-type Cyclins which phosporylate the retinoblastoma protein, pRb. Once pRb is phosporylated it stimulates CDK2 and E-type Cyclins which, in turn, further phosphorylate pRb. At this point the cell cycle becomes independent of CKD4/6 and Cyclin D complexes, and it can proceed into the S phase. One way to stop cell cycle progression to the S phase is by activating proteins of the p16 (p16, p15, p18, p19) and p21 family (p21 , p27, p57), since when activated the p16 family members are able to inactivate CDK4/6, while p21 family members can inactivate Cyclins. Both resulting in the prevention of pRb phosporylation during the G1 phase, and stopping the cell cycle. A shorter G1 phase and early transition to the S phase has been linked to ERBB2 overexpression, leading to cell hyperproliferation [86], which is key to tumor development. This effect is thought to be mediated by the up-regulation of CDK6 and Cyclins D1 and E, as well as enhanced degradation and relocalization of p27 [86].

The study of the ERBB signaling network as a whole, and biological networks in general, would not be possible without the advent of high-throughput technologies, such as gene microarrays, Reverse-Phase Protein Arrays (RPPA), or RNA interference (RNAi) experiments.

RNAi

RNA interference is a technique that uses double-stranded RNA (dsRNA) to interfere with specific sequences of complementary mRNA and inhibit the expression of corresponding genes.

RNAi was first discovered by Fire and Mello [18], who noted that double-stranded RNA (dsRNA) served as mediator in post-transcriptional silencing in *Caenorhabditis elegans*, and has since then been widely used to inhibit the expression of targeted genes in an high-throughput fashion. RNAi has also a potential application in therapy, by silencing the expression of specific genes responsible for the disease [68].

An overview of the RNAi mechanism is shown in figure 1.1. The process can start either with long double-stranded RNA (dsRNA) or pre-microRNA segments, however we will focus on the dsRNA mediated process, since RNAi experiments are usually based on this process. After entering the cell cytoplasm, the dsRNA is spliced into smaller dsRNA segments by the Dicer enzyme. The resulting dsRNA segments are termed small-interfering RNA (siRNA), and are then incorporated into a multi protein complex that includes the cleaving enzyme Argonaute 2 (Ago2) and the protein complex RNA-induced silencing complex (RISC). In this RISC/Ago2 multi protein complex, the siRNAs double strand is separated into guide and passenger strands, the latter being discarded, while the first will target a complementary mRNA sequence to be cleaved by the RISC/Ago2 and degraded, leading to post-transcriptional gene silencing. However, the use of long dsRNA segments for post-transcriptional gene silencing in mammals triggers an interferon anti-viral response. Therefore, chemically synthesized siRNAs are directly introduced in the cell cytoplasm to be incorporated into the RISC/Ago2 complex.

RNAi, by silencing gene expression in between transcription and translation is advantageous in relation to DNA gene knockout, since the chance that compensatory mechanisms are activated is considerably lower. Furthermore, it is faster, less expensive, and the time frame of the knockdown can be controlled by the experimenter [44], hence allowing the systematic study of gene function and respective role in given biological processes. Still, an RNAi experiment must be carefully designed in order to reduce possible off-target effects, i.e., silencing genes other than the targeted ones.

Reverse-Phase Protein Arrays

Reverse-Phase Protein Arrays (RPPAs) [65] is the technique used to produce the experimental data analyzed in this work. By using RPPAs it is possible to study a cell's response to external stimulus, as this usually reflects on protein post-translational

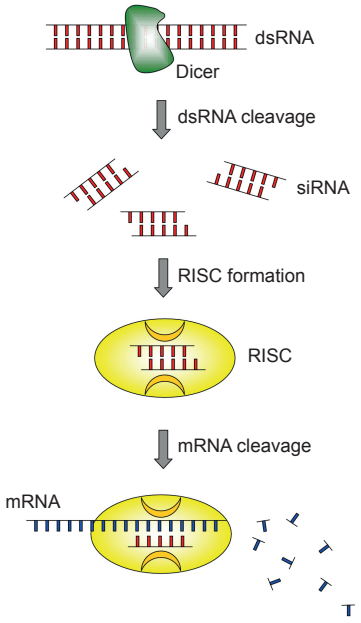


Figure 1.1: The RNA interference process described in this figure starts with a dsRNA segment, which is cleaved by the enzyme Dicer, resulting in several siRNA segments. The siRNA segments are then included into the RISC complex and degrade complementary mRNA sequences, resulting in post-transcriptional gene silencing.

modifications, such as phosphorylation or change in protein activity. To measure protein abundance by using RPPAs, the cells of interest are first lysed, and the protein lysates are spotted in the array, which are then probed with an antibody specific for the target protein, the primary antibody. Next, the array is incubated with a secondary antibody which is labeled with a near-infrared dye, the purpose of which is to detect the primary antibody and quantify the amount of protein present in the spot.

Applications of RPPAs include quantitative analysis of protein expression in cancer cells, cell signaling analysis, or clinical prognosis/diagnosis/therapeutic prediction, among many others. However, we are particularly interested in the use of RPPAs to study signaling pathways, by monitoring protein dynamics in response to perturbations.

1.1 Approaches to Network Inference in Biology

There are several approaches to infer biological networks, all of them modeling the network as a graph, in which the nodes represent the network components – e.g., genes, proteins, transcription factors, metabolites –, and the edges represent the

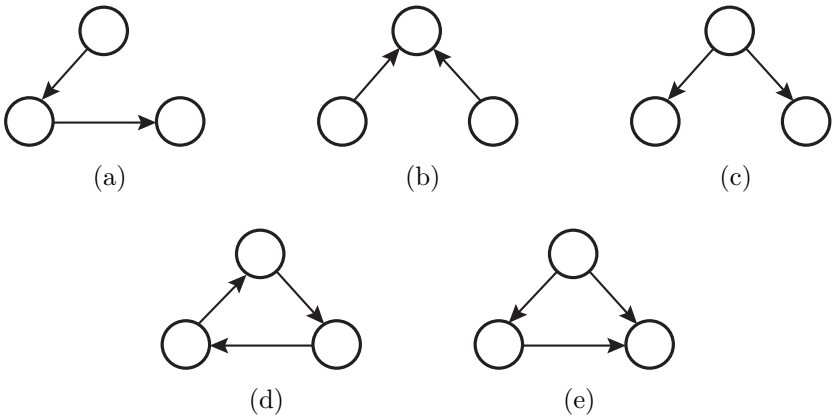


Figure 1.2: Several motifs present in most biological networks are: (a) cascade: one node influences the next in a sequential way; (b) fan-in: two nodes influence a third node; (c) fan-out: one node influences two other nodes; (d) feedback loop: one node influences a second node, which influences a third node, and the third node influences the first one; and (e) feedforward loop: one node influences two other nodes, and one of these influences the remaining node.

interactions among these components. These edges might have a direction or not, in the former case an edge from node A to node B means that A influences B, in the latter it simply means that A and B are correlated. In addition, directed edges can have a sign or not, specifying whether the influence of one node over another is activating or inhibiting.

Although a given method is usually developed with the intent of inferring either STNs or GRNs, often the same approach can be used to infer both types of networks. For instance, Bayesian networks can be used either to infer GRNs [37] or STNs [67]. This is possible because STNs and GRNs have a similar topology that results in similar inference challenges. For instance, both types of networks possess the same kind of motifs, small substructures present in the network topology, such as feedback loops, in which one activates a second node, which activates a third node, that in turn activates the first node again. For more motif examples and description, please see figure 1.2 and respective caption. Therefore, the choice of which approach to use when modeling a GRN or a STN depends mostly on the data one has available, for instance, if it is time-series data or was measured at a single time point, if it contains perturbations or not.

Classical approaches

Some classical approaches to infer biological networks include Boolean networks

[41], Bayesian networks (BNs) [21, 73], and the use of differential equations [4, 10, 31, 43, 75]. Boolean networks assume each node is either active or inactive and infer the network by using logical rules. Yet, the data used to infer networks is usually continuous and thus needs to be discretized, leading to loss of information and possibly having a negative impact in the network inference results, since finding the right threshold to discretize the data is usually not a straightforward process. Several methods based on this approach exist, and can be applied to time-series data [57, 36] or perturbation data measured at a single time point [92]. On the other hand, Bayesian networks do not necessarily require data discretization and can be used with continuous data. This formalism infers the connections among biological components from a component's state given the parents state, using data from measurements at a single time-point. Some advantages of Bayesian networks are their ability to accommodate noisy data and represent complicated stochastic nonlinear connections between several nodes. However, Bayesian networks have a few drawbacks: due to its probabilistic nature they need several repeated measurements to infer the network topology [73], besides, BNs are limited to be acyclic and cannot infer feedback loops in the network structure [73]. The latter can be circumvented by using Dynamic Bayesian Networks (DBNs) [22] instead, which use time-series data to unfold the network behavior in time, thus capturing loop structures when present. Network inference based on differential equations also requires time-series data, using it to model the genes/proteins behavior along time and from that infer the respective network; a particular method is the S-System formalism within the Biochemical Systems Analysis [76, 77]. This approach is usually computationally very expensive.

One common trait among the methods described above, whether they use single time-point measurements or time-series data, is that these assume the underlying network to be in a stationary state, i.e. the connections among biological components do not change along time. Even though the connections that are active at a given time point t may not be the same than the ones active at $t+1$, this is only due to the fact that not every node has the same state in both time points, and not because the underlying network topology has changed. Usually, the underlying network topology only changes when there are more drastic changes in the organism, for instance due to development [71].

Time-dependent networks

In case there is a need to consider changes in the underlying network, i.e. the

network is not in a stationary state, the methods described previously are no longer enough to infer these changes.

In this situation, the conditional probability of a gene to be in state $X[t+1]$ at time $t+1$ given that it was in state $X[t]$ at time t ($P(X[t+1]|X[t])$), is not independent of t because the distribution generating the time-series changes with time, i.e. the underlying process is non-stationary. Some methods developed to address this problem are non-stationary or non-homogeneous DBNs [33, 34, 35, 47, 72, 71], which are a type of piece-wise stationary model that determines the time points at which the probability $P(X[t+1]|X[t])$ changes.

In short, there are two types of methods suited to infer networks from time-series data, one assumes that changes in the underlying network can occur, while the other type assumes that the underlying network is the same throughout all measurements, and the changes in the measurements results are only due to change of genes/proteins states. The latter one is the type of networks we aim to infer with our method, assuming a static underlying network.

Comparing to data measured at a single time point, the use of time-series data has several advantages. One example is that, if we assume the expression of a given protein to change before it can influence its target proteins later in time, then time-series data helps to resolve causal relationships [12]. Yet, the measurements need to be performed at appropriate times, i.e. after the activation of one protein and before the activation of its targets.

Another type of data that has been shown to lead to better inference results is perturbation data [56], where perturbation of the network can occur either by performing RNA interference experiments, gene knockouts, or by the use of drugs. Considering two genes with similar expression values at a given time point, it may not be possible to infer a causal relation, i.e. which gene influences the other. However, if each gene is perturbed in a different experiment, and the effect on the expression of the other gene is measured, one can infer the causal relation between them.

Briefly, by using time-series and/or perturbation data causal relations between genes/proteins can be resolved. This also helps to distinguish direct from indirect connections, and to infer network structures such as feedback loops, that are usually hard to infer.

Approaches using perturbation data

Different approaches exist that can take advantage of perturbation data, includ-

ing approaches based on BNs [73, 66], DBNs [14], Boolean networks [92], Bayesian networks with probabilistic Boolean threshold functions [40], or differential equations [62]. However, a major recent approach is the Nested Effects Models (NEMs) formalism [54, 55] to infer STNs.

Nested Effects Models

It is a probabilistic approach that aims at inferring a signaling network by perturbing a set of genes and measure its influence on downstream genes. Thus, this method infers networks from indirect steady-state observations rather than direct ones, as most methods do. The formalism divides the nodes into S-genes and E-genes; S-genes are the set of candidate pathway genes that are silenced, while E-genes are the genes downstream of the S-genes which show the effects of silencing S-genes. The data for this method is collected by stimulating the pathway while perturbing the S-genes (using RNAi, gene knockdowns, or protein inhibiting drugs), and only the expression of E-genes is measured, which must be significantly different when the pathway is stimulated from when it is not. In this way, the relationships among the S-genes are inferred from the observations of E-genes, which are used solely as reporters of the pathway's signal flow.

However, this approach can infer only small-scale networks (up to 6 genes). Hence, it was later extended to infer networks on a pathway-wide scale by clustering the E-genes based on their phenotypic profiles and using a divide and conquer approach [55]. Yet, this approach has some disadvantages: it cannot distinguish inhibiting from activating edges; requires data discretization leading to informations loss; cannot distinguish direct from indirect connections; and requires a large number of observations of downstream effects for a small number of perturbations.

NEMs extensions

The NEM formalism has since then been extended by other authors [87, 23, 26, 93, 90, 25, 3, 28, 27]. In particular Froehlich *et al* [23] developed a new approach that does not require data discretization, instead it represents the likelihood that a set of E-genes is influenced by an S-gene using p-values. Two other extensions include the ability of using time-series data, the first of these extensions is D-NEMs and was developed by Anchang *et al* [3]. It aims at modeling the temporal evolution of multiple rounds of signaling, gene regulation, and gene expression, by inferring the time delays between an S-gene perturbation and its downstream effects on the E-genes. However, to determine the time delays Anchang *et al* use Gibbs sampling, which makes the

whole process of network inference very time consuming. On the other hand, Froehlich *et al* developed a second extension of NEMs to use time-series data, DynoNEMs [27]. DynoNEMs unrolls the signal flow over time in a way similar to DBNs, and simply calculates the time lag between a perturbation and a downstream effect, therefore avoiding the use of Gibbs sampling and rendering the approach less computationally expensive. This approach also allows the use of combinatorial perturbations, which the original model did not allow. Furthermore, both of the approaches to use time-series data enable, in principle, the inference of feedback loops in the network topology and the distinction between direct and indirect connections.

Deterministic Effects Propagations Networks

An approach developed to take advantage of RNAi experiments which uses direct observations instead of indirect ones is the Deterministic Effects Propagations Networks (DEPNs) model published by Froehlich *et al* [29]. DEPNs is a special case of Bayesian networks which uses both deterministic and Gaussian variables to infer STNs. Briefly, the approach works as follows: i) for each perturbation experiment its expected downstream effects are calculated, ii) knowing the downstream effects of all perturbations, each protein is considered either as perturbed or unperturbed, iii) each protein is then set as part of one out of two distributions (one for perturbed proteins, another for unperturbed proteins), which are assumed to be Gaussian and whose parameters are calculated either in a maximum likelihood or a Bayesian way.

This formalism was designed to be used in the opposite context of NEMs, instead of requiring little perturbations and high-dimensional downstream measurements, DEPNs benefit from as many perturbations as possible (which can also be combinatorial) and do not need high-dimensional measurements. Another advantage in relation to NEMs is that DEPNs do not require data discretization. Furthermore, when comparing to Bayesian networks, DEPNs allow for the inference of loops in the network topology, which is possible because the model relies on deterministic effects propagation, rendering all measurements statistically independent. Yet, this formalism cannot take advantage of time-series data, this would be the motivation for the Dynamic Deterministic Propagations Networks (DDEPNs) model. Moreover, the network topologies inferred by DEPN are always transitively closed, and the model is computationally expensive, i.e. it is not suitable for inference of large networks.

A Linear Programming approach to STN inference

Knapp and Kaderali [45] developed a model based on linear programming (LP)

that, like DEPN, uses perturbation steady-state data from direct observations to infer signal transduction networks. In particular, it supports RNAi data in which multiple genes are silenced in a single knockdown experiment. This model assumes the network to be modeled as an information flow, in which the signaling starts at one or more source nodes and propagates downstream in a deterministic way until it reaches the sink nodes. The network is then inferred by formulating a linear problem, in which the sum of all edges, baseline node activities, and slack variables must be minimized under certain constraints.

One key advantage of this model is its reduced running time. In particular, when comparing to DEPN, it achieves better sensitivity and specificity in a significantly shorter amount of time.

Since the subject of this dissertation is the extension of this model to take advantage of time-series data, it will be further described in section 2.2.

Dynamic Deterministic Effects Propagations Networks

DDEPN [6] uses perturbation time-series data from direct observations to infer signaling networks. The perturbations can be both stimuli or inhibitions. Since we are comparing the model developed in this thesis to DDEPN, it will now be described in more detail than the previous approaches. The reason to choose DDEPN for results comparison is that, like the model developed in this thesis, it is able to take advantage of perturbation time-series data obtained from direct observations.

The first step in the model is, given a possible network topology, to generate a matrix containing all the system's reachable states. Consider $V = \{v_i : i \in 1, ..., N\}$ as the set of nodes, where N is the number of nodes, and $\Phi = V \times V \rightarrow \{0, 1, 2\}$ as the adjacency matrix that defines the network,$\Phi_{ij} = 0$ means no edge, $\Phi_{ij} = 1$ means an activation edge, and $\Phi_{ij} = 2$ an inhibition edge between two nodes. The signal flow through the given network is then encoded in a matrix containing a series of possible system states: $\Gamma = \{\gamma_{ik} \in \{0, 1\} : i \in 1, ..., N, k \in 1, ..., M\}$, where $\gamma_{\mathbf{k}} = \{\gamma_i : i \in 1, ..., N, \gamma_i \in \{0, 1\}\}$ and $0 \leq M \leq 2^N$ is the number of reachable system states. Assuming the stimuli nodes (which can be either activating or inhibiting) to be always active, and all other nodes to be inactive at the first time point, the signal propagates downstream in the network as follows: let $pa\,(v_i)$ be the set of parents of node v_i and ϕ_{wv_i} an edge from node w to node v_i, then

$$E^{+}_{k-1}\,(v_i) = \{\gamma_{wk-1} : \phi_{wv_i} = 1, \forall w \in pa\,(v_i)\}$$

$$E_{k-1}^{-}(v_i) = \{\gamma_{wk-1} : \phi_{wv_i} = 2, \forall w \in pa(v_i)\}$$

is the set of parent nodes of v_i at step $k-1$ connected either by an activating or inhibiting edge, respectively. In this way, an entry γ_{v_ik} in Γ is set as:

$$\gamma_{v_ik} = \left(\bigvee_{e^+ \in E_{k-1}^{+}(v_i)} e^+ \right) \wedge \neg \left(\bigvee_{e^- \in E_{k-1}^{-}(v_i)} e^- \right) \tag{1.1}$$

i.e., a node is active if, in the previous time point, at least one parent node connected by an activation edge is active, and all parent nodes connected by inhibiting edges are inactive, otherwise the node will be inactive.

Having the matrix of reachable system states, an optimal state sequence is calculated using Hidden Markov Models (HMMs). Thus, consider $t \in 1, ..., T$ as the time point index, $r \in 1, ..., R$ as the replicate measurements index, $X = \{x_{itr} : i \in 1, ..., N,\ t \in 1, ..., T,\ r \in 1, ..., R\}$ as the data matrix, and $\Gamma^* = \{\gamma^*_{itr} : i \in 1, ..., N,\ t \in 1, ..., T,\ r \in 1, ..., R\}$ as the unknown true sequence of reachable system states. Each entry in Γ^* represents the state of a node i (active or inactive) at time point t, where replicate measurements are assumed to have the same state. Each entry corresponds also to a measurement x_{itr}. To infer an estimate $\hat{\Gamma}$ for Γ^* an HMM $H = (W, \Gamma, A, e)$ is used, where W is the range of all possible values for the observations, Γ is the set of states as derived in 1.1, A is the matrix of states transition probabilities, and e is the emission probability $e(\mathbf{x_t}) = p\left(\mathbf{x_t}|\hat{\gamma_t}, \hat{\Theta}\right)$, in here $\mathbf{x_t}$ is a column in the observation matrix containing all entry states at time point t, $\hat{\Theta}$ is the matrix of estimated model parameters, and e is the probability of observing $\mathbf{x_t}$ given the node state $\hat{\gamma_t}$. To find the optimal state sequence of system states and optimize both A and $\hat{\Theta}$, the Viterbi algorithm is used. The matrix $\hat{\Gamma}$ is initialized by sampling random states from Γ while preserving the states order, and the transition matrix A is initialized to uniform probabilities for all state transitions. Then the model parameters $\hat{\Theta}$ are calculated based on $\hat{\Gamma}$, which is updated using the HMM, and everything is repeated until convergence is achieved.

To calculate the total network likelihood and, consequently, the emission probabilities $e(\mathbf{x_t})$, the following score was set up:

$$\begin{aligned} p\left(X|\Phi\right) &= p\left(X|\hat{\Gamma},\hat{\Theta}\right) = \prod_{t=1}^{T} p\left(x_t|\hat{\gamma}_t,\hat{\Theta}\right) \\ &= \prod_{t=1}^{T}\prod_{i=1}^{N}\prod_{r=1}^{R} p\left(x_{itr}|\hat{\theta}_{i\hat{\gamma_{itr}}}\right) \end{aligned} \tag{1.2}$$

where $\hat{\Theta} = \left\{\hat{\theta}_{i0}, \hat{\theta}_{i1}\right\} = \left\{\left(\hat{\mu}_{i0}, \hat{\sigma}_{i0}\right), \left(\hat{\mu}_{i1}, \hat{\sigma}_{i1}\right)\right\} \forall i \in 1, ..., N$, and $\hat{\mu}_{i0}, \hat{\sigma}_{i0}$ are the mean and standard deviation of the observations $\mathbf{x_i}$ calculated for each node i (at all time points and replicates) that is inactive according to matrix $\hat{\Gamma}$, while $\hat{\mu}_{i1}, \hat{\sigma}_{i1}$ are the mean and standard deviation of $\mathbf{x_i}$ calculated for each node i that is active according to $\hat{\Gamma}$.

Finally, a Genetic Algorithm (GA) is used to search for the network topology that maximizes the above network likelihood (eq. 1.2). The GA optimizes a population of networks by selecting and mutating individuals. To avoid overfitting the Bayesian Information Criterion is used, which is calculated as follows:

$$BIC = -2log\left(p\left(X|\Phi\right)\right) + K\,log\left(n\right)$$

where K is the number of connections in the network Φ, and n the number of data points. At each evolution step of the GA, only networks whose BIC is less than the median of all networks BIC are selected to be part of the next generation population.

An alternative to using the BIC score to select the networks that will be part of the next generation population, is to use a prior probability for the network structure [7]. This will lead to the selection of networks whose structure is similar to the network structure defined by the given prior probability.

Consider now a matrix $B = V \times V \rightarrow [-1, 1]$ that contains prior confidences for each edge in the network, then the weighted difference between an edge in the inferred network Φ and in the prior network B is defined as:

$$\Delta_{ij} = \left|\phi_{ij} - b_{ij}\right|^{\gamma}$$

where $\gamma \in \mathbb{R}^{+}$ is introduced so that the difference between edges is weighted.

Assuming that all edge probabilities are independent, the prior belief for a given network structure Φ is defined as:

$$P\left(\Phi|B, \lambda_p, \gamma\right) = \prod_{i,j} P\left(\phi_{ij}|b_{ij}, \lambda_p, \gamma\right),\ i, j \in \{1, ..., N\} \tag{1.3}$$

where the prior belief for a given edge in the network is defined in a way that penalizes large differences from the inferred network Φ to the prior belief B:

$$P\left(\phi_{ij}|b_{ij},\lambda_p,\gamma\right)=\frac{1}{2\lambda_p}e^{\frac{-\Delta_{ij}}{\lambda_p}}$$

where $\lambda_p \in \mathbb{R}^+$.

Because $\phi_{ij} \in \{0,1,-1\}$ and $b_{ij} \in [-1,1]$, all differences Δ_{ij} lie in the interval $[0,2^\gamma]$. Assuming $\gamma = 1$, we will now derive the upper and lower bonds for the prior influence, when both positive and negative edges are considered. Consider the following limits:

$$\lambda_p \to \infty \Rightarrow \begin{cases} e^{\frac{-\Delta_{ij}}{\lambda_p}} \to 1 & if\ \Delta_{ij}=0 \\ e^{\frac{-\Delta_{ij}}{\lambda_p}} \to 1 & if\ \Delta_{ij}>0 \end{cases}$$

$$\lambda_p \to 0 \Rightarrow \begin{cases} e^{\frac{-\Delta_{ij}}{\lambda_p}} \to 1 & if\ \Delta_{ij}=0 \\ e^{\frac{-\Delta_{ij}}{\lambda_p}} \to 0 & if\ \Delta_{ij}>0 \end{cases}$$

which lead to

$$0 \leq P\left(\phi_{ij}|b_{ij},\lambda_p,\gamma\right) \leq \frac{1}{2\lambda_p} \forall \lambda_p \in \mathbb{R}^+, \gamma = 1$$

and because $\Delta_{ij} \geq 0$, $\forall \gamma \in \mathbb{R}^+$, these bounds are valid for $\gamma \in \mathbb{R}^+$ in general.

To use prior information on the edge types, the value of λ_p needs to be chosen so that the prior belief is higher than the network likelihood, if we wish for a strong prior influence. In general, one should track the absolute differences between the network likelihood and the prior for each individual to a population quantile. The value for λ_p would then be chosen such that the differences between network likelihood and prior belief are slightly above zero, so that the final result is biased towards the prior network. Yet, the λ_p value to choose might depend on the dataset in use [7], and several possible values should be tested in advance. However, when using experimental datasets it may not be possible to test which λ_p value provides the best results.

This model, is implemented as an R package, ddepn [5].

1.2 Thesis objective and organization

The objective of this thesis is to extend Knapp and Kaderali [45] model, so that it can take advantage of perturbation time-series data.

This dissertation is organized as follows: in chapter 2, we introduce the linear programming formalism and the simplex method, which is the method that we use to solve our linear program. Additionally, we further detail the original model developed by Knapp and Kaderali, and only then describe the model extensions developed during this thesis. In chapter 3, we explain how the artificial networks and respective data were generated for our tests, describe the experimental datasets used and how these were preprocessed, and finally explain how our results were processed and evaluated, as well as, the experimental setup and execution settings. In chapter 4, we present the results obtained with our model for simulated data under different conditions: 1) inference of different motifs found in biological networks, 2) for an increasing number of time points, 3) for an increasing number of knockdown experiments, 4) for growing levels of noise; and for the experimental datasets. Furthermore, we discuss and compare our results to the ones obtained with DDEPN and the original LP model. Finally, in chapter 5, we present our final remarks.

Chapter 2

A Linear Programming Approach to Network Inference

2.1 Linear programming

In this section a short introduction to linear programming and the simplex algorithm will be given, as this is the formalism used to solve the network inference problem in this thesis. This section is based mostly on the linear programming books by Eiselt and Sandblom [16] and by Dantzig and Thapa [11].

Linear programming, or linear optimization, is a mathematical method that aims at solving a problem by maximizing or minimizing a linear function under certain linear constraints, and whose variables are independent and nonnegative. A linear program with n variables and m constraints can always be stated in the canonical form:

$$\text{maximize} \sum_{j=1}^{n} c_j x_j \tag{2.1}$$

$$\text{subject to} \sum_{j=1}^{n} a_{ij} x_j \leq b_i, i \in [1, m] \tag{2.2}$$

$$x_j \geq 0, j \in [1, n] \tag{2.3}$$

where a_{ij}, c_j, $b_i \in \mathbb{R} \forall \{i, j\}$ are known coefficients, and $x_j \in \mathbb{R}_0^+ \forall j$ are the variables to be determined by maximizing the objective function, eq. 2.1, while satisfying constraints 2.2 as well as the nonnegativity condition 2.3. The Linear Programming formalism is very useful to model real-world situations that can be approximated by

a linear function. An example of such a situation is the following: imagine you have a set of resources x_1, x_2, x_3 that you need in order to produce a given amount of three different products: b_1, b_2, b_3, and you need a different amount of resources for each product. Now, you want to know what is the minimum amount of resources you need in order to produce the required quantity of products. The problem can then be stated in the form:

$$\begin{aligned} \text{Min} \quad & x_1 + x_2 + x_3 \\ \text{s.t.} \quad & 3x_1 + x_2 + x_3 = b_1 \\ & x_1 + 2x_2 + 3x_3 = b_2 \\ & x_1 + 3x_2 + x_3 = b_3 \\ & x_1,\, x_2,\, x_3 \geq 0 \end{aligned}$$

by solving it you will know the exact quantity of resources x_1, x_2, x_3 that you need in order to produce b_1, b_2, b_3. In a real-world context it generally makes sense for the problem's variables to be nonnegative, in this particular example for instance, it is impossible to have negative resources.

In order to solve an LP problem using the simplex method or other, it needs to be stated in the standard form by substituting the inequalities of the type $\leq$ in constraints 2.2 by equations. The way to do this is to insert slack variables $S_i \in \mathbb{R}_0^+ \forall i$ and thus have:

$$\text{maximize} \sum_{j=1}^{n} c_j x_j \tag{2.4}$$

$$\text{s.t.} \quad a_{ij}x_j + S_i = b_i, i \in [1, m] \tag{2.5}$$

$$x_j,\, S_i \geq 0, j \in [1, n], i \in [1, m] \tag{2.6}$$

or, in matrix form:

$$\text{maximize } \mathbf{cx} \tag{2.7}$$

$$\text{s.t.} \quad \mathbf{Ax} + \mathbf{IS} = \mathbf{b} \tag{2.8}$$

$$\mathbf{x}, \mathbf{S} \geq 0 \tag{2.9}$$

	$\mathbf{x}$	$\mathbf{S}$	1	
m rows	$\mathbf{A}$	$\mathbf{I}$	$\mathbf{b}$	constraints
1 row	$-\mathbf{c}$	$\mathbf{0}$	z_0	objective function

Table 2.1: The above simplex tableau includes all constraints $\mathbf{Ax} + \mathbf{IS} = \mathbf{b}$ and the objective function $\mathbf{cx}$, where the coefficient z_0 denotes the objective function value when $x_j = 0$, $\forall j \in [1, n]$. The nonnegativity constraints are handled implicitly.

where $\mathbf{c}$, $\mathbf{b}$ are the vectors of coefficients c_j, b_i, $\mathbf{x}$ and $\mathbf{S}$ are the vectors of variables x_j and S_i, respectively, $\mathbf{A}$ is the matrix of coefficients a_{ij}, and $\mathbf{I}$ is the identity matrix.

An approach to solve this problem is to use a simplex tableau, whose general appearance is shown in table 2.1.

Before proceeding any further, some definitions need to be introduced:

- A *feasible vector* is a set of values for x_j, $j \in [1, n]$ that satisfies the LP problem constraints;
- An *optimal feasible vector* is a feasible vector that maximizes the objective function. There are only two reasons why an optimal feasible vector may not exist:
  1. The constraints are incompatible and thus there are no feasible vectors;
  2. The value for the objective function is unbounded, i.e., there are variables that can take infinitely high values while satisfying the constraints;
- A *basic solution* is a special solution obtained when setting the LP problem independent (nonbasic) variables to zero and solving for the dependent (basic) variables. In the simplex tableau presented in table 2.1 the dependent variables would be S_i, $i \in [1, m]$, and the independent variables x_j, $j \in [1, n]$.
- A basic solution is considered to be *degenerate* when the value of one or more dependent (basic) variables is zero.
- A *basic feasible solution* is a basic solution whose variable values are all nonnegative. A basic variable x_j under which appears a unit vector $\mathbf{e_i}$ is said to be in the basis in the ith row, and its current value is $x_j = b_i$. All nonbasic variables values are considered to be zero.

We are now ready to introduce the simplex algorithm, one of the most used methods to solve a linear program, partly due to its running time which was shown to be, on average, polynomial for real-valued variables [9].

2.1.1 Simplex algorithm

The key idea of the simplex algorithm is to start with a feasible basic vector and do a series of exchanges between basic and nonbasic variables, while increasing the value of the objective function (or at least not decreasing it).

The simplex method can be decomposed in two phases: in phase I the goal is to find a basic feasible vector, in case one does not exist already; in phase II the goal is to actually solve the LP problem, by maximizing the value of the objective function. Thus, phase I is only executed when there are no basic feasible vectors to start directly with phase II. This usually happens when the relations in the LP problem constraints are not only of the type $\leq$, for instance consider the following:

$$\begin{aligned} \text{maximize } & 3x_1 + x_2 \\ \text{s.t. } & 3x_1 + 2x_2 \leq 24 \\ & 4x_1 - x_2 \geq 8 \\ & x_1 - 2x_2 = 0 \\ & x_1, x_2 \geq 0 \end{aligned}$$

To convert the above problem to the standard form, we start by adding a slack variable S_1 in the first constraint and substituting $\leq$ by $=$, in the second constraint we subtract an excess variable E_2, add an artificial variable A_2, and substitute $\geq$ by $=$, in the third constraint we add only an artificial variable A_3, and we get:

$$\begin{aligned} \text{maximize } & 3x_1 + x_2 \\ \text{s.t. } & 3x_1 + 2x_2 + S_1 = 24 \\ & 4x_1 - x_2 - E_2 + A_2 = 8 \\ & x_1 - 2x_2 + A_3 = 0 \\ & x_1, x_2, S_1, E_2, A_2, A_3 \geq 0 \end{aligned}$$

Note that the above LP problem has no basic feasible solutions due to the presence of artificial variables $A_i \in \mathbb{R}_0^+ \, \forall i$. Therefore, to solve such a problem, we start by phase I, which allows us to find a basic feasible vector by driving these variables out of the problem. The need to reduce the artificial variables to zero and eliminate them from the LP problem can be easily understood when considering the following

two situations: 1) suppose $A_i > 0$ is added to a constraint $\mathbf{a_i}.\mathbf{x} = b_i$[1], thus having $\mathbf{a_i}.\mathbf{x} + A_i = b_i$. However, because $A_i > 0$, this implies $\mathbf{a_i}.\mathbf{x} < b_i$, which violates the original constraint $\mathbf{a_i}.\mathbf{x} = b_i$; 2) consider the constraint $\mathbf{a_i}.\mathbf{x} \geq b_i$, which is transformed to $\mathbf{a_i}.\mathbf{x} - E_i + A_i = b_i$, then the columns under the excess variable $E_i \in \mathbb{R}_0^+ \, \forall i$ and artificial variable A_i will be $-\mathbf{e_i}$ and $\mathbf{e_i}$ respectively, i.e., these are linearly dependent and cannot be in the basis at the same time. On the other hand, if $A_i > 0$ then it must be a basic variable and E_i should be nonbasic and equal to zero, implying $\mathbf{a_i}.\mathbf{x} > b_i$ which violates the original constraint. Hence, while at least one artificial variable $A_i > 0$ exists there are no basic feasible solutions for the LP problem.

To eliminate the artificial variables by setting their value to zero, an artificial objective function is defined: Min$w = \sum_i A_i$. Since the value of all artificial variables is nonnegative, the minimum value of w will be 0 when $A_i = 0, \, \forall i \in [1, m]$. Yet, the artificial variables are part of the initial basic feasible vector, and thus need to be expressed in terms of nonbasic variables. Reordering the constraints, in which $\leq$ relations are in the first k rows, and $\geq$ relations correspond to the last $m - k$ rows, in which artificial variables were included, and considering the variables vector $\mathbf{x}$ to contain all variables except the artificial ones, we have:

$$\mathbf{a_i}.\mathbf{x} + A_i = b_i, \, \forall i \in [k+1, m] \Leftrightarrow A_i = b_i - \mathbf{a_i}.\mathbf{x} \, \forall i \in [k+1, m]$$

and we can now write:

$$\text{Min} \sum_{i=k+1}^{m} A_i = \text{Min} \sum_{i=k+1}^{m} b_i - \sum_{i=k+1}^{m} a_i.x$$

and define the parameters $w_j = -\sum_{i=k+1}^{m} a_{ij}$, $j \in [1, n]$ and $w_0 = -\sum_{i=k+1}^{m} b_i$. The corresponding initial simplex tableau is shown in table 2.2, which is a more general case of the simplex tableau presented in table 2.1. These tableaus will be used as a reference in the description of phase I and II of the simplex algorithm, respectively.

A final note on converting an LP problem to the standard form, ready to be solved by the simplex algorithm, is that the right-hand side of the constraints, b_i, $\forall \, i \in [1, m]$, has to be nonnegative. Thus in case there is a negative b_i, the constraint needs to be multiplied by -1 and the relation types modified accordingly.

Before presenting the simplex method, we will define a *pivot operation* and describe how to perform it, since it is a crucial step in the simplex algorithm. A pivot operation aims at replacing a system of equations by an equivalent system, in which a given

[1] $\mathbf{a_i}$ is the vector of coefficients a_{ij} for which i is constant.

x	S	E	A	1	
A	I	0	0	b	original $\leq$ constraints
	0	−I	I		original $\geq$ constraints
		0			original equalities
−c	0			z_0	given objective function
w	0	e	0	w_0	artificial objective function

Table 2.2: The above simplex tableau is used in Phase I of the simplex algorithm, and includes the constraints $\mathbf{Ax} + \mathbf{IS} = \mathbf{b}$, $\mathbf{Ax} - \mathbf{IE} + \mathbf{IA} = \mathbf{b}$, and $\mathbf{Ax} + \mathbf{IA} = \mathbf{b}$, where $\mathbf{A}$ and $\mathbf{E}$ are the vectors of artificial and excess variables, respectively. It also includes both the objective function $\mathbf{cx}$ and the artificial objective function $\sum_{i=k+1}^{m} b_i - \sum_{i=k+1}^{m} a_i.x$.

variable possesses a unit coefficient in one equation and a zero coefficient in all other equations. The steps to perform such an operation are the following:

1. choose an equation r and a variable s, so that you have a pivot term $a_{rs}x_s$, and $a_{rs} \neq 0$;
2. replace the rth equation by the product of itself and $1/a_{rs}$;
3. replace every $i \neq r$ equation by the sum of itself and the product of the rth equation and $-a_{is}$.

The first phase of the simplex algorithm, can be decomposed in four steps:

Step 1 If $w_j \geq 0 \, \forall j$: go to Step 3; else: go to Step 2.

Step 2 Select some $w_s < 0$; the sth column becomes the pivot column. Afterwards select the pivot row r that satisfies $\frac{b_r}{a_{rs}} = min_{i=1,\ldots,n} \left\{ \frac{b_i}{a_{is}:a_{is}>0} \right\}$, where $a_{rs} > 0$ is the pivot. Now do the tableau transformation by performing a pivot operation and go back to Step 1.

Step 3 If all artificial variables are nonbasic, drop the artificial objective function together with all the artificial variables and the respective columns in the tableau and go to the second phase of the simplex algorithm; else: go to Step 4.

Step 4 If $w_0 = 0$, then the current solution is feasible. Select a pivot $a_{rs} \neq 0$ such that $b_r = 0$ and some artificial variable A_i is basic in row r; perform a tableau transformation by doing a pivot operation, and repeat the procedure until all artificial variables are nonbasic. Finally, delete all artificial

variables and the respective columns together with the artificial objective function, and proceed to the second phase of the simplex algorithm. If $w_0 < 0$, the problem has no feasible solution.

After Phase I the LP problem is either unfeasible and the algorithm stops, or has a basic feasible vector and we proceed to Phase II, which is divided in five steps:

Step 1 If $c_j \geq 0 \, \forall j \in [1, n]$: the current solution is optimal and we can stop; else: go to Step 2.

Step 2 Select a nonbasic variable x_s as an entering variable, such that $c_s < 0$ is satisfied; the sth column is the pivot column.

Step 3 If there is an element $a_{is} > 0$, $i \in [1, m]$ in the pivot column go to Step 4; else: there are unbounded "optimal" solutions, stop.

Step 4 Select the rth row as pivot row that satisfies $\frac{b_r}{a_{rs}} = min_{1,\ldots,m} \left\{ \frac{b_i}{a_{is}} : a_i > 0 \right\}$. The element $a_{rs} > 0$ is the pivot, and the variable which is in the basis in the rth row leaves the basis.

Step 5 Do one tableau transformation by performing a pivot operation with a_{rs}, and go back to Step 1.

Special cases in Linear Programming

In linear programming there are a few special cases that may occur and render the LP problem unsolvable. As previously stated, a feasible solution may not exist, either because the constraints are incompatible or the objective function is unbounded. The latter situation is identified in Step 3 of phase II when there is a $c_j \geq 0$ and $a_{is} \leq 0$, whereas the former situation is identified in Step 4 of phase I, when $w_0 < 0 \wedge w_j \geq 0 \, \forall j$ which means there is at least one artificial variable $A_i > 0$ but no pivot column in the artificial objective function. Two other cases that may occur are primal and dual degeneracy. Dual degeneracy occurs when a nonbasic variable has a zero coefficient in the objective function row, this indicates that there is another solution with the same value as the current one. Primal degeneracy occurs when one or more right-hand side values b_i are null, which may lead to *stalling/cycling*, i.e., several bases are generated but the objective function value keeps constant. To circumvent this type of degeneracy and avoid *cycling*, some rules can be used, such as *Bland's Rule*: when the usual choice for selecting a pivot a_{rs} will result in a zero change of the objective value of the basic feasible solution, the choice for incoming/outgoing columns is performed according to:

1. Incoming Column: choose the pivot column $j = s$, for which $c_j < 0$ has the smallest index j.

2. Outgoing Column: choose the outgoing basic column j_r among those eligible for dropping that has the smallest index j.

For illustrative examples on how phases I and II work, please check the book by Eiselt and Sandblom [16], section 3.2.

2.2 Linear programming applied to network inference from RNAi data

Using the linear programming formalism, Knapp and Kaderali [45] developed a model to infer signal transduction networks from RNAi data measured at a single time point.

Approach to network inference

In this model it is assumed that the signaling in the network propagates as an information flow. Taking figure 2.1 as an example, the signal starts at *source* node S and propagates downstream in a deterministic way until it reaches the *end* node E, and a given node a can only influence another node b if there is a path (direct or indirect) from a to b. In a biological setting, the *source* nodes would be receptor proteins, and the *end* nodes transcription factors for instance.

When knocking down a node, that node is considered to be silenced and the signal cannot propagate downstream through the silenced node. For instance, in figure 2.1, if node 1 is silenced then the signal will reach E only through the path $3 \to 4 \to 5$, as the signal cannot pass through 2. Therefore, nodes S, 3, 4, 5, and E are considered to be active, while nodes 1 and 2 are considered to be inactive, node 1 because it was silenced, and node 2 because the signal simply did not reach it. When a node is inactive because a node upstream was knocked down and the signal cannot reach it, we will say it is inactive due to the knockdown's effects. In general, a node is active when it is receiving an activating signal from an upstream node, and inactive when it is either silenced, no signal has reached it, or is inhibited by parent nodes.

Method

More formally, we have $n \in \mathbb{N}$ different nodes, which in our case would be proteins, and $K \in \mathbb{N}$ different knockdown experiments, which may include the silencing of one

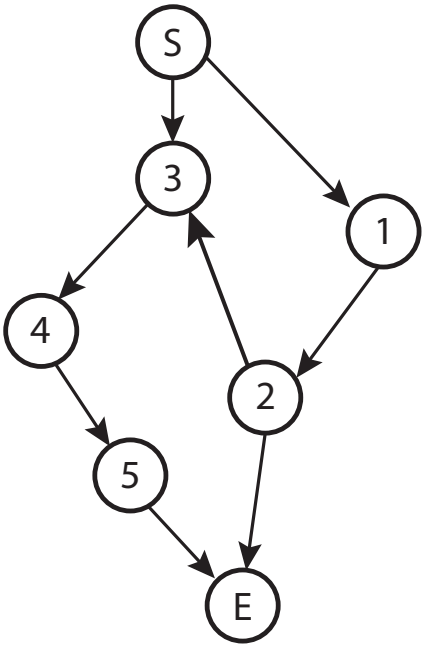


Figure 2.1: The signal propagation in a network is modeled as the depicted information flow. The flow starts at the *source* node S, and ends at node E, the *end* node.

or several nodes in the same experiment. In addition, consider the observation matrix X, which contains the steady-state measurements of each node after a knockdown experiment, $x_{ik} \in \mathbb{R}_0^+ \forall \{i,k\}$, where $i \in [1,\, n]$, $k \in [1,\, K]$, and $x_{ik}(t) = x_{ik}(t+1)$, i.e. the measurements are obtained from a network in steady-state. A node is then classified as active or inactive according to:

$$x_{i,k} \begin{cases} \geq \delta_i & \text{node } i \text{ is assumed to be active} \\ < \delta_i & \text{node } i \text{ is assumed to be inactive} \end{cases} \tag{2.10}$$

where δ_i is a threshold whose value is set by the user, and distinguishes an active state from an inactive one. In order to specify which nodes were silenced, an activation matrix B is also defined, in which an entry $b_{ik} = 0$ means node i under knockdown experiment k was silenced, while $b_{ik} = 1$ means that node i under knockdown experiment k was not silenced.

A signaling network can then be modeled by a graph $G(V, W)$, in which the nodes $v_i \in V$ represent proteins, and the edges $w_{ij} \in W$, $w_{ij} \in \mathbb{R}_0^+ \forall \{i,j\}$ represent an influence of node i over node j. If $w_{ij} > 0$ this influence is positive and means that node i activates node j, if $w_{ij} < 0$ node i inhibits node j, and if $w_{ij} = 0$ there is

no connection.

Another variable to be considered in this approach is the baseline or intrinsic activity of a node, w_i^0, which accounts for cases in which a node has no incoming connections but is still active, as is the case for source nodes. Therefore, the expression of a given node i in each knockdown experiment k is assumed to be the sum of its baseline activity and its incoming connections times the parent nodes values:

$$w_i^0 + \sum_{j \neq i} w_{ji} x_{jk}$$

This value will be called the *activity* of node i. Furthermore, it is assumed that a node i does not influence itself, i.e., there are no self-loops.

Model description

Taking the above considerations into account, a model based on linear programming was developed [45]. In this model, the network is assumed to be sparse, i.e., the number of edges that actually exist is much smaller than the number of possible edges, which is a known characteristic of biological networks in general [48]. Hence, the LP objective function aims at minimizing the edge weights w_{ij}, and the LP problem is formulated as:

$$\underset{\{w_{ji}^+, w_{ji}^-, w_i^0, \xi_l\}}{min} \left(\sum_{i,j} \left(w_{ji}^+ + w_{ji}^- \right) + \sum_i w_i^0 + \frac{1}{\lambda} \sum_l \xi_l \right) \tag{2.11}$$

$$\text{s.t. if } x_{ik} \geq \delta_i \text{ and } b_{ik} = 1: \quad w_i^0 + \sum_{j \neq i} \left(w_{ji}^+ - w_{ji}^- \right) x_{jk} \geq \delta_i \tag{2.12}$$

$$\text{if } x_{ik} < \delta_i \text{ and } b_{ik} = 1: \quad w_i^0 + \sum_{j \neq i} \left(w_{ji}^+ - w_{ji}^- \right) x_{jk} \leq 0 + \xi_l \tag{2.13}$$

$$\text{if } i \in V \setminus S: \quad \sum_{j \in V, j \neq i} \left(w_{ji}^+ + w_{ji}^- \right) \geq \delta_i \tag{2.14}$$

$$\text{if } i \in V \setminus F: \quad \sum_{j \in V, j \neq i} \left(w_{ij}^+ + w_{ij}^- \right) \geq \delta_i \tag{2.15}$$

$$w_{ji}^+,\ w_{ji}^-,\ w_i^0,\ \xi_l \geq 0 \tag{2.16}$$

where the objective function 2.11 minimizes the sum of the edge weights $w_{ji} = w_{ji}^+ - w_{ji}^-$, baseline activities w_i^0 and slack variables ξ_l. w_{ji} is decomposed in positive edges w_{ji}^+ and negative edges w_{ji}^- to satisfy the nonnegativity constraint, while still allowing positive and negative connections. However, in the objective function

the value to minimize is the sum of the absolute values of each component, otherwise there would be a bias towards negative connections. The minimization of this objective function is done while satisfying constraints 2.12-2.16, where constraint 2.16 is the "intrinsic" nonnegativity constraint of the Linear Programming formalism. The first two constraints implement the basic assumption that if a protein i is active, its *activity* should be greater than the respective threshold δ_i, and if protein i is inactive then its *activity* should be null, still, a slack variable $\xi_l \in \mathbb{R}_0^+$ is inserted in constraint 2.13 to account for noise in the data, which can lead to incompatible constraints that render the LP problem unfeasible. The number of slack variables ξ_l introduced is $L = |\{x_{ik} : x_{ik} < \delta_i, \forall i, k\}|$, i.e., it is equal to the number of inactive nodes in the network. The introduction of slack variables is penalized in the objective function according to the value of parameter λ.

The best value for λ is determined using a Cross Validation (CV) technique: either Leave One Out CV (LOOCV) or k-fold CV, depending on the number of nodes in the network, as when the number of nodes increases using LOOCV becomes inefficient in terms of execution time. The procedure to find the best value for λ is the following:

1. calculate the range of possible values for λ, which goes from 0 to $\Xi = L \times \sigma^2(x_{ik})$, where $\sigma^2(x_{ik})$ is the observations x_{ik} variance, and the upper bound for λ is chosen based on the worst case scenario, in which all the introduced slack variables are non-null;

2. start the cross-validation step:

   (a) take out one or more entries of the observation matrix X;

   (b) infer the network for the observation matrix X', obtained by removing at least one entry from X;

   (c) predict the values of the missing entries in X' according to their state being either active or inactive. This state is determined according to the inferred network and which nodes have been silenced plus the respective downstream effects;

   (d) calculate the squared error between the true values of the removed entries from X and their predicted values in the previous step;

3. repeat steps 2a) - 2d) a given number of times and calculate the Mean Squared Error (MSE) of the squared errors calculated in step 2d);

4. after repeating steps 2 - 3 for each possible value of λ, return the set of inferred networks corresponding to the minimum MSE value.

When a protein i in knockdown experiment k is silenced, $b_{ik} = 0$, or its expression value is missing, $x_{ik} = NA$, nothing is said in terms of the LP problem constraints. The reasoning in the first case is that a silenced protein i cannot be influenced by other proteins, so there is no need to calculate the influence of other proteins on it. In the latter case we simply do not know anything about the node in question. In case the observation value x_{jk} of a parent node j in experiment k is missing, $x_{jk} = NA$, two situations may occur: if there are other observations $x_{jk' \neq k}$ for node j available, then the edge w_{ji} will exist or not depending on these, but if there are no other observations available, then the edge w_{ji} will probably be zero in order to minimize the value of the objective function, unless there is a need to have $w_{ji} > 0$ to satisfy constraints 2.14 and 2.15.

Regarding constraints 2.14 and 2.15, these are one way to include prior knowledge about the network to be inferred. If the source nodes of the network, corresponding to receptor proteins in the cell membrane, are known in advance these can be included, and constraint 2.14 will force all other nodes to have at least one incoming connection. On the other hand, if the end nodes are known in advance - these can be, for instance, transcription factors - constraint 2.15 will force all other nodes to have at least one outgoing connection. Another way to include prior knowledge is, when it is known that a certain edge is part of the network to be inferred, to include it as an extra constraint. For instance, if we know that there is a positive edge from node 2 to node 3 we can formulate the respective constraint as follows:

$$w_{32} > 0$$

For the sake of clarity, from now on we will refer to this model as lpNet.

2.3 Linear programming applied to network inference from RNAi time-series data - an extension

As stated previously, the objective of this thesis is to propose an extension for the model developed by Knapp and Kaderali [45], lpNet, in order to use time-series data and, in principle, obtain better results by taking advantage of this type of data. In

this section, the assumptions for such an extension are detailed and the corresponding LP problem is formulated.

Method

The key difference to lpNet is that, instead of having an observation matrix containing the measurements of all nodes at a single time point for each knockdown experiment, we assume having a series of measurements of all nodes along time, while the signal propagates downstream, for each knockdown experiment.

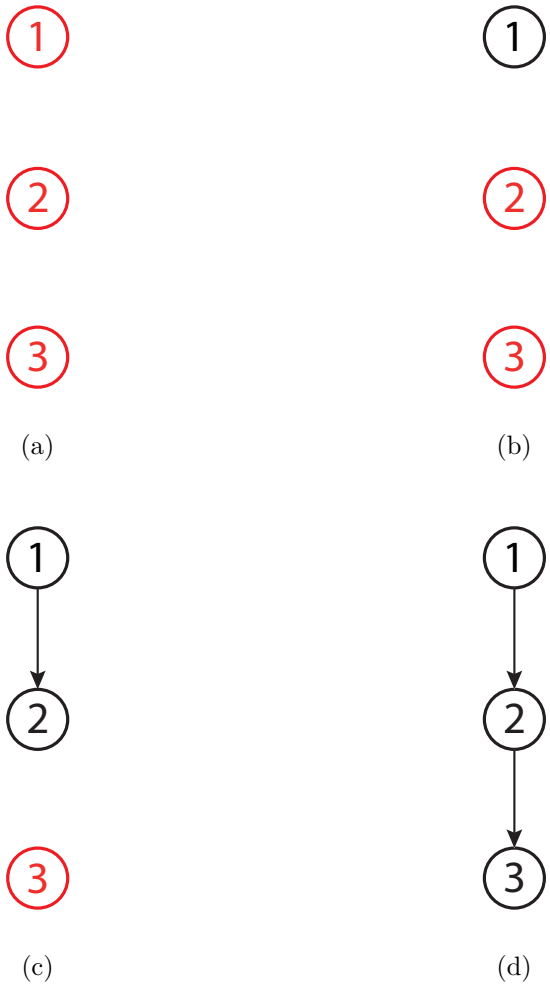


Figure 2.2: Red nodes represent inactive nodes, while black nodes represent active ones. (d) Is the underlying network to be inferred; (a) represents the node states in the network at $t = 1$; (b) represents the node states and active edges at $t = 2$; (c) represents the node states and active edges at $t = 3$; (d) represents the node states and active edges at $t = 4$, when the signal propagation is finished. This network also corresponds to the underlying network.

Consider figure 2.2 and the underlying network in fig. 2.2d. To infer this network using lpNet we only need to perform several knockdown experiments and measure the state of each node when the network is in a steady state. On the other hand, when using the model's extension developed in this thesis to infer the underlying network in fig. 2.2d, we can take advantage of measurements of the expression of each protein at several time-points. Ideally, when no knockdown experiments are performed, we have measurements for each node at each of the time points depicted in figures 2.2a-2.2d, in the first time point $t = 1$ no nodes are active, at $t = 2$ only the source nodes are active, at $t = 3$ both the source nodes and their children are active, and so on until all nodes are active. It is assumed that, once a node j is active at time point t, its children will be active at $t + 1$, i.e., it takes only one time point for the signal to propagate from parent to children and all time steps are equally long. The observation matrix X is now composed of entries $x_{ikt} \in \mathbb{R}_0^+ \, \forall \, \{i, k, t\}$, $i \in [1, n]$, $k \in [1, K]$, $t \in [1, T]$, where T is the number of time-points at which the network nodes values were measured.

Regarding knockdown experiments, these are assumed to be permanent across all time points, therefore the activation matrix B is the same as in section 2.3. The criteria for considering a node as active ($x_{ikt} \geq \delta_i$) or inactive ($x_{ikt} < \delta_i$) are also the same, except that the observation matrix entries x_{ik} are now x_{ikt}. Finally, the network to be inferred is still modeled by a graph $G(V, W)$ with edges $w_{ij} \in \mathbb{R}_0^+ \, \forall \, \{i, j\}$, since only the observations change with time.

Model description

A straightforward way to extend Knapp's model for time-series data is to assume that the expression of protein i at time point t can only be influenced by its parents j at time point $t - 1$, resulting in the following LP problem:

$$\underset{\{w_{ji}^+, w_{ji}^-, w_i^0, \xi_l\}}{min} \left(\sum_{i,j} \left(w_{ji}^+ + w_{ji}^- \right) + \sum_i w_i^0 + \frac{1}{\lambda} \sum_{l,t} \xi_{lt} \right) \tag{2.17}$$

$$\text{s.t. if } x_{ikt} \geq \delta_i \text{ and } b_{ik} = 1: \quad w_i^0 + \sum_{j \neq i} \left(w_{ji}^+ - w_{ji}^- \right) x_{jkt-1} \geq \delta_i \tag{2.18}$$

$$\text{if } x_{ikt} < \delta_i \text{ and } b_{ik} = 1: \; w_i^0 + \sum_{j \neq i} \left(w_{ji}^+ - w_{ji}^- \right) x_{jkt-1} \leq 0 + \xi_{lt} \tag{2.19}$$

$$\text{if } i \in V \setminus S: \sum_{j \in V, j \neq i} \left(w_{ji}^{+} + w_{ji}^{-}\right) \geq \delta_i \tag{2.20}$$

$$\text{if } i \in V \setminus F: \sum_{j \in V, j \neq i} \left(w_{ij}^{+} + w_{ij}^{-}\right) \geq \delta_i \tag{2.21}$$

$$w_{ji}^{+},\ w_{ji}^{-},\ w_i^0,\ \xi_{lt} \quad \geq 0 \tag{2.22}$$

where the only differences from lpNet is the substitution of ξ_l by ξ_{lt}, x_{ik} by x_{ikt}, and x_{jk} by x_{jkt-1}, reflecting that the observation x_{ikt} can only be influenced by the expression of the parent proteins at the previous time point, x_{jkt-1}.

In terms of calculating the best value for λ, the procedure described in section 2.3 is followed, except for steps 1 and 2a) that suffer slight changes. In step 1 the range of possible values for λ now goes from 0 to $\Xi = L \times \sigma^2(x_{ikt})$, and in step 2a) the entries to be removed from the observation matrix X cannot be taken out from the first time point, as the prediction of a given entry depends on entries from the previous time point. The *activity* of a node is now defined as $w_i^0 + \sum_{j \neq i} w_{ji} x_{jkt-1}$.

This model will be referred to as lpNet-dyn from here on.

Analyzing the way constraints 2.18 and 2.19 are formulated, these allow a node j that is inactive at $t-1$ to influence its children and contribute to their *activity*. Yet, if we consider a biological scenario in which the signal inside the cell propagates downstream by means of phosphorylation: one protein phosphorylates another and so on. Then it would make no sense for a protein that has not been phosphorylated (and thus is inactive), to be already phosphorylating other proteins in the signaling cascade, i.e. contributing to their *activity*.

Additionally, by assuming that an inactive parent node at $t-1$ can contribute to its children *activity* at t, the above LP problem might miss some edges. For instance, consider having a dataset with 7 time points, an edge w_{ji} that is part of the underlying network, and node j being the only parent of node i. Yet, node j is active only from $t = 5$ onwards, thus activating node i at $t = 6$, which had been inactive until $t = 5$. Hence, at $t = 2, 3, 4, 5$, constraint 2.19 regarding node 2 should be satisfied, however, both w_{ji} and x_{jkt-1} are greater than zero and slack variables ξ_{lt} need to be introduced, meaning the addition of an extra weight in the objective function. Taking into account that the objective function value should be minimized, it may happen that edge w_{ji} is not inferred due to the addition of four extra slack variables (one per time point in which node i is inactive) that it would cost. One

way to avoid such a situation is to set the value of x_{jkt-1} to zero when node j is inactive and, assuming $w_i^0 = 0$, constraint 2.19 is satisfied without the need to add slack variables.

Therefore, a second extension was developed, in order to prevent an inactive node from influencing other nodes: only active nodes can influence their children and contribute to their *activity*. The way to achieve it, is to set the parent node value x_{jkt-1} to zero when it is less than the respective threshold δ_j, hence resulting in the LP problem:

$$\min_{\{w_{ji}^+, w_{ji}^-, w_i^0, \xi_l\}} \left(\sum_{i,j} \left(w_{ji}^+ + w_{ji}^- \right) + \sum_i w_i^0 + \frac{1}{\lambda} \sum_{l,t} \xi_{l,t} \right)$$

$$\text{s.t. if } x_{ikt} \geq \delta_i,\ b_{ik} = 1, \text{ and } x_{jkt-1} \geq \delta_j: \quad w_i^0 + \sum_{j \neq i} \left(w_{ji}^+ - w_{ji}^- \right) x_{jkt-1} \geq \delta_i \tag{2.23}$$

$$\text{if } x_{ikt} \geq \delta_i,\ b_{ik} = 1, \text{ and } x_{jkt-1} < \delta_j: \quad w_i^0 + \sum_{j \neq i} \left(w_{ji}^+ - w_{ji}^- \right) \cdot 0 \geq \delta_i \tag{2.24}$$

$$\text{if } x_{ikt} < \delta_i,\ b_{ik} = 1, \text{ and } x_{jkt-1} \geq \delta_j: \quad w_i^0 + \sum_{j \neq i} \left(w_{ji}^+ - w_{ji}^- \right) x_{jkt-1} \leq 0 + \xi_{lt} \tag{2.25}$$

$$\text{if } x_{ikt} < \delta_i,\ b_{ik} = 1, \text{ and } x_{jkt-1} < \delta_j: \quad w_i^0 + \sum_{j \neq i} \left(w_{ji}^+ - w_{ji}^- \right) \cdot 0 \leq 0 + \xi_{lt} \tag{2.26}$$

The last three constraints are omitted because they do not change, prior knowledge can be included in the same way as for the original lpNet model.

Assuming that the perturbation data to be used comes from RNAi experiments, in which the knocked down protein has an expression value lower than its threshold, there is no need to check the value of b_{ik}. However, there are perturbation techniques in which the protein itself is still active, $x_{jkt} \geq \delta_j$, but cannot influence other proteins, e.g. by phosphorylation, in this case there is a need to check the value of b_{ik}. In this way, less non-null edges should be missed.

From here on, this model will be referred to as lpNet-dyn2 and, when referring to both lpNet-dyn and lpNet-dyn2 at once, we will use lpNet-dyn/2.

Chapter 3

Methods

In this chapter we start by describing how to predict the value of the entries removed in the Cross-Validation step of our model. Afterwards, we explain how the networks used for tests with simulated data are generated, as well as how the data is simulated, and then describe the experimental datasets used and how the necessary model parameters are calculated. Finally, we describe how the results are processed and evaluated, the experimental setup where the models were executed, and the models execution settings.

3.1 Prediction of removed entries in Cross-Validation step

In the previous chapter we described how the parameter λ in the LP problem objective function of the lpNet and lpNet-dyn/2 models is calculated. This involves using a cross-validation method where, in the end, the entries removed from the observation matrix are predicted either as active or inactive. Here we describe the rules used to predict the state of such an entry.

- A node i in knockdown experiment k and time point t is considered to be active, when:
  - it is a source node and its *activity*, which is only due to its baseline activity w_i^0, is greater than or equal to the threshold δ_i;
  - all of its parent nodes observations at $t-1$ are not missing and its *activity* is greater than or equal to the threshold δ_i;

  - one or more parent nodes values at $t-1$ are missing but none of them is connected by a negative edge w_{ji}^{-} - or it is connected by a negative edge but the parent is inactive due to a knockdown - and its *activity* is still greater than or equal to the threshold δ_i.

- A node i in knockdown experiment k and time point t is considered as inactive when:

  - it was directly silenced in the respective knockdown experiment or is inactive due to the experiment's effects;
  - it is a source node and its *activity*, which is only due to its baseline activity w_i^0, is less than its threshold δ_i;
  - all of its parent nodes observations are not missing and its *activity* is less than the threshold δ_i.

- A node i value in knockdown experiment k and time point t is not predicted and thus set to NA, when:

  - the connections to its parents are all positive, but at least one of the values of its parents is missing and its *activity* is less than the threshold δ_i;
  - the value of at least one parent connected by a negative edge is missing and the parent is not inactive due to a knockdown.

The value of a node predicted either as active or inactive is generated from a normal distribution $\mathcal{N}(\mu, \sigma)$, where μ is calculated from the data and has a different value according to whether the node is predicted as active, μ_{act}, or inactive, μ_{inact}, and the value of σ it is usually a small percentage of μ.

3.2 Network generation

Two groups of network topologies, each one generated in a different way, were used to evaluate the performance of lpNet-dyn/2. The first group comprises networks with five nodes only, and each of these networks contains one or more motifs common in biological networks. By using these networks, we aim at understanding how well the model can infer the given motifs. The second group of networks contains ten nodes and was generated by using the function `signalnetwork` from the ddepn R package [6, 7]. Given a number of nodes and a number of source nodes, this function

randomly chooses which nodes will be the source nodes and then randomly selects their children nodes, which will be connected by activating edges. The children nodes become the parent nodes, and new children nodes are randomly chosen again, which will be connected by activating edges. The procedure is repeated until the network is fully connected, and in the end a given percentage of the edges sampled so far will be added as inhibiting edges.

Regarding the networks generated with the ddepn R package, two sets of ten ten-node networks were generated, one set possesses only activating edges and the other set has 20% of inhibiting edges. These networks possess 2 to 3 source nodes, and were chosen in such a way as to include some of the motifs mentioned in chapter 1. Moreover, these two groups of networks were used to test the general performance of our model and compare it to the original lpNet model and to the DDEPN model. The networks structure is presented in section A.1.

3.3 Data Simulation

In this section, we describe how the data corresponding to the network's nodes states was simulated for lpNet-dyn/2, lpNet,and DDEPN.

lpNet-dyn/2

To simulate the data used to test our model with simulated networks, we consider two types of nodes: active nodes and inactive nodes, i.e. nodes whose expression x_{ikt} is equal or greater than its threshold value δ_i, and nodes whose expression x_{ikt} is less than its threshold δ_i, respectively. All nodes observation values are generated from normal distributions, active nodes from the distribution $\mathcal{N}(\mu_{act}, \sigma) = \mathcal{N}(0.95, \sigma)$, and inactive nodes from $\mathcal{N}(\mu_{inact}, \sigma) = \mathcal{N}(0.56, \sigma)$, where $\sigma = \{0.01, 0.05, 0.2\}$ when there are knockdown experiments and $\sigma = \{0.01, 0.05, 0.2, 0.4, 0.7\}$ for simulations with no knockdown experiments. This restriction is made due to increased execution time of simulations that include knockdown experiments. The value of each node in each knockdown experiment and time point, x_{ikt}, is generated three times and then the mean value is calculated and set as the value of x_{ikt}. Regarding δ_i value, it is also generated from a normal distribution $\mathcal{N}(\mu, \sigma)$, where σ has the same value as the one used to generate the nodes observations, and μ is the mean value of μ_{act} and μ_{inact}. Unless stated otherwise, the value used for σ is 0.01

To generate this data for time-series simulations, we consider all nodes to be inactive at the first time point $t = 1$, the source nodes are then active at $t = 2$, and

then at each time step the signal will propagate downstream, from the active nodes to their children as depicted in figure 3.1. The signal propagation stops either when the nodes have no more children, or when all edges present in the underlying network have been added. The last condition is imposed so that the function does not enter in an infinite loop in case there are feedback loops present in the network. For the network in fig. 3.1 the resulting vectors of the nodes state at each time point would be:

$$t=1:\begin{bmatrix}0\\0\\0\end{bmatrix};\ t=2:\begin{bmatrix}1\\0\\0\end{bmatrix};\ t=3:\begin{bmatrix}1\\1\\0\end{bmatrix};\ t=4:\begin{bmatrix}1\\1\\1\end{bmatrix}$$

where an active node is represented by 1 in the vector of node states, and an inactive node is represented by 0. The first entry in each vector corresponds to the state of node one, the second entry to the state of node two, and so on. The number of time points T for each network is equal to the number of steps it takes for the signal to stop propagating. Therefore, the data for different networks may include different numbers of time points T.

When negative edges are part of the network, a node will be considered as active when the number of incoming positive edges from active parents is greater than the number of incoming inhibiting edges from active parents; if this number is equal or less than that, then the node will be considered as inactive. To exemplify the signal propagation in this case consider fig. 3.2. The resulting node states at each time point are then:

$$t=1:\begin{bmatrix}0\\0\\0\\0\\0\\0\\0\end{bmatrix};\ t=2:\begin{bmatrix}1\\0\\0\\1\\0\\0\\0\end{bmatrix};\ t=3:\begin{bmatrix}1\\1\\0\\1\\1\\0\\0\end{bmatrix};\ t=4:\begin{bmatrix}1\\0\\1\\1\\1\\1\\0\end{bmatrix};\ t=5:\begin{bmatrix}1\\0\\0\\1\\1\\1\\1\end{bmatrix}$$

Note that when node 5 inhibits node 2 at $t=4$, at $t=5$ node 3 is also inactive, since it has no active parents connected by activating edges, i.e. it is receiving no signal.

Finally, to generate this time-series data, one also needs to take into account the knockdown experiments and their influence on downstream nodes. To ensure that all

knockdown experiments have the same number of time points, which should also be equal to the number of time points when no knockdown experiments are included, the number of time points is calculated in advance by propagating the signal in a network with no silenced nodes. Only then are the node states generated for each knockdown experiment. In case the signal stops propagating at $t < T$, the state of all nodes keeps constant for all time points $t < t' \leq T$. Similarly, if the signal propagation only stops at $t > T$, it is forced to stop at T. The reasoning here is that in an experimental real scenario the measurements are taken at fixed time points, independently of whether the signal is still propagating or not. To illustrate this case consider figures 3.3 and 3.4. The vectors of the nodes state evolution along time are:

$$t = 1 : \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}; t = 2 : \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}; t = 3 : \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}; t = 4 : \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

and

$$t = 1 : \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}; t = 2 : \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}; t = 3 : \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \\ 0 \end{bmatrix}; t = 4 : \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

Note that in the second example, fig 3.4, when node 1 is silenced it would take 5 time points until the signal propagation stopped. Yet, when node 1 is active it only takes 4 time points for the signal to stop propagating, because node 5 is never active and thus the signal cannot reach node 7.

lpNet

To generate the data for Knapp's lpNet model, the method described in the previous paragraph was also used, however, only the data from the last time point was considered. The reasoning here is that the network is assumed to be in the closest state to steady-state at the last time point.

DDEPN

The data used in DDEPN was exactly the same that is used for lpNet-dyn/lpNet-

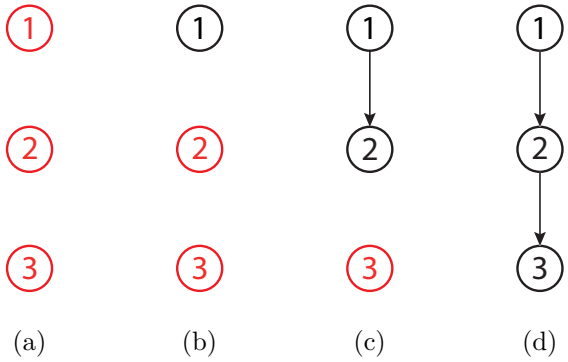


Figure 3.1: Red nodes represent inactive nodes while black represents active nodes. (a) Depicts the node states at $t = 1$, all nodes inactive; (b) node states at $t = 2$, the source node is active; (c) node states at $t = 3$, nodes 1 has activated node 2; (d) node states at $t = 4$, all nodes are active.

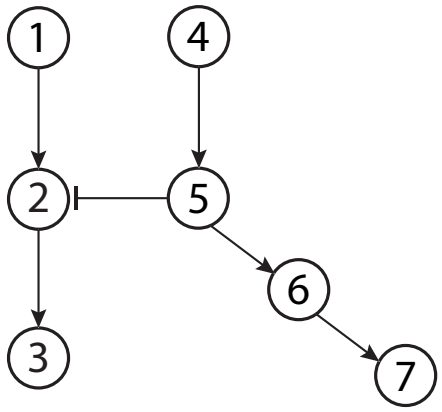


Figure 3.2: Network with two source nodes, in which node 5 inhibits node 2 and thus node 3 is active only at $t = 4$.

Figure 3.3: Three-node network in which node 2 is silenced. Hence, node 3 is never active.

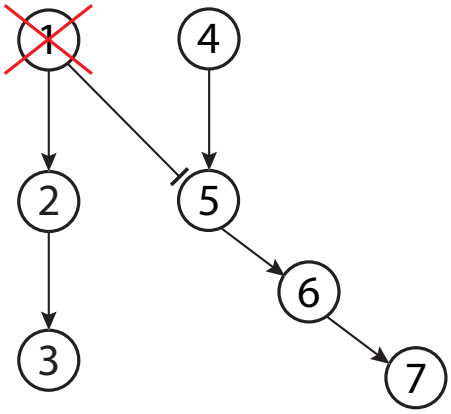


Figure 3.4: Seven-node network in which node 1 is silenced, allowing the signal to reach node 7.

dyn2, so that the results comparison can be done in the same conditions. However, due to time constraints and since the DDEPN's running time is very long, only one value for the standard deviation of the normal distributions was used, $\sigma = 0.01$.

Unless stated otherwise, the above described methods are the ones used to generate all simulated data, and we perform tests both with and without knockdown experiments. In the latter case $K = 1$ and no nodes are silenced, and in the former case $K = n + 1$ knockdown experiments are performed: each node is silenced in a different experiment and there is also one experiment in which no nodes are silenced. n is the number of nodes in the network.

Although our model was built to take advantage of time-series data with knockdown experiments, it is also interesting to test its behavior when no knockdown experiments are included. In addition, we use one experimental dataset which has no knockdown experiments, therefore it is important to understand the impact of using time-series data with no knockdown experiments.

The motivation to use artificial network with simulated data, is to be able to quantify and compare the models performance, since when using experimental data no gold standard network exists, only reference networks.

3.4 Experimental Data

3.4.1 ERBB regulated G1/S cell cycle transition

In order to try to understand the influence of ERRB signaling in the human cell cycle G1/S transition, Froehlich *et al* [29] have studied 16 proteins that participate in this process. Therefore, several RNAi knockdown experiments were performed using chemically synthesized siRNAs: 13 single RNAi knockdowns (ERBB1, IGF1R, ER-alpha, pAKT1, pERK1/2, MYC, Cyclin D1, p27, p21, Cyclin E1, CDK6, CDK4, CDK2) and 3 double RNAi knockdowns (ERBB1+ERBB2, ERBB2+ERBB3, ERBB1+ERBB3), as well as an experiment with MOCK transfected cells to be used as a negative control. After siRNA transfection, the cells were stimulated with EGF for 12h, and the expression of 10 proteins (ERBB1, ERBB2, pAKT1, pERK1/2, Cyclin D1, p27, p21, CDK4, CDK2 and pRB1) was quantified by RPPA measurements, before and after EGF stimulation, in 4 technical and 3 biological replication.The data was then normalized using quantile integration [8]. For further details please see [29, 74].

To be able to use this dataset with our model and infer the respective network, we summarized the measurements replicates by taking their average value, and calculated a threshold δ_i for each gene: $\delta_i = \underset{k,t}{max}\{x_{ikt} : b_{ik} = 0\} + \epsilon$. The key idea here is that we know the protein expression values when these are silenced (and thus inactive), and assume that all values less or equal to this correspond to inactive proteins, while values greater than this correspond to active proteins. However, since in our model formulation we assume that an expression value greater or equal to δ corresponds to an active state, a small ϵ was added to the δ_i calculation, so that the inequalities are still valid and the value of δ_i is correct. In this case $\epsilon = 10^{-5}$, and the data was previously rounded to 5 decimal places. Having determined δ_i, the values for the means of the normal distributions $\mathcal{N}(\mu, \sigma)$ used to predict the value of a node in the CV step, $\mu_{act,i}$ and $\mu_{inact,i}$, were set as $\mu_{act,i} = \underset{k,t}{average}\{x_{ikt} : x_{ikt} \geq \delta_i\}$ and $\mu_{inact,i} = \underset{k,t}{average}\{x_{ikt} : x_{ikt} < \delta_i,\}$ while the standard deviation σ was set as $\sigma = 0.01$.

For a matter of clarity we will call this dataset the ERBB G1/S dataset from now on.

The dataset is available as part of the nem bioconductor package [**24**].

3.4.2 ERBB signaling cascade

Bender *et al* [6] have studied 16 phosphoproteins related to ERBB signaling in the human breast cancer cell line HCC1954, which overexpresses ERBB2. The list of studied proteins and corresponding phosphorylation sites is shown in table 3.1. Three experiments were performed: stimulation with EGF only, stimulation with HRG only, and stimulation with both EGF and HRG. RPPAs were used to measure the proteins expression 0, 12, 16, 20, 30, 40, 50, and 60 min after stimulation for each experiment in 5 biological and 3 technical replicates. Fast Green FCF dye was used for sample normalization, so that different protein concentrations in each array spot are accounted for. Plus, to remove systematic shifts in the signal intensities, replicate time courses were centered around their common mean. For more details on the experimental setup please see ref. [6].

To use this dataset with our model and infer the respective network, all the replicate measurements were averaged into a single measurement per protein per time point per stimulation experiment, x_{ikt}. In addition, the data was normalized so that the range, and consequently the number, of λ values to use is smaller and the model running time is reduced. Thus, the expression value of each protein is set as $x_{ikt}/\underset{t}{max}\{x_{ikt}\}$, where $\underset{t}{max}\{x_{ikt}\}$ is calculated after removing the outliers. Each stimulation experiment is considered as a knockdown experiment in which no nodes are silenced, the idea is to account for the different effects of each stimulus on the network, which are unknown. To calculate the threshold value, δ, we consider a different threshold per node per stimuli experiment, δ_{ik}, since we assume that different stimuli can lead to different activation levels, i.e., a node may have different expression values for different stimuli and still being in the same state. Thus, we set $\delta_{ik} = x_{ikt=0} + \epsilon$ for all nodes except the nodes corresponding to the receptors ERBB1-4, and $\delta_{ik} = x_{ikt=0}$ for ERBB1-4. The reasoning here is that we assume the receptor proteins to be active at the first time point, $t = 0$, while all other proteins are inactive, and thus add $\epsilon = 10^{-5}$ to the value of the latter proteins at $t = 0$, after rounding the data to 5 decimal places. Regarding the parameters of the normal distribution $\mathcal{N}(\mu, \sigma)$ used to predict the value of a node in the CV step, we have: $\mu_{act,i,k} = \underset{t}{average}\{x_{ikt} : x_{ikt} \geq \delta_i\}$, $\mu_{inact,i,k} = \underset{t}{average}\{x_{ikt} : x_{ikt} < \delta_i\}$, and $\sigma = 0.01$.

For the sake of clarity we will call this dataset the HCC1954 dataset from now on.

The dataset is available as part of the ddepn R package.

Protein	Phosp. sites	Protein	Phosp. sites	Protein	Phosp. sites
ERBB1	Y1068	AKT	S473	SRC	Y416
ERBB2	Y1112	GSK3	Y279, Y216	MEK1/2	S217, S221
ERBB3	Y1289	p38	T180, Y182	p70S6K	T389
ERBB4	Y1162	PKCα	S657, Y658	PLCγ	S1248
ERK1/2	T202, Y204	mTOR	S2448	PDK1	S241
PRAS	T246				

Table 3.1: Proteins and repective phosphorylation sites measured in the study of the ERBB signaling cascade in a human breast cancer cell line.

3.5 Results processing and evaluation

In this section, we describe how the obtained results from each model assessed in this thesis are processed and evaluated.

3.5.1 lpNet and lpNet-dyn/2

Each time lpNet-dyn/2 is executed, a given number of networks are inferred for each possible value of the λ parameter. After determining the best value for λ, i.e. the value that minimizes the MSE between the predicted values for the removed entries in the observation matrix and their true values, the set of corresponding inferred networks is returned. These networks need to be summed up into a single network. Since the value of each edge is continuous, this is done by calculating the median and the Median Absolute Deviation (MAD) values of each edge inferred in all networks. Only edges whose median value is greater than its MAD are included in the final network. In this way, only edges that are inferred in most of the networks with similar values w_{ji} are included in the final network, avoiding the inclusion of spuriously inferred edges. The final network is then compared to a true network – if available – and sensitivity (SN), specificity (SP), and precision (PR) values are calculated.

3.5.2 DDEPN

When using the DDEPN model a set of networks is produced in the end, whose values are in the set $\{0, 1, 2\}$, where 1 stands for an activating edge, 2 for an inhibiting edge, and 0 for a null edge. These networks also need to be summed up into a final network, however, because the value of an edge is discrete, the number of times an edge with a given value is inferred is divided by the total number of inferred networks,

thus producing a probability of inferring the given edge. Only edges whose probability of being inferred is greater than 0.5 are included in the final network. If the probability for an edge to be inhibiting is equal to the probability of it being activating, the edge is considered to be null. This final network is then compared to a true network – if available – and sensitivity, specificity, and precision values are calculated.

Since both the models developed in this thesis and the models to which comparisons are established can infer positive, negative, and null edges, this classification problem is a three class problem. Hence, an edge in the final network is classified, according to table 3.2, as false positive (FP): an edge incorrectly predicted as positive or negative, true positive (TP): a positive or negative edge predicted correctly, false negative (FN): a null edge predicted as positive or negative, or true negative (TN): a null edge predicted as such.

		Predicted		
		Positive edge	Negative edge	Null edge
Actual	Positive edge	TP	FP	FN
	Negative edge	FP	TP	FN
	Null edge	FP	FP	TN

Table 3.2: Confusion matrix used to map a three class classification problem onto a two class problem.

Simulated data

We use either boxplots or median ± MAD values to represent the results from simulated datasets in terms of SN, SP, and PR because the distribution of these values is usually highly skewed. Therefore, a boxplot or median ± MAD values are able to better represent these results. Furthermore, we choose to evaluate all results in terms of sensitivity, specificity, and precision values, instead of using Receiver Operating Characteristic and Precision to Recall curves, because the former measures provide a more direct insight into the model's weaknesses, e.g. whether it infers too many false positive or too many false negative edges.

Each underlying network is inferred at least 30 times in order to obtain reliable values, and for each time a new dataset is generated according to the same parameters, μ and σ. Afterwards, SN, SP, and PR values are calculated for the resulting network of each run.

Our results are also compared against random prediction. Given an adjacency matrix that represents the true network, random prediction was performed by permuting the entries of each column and row individually a 100 times, after which the result is evaluated. This step is repeated the same amount of times that the network is inferred by lpNet-dyn/2.

Experimental data

To analyze the results for each experimental dataset we use heatplots to represent each inferred edge, and then analyze each one individually by checking whether it is also found in the literature. Only edges whose median value is greater than its MAD are considered to exist. Additionally, we used Ingenuity IPA [1] to assemble a network constituted by the dataset's proteins. This network is used as a true network to evaluate the results obtained with our model in terms of SP, SN, PR and accuracy. To assemble the true network with Ingenuity IPA we considered only experimentally observed direct connections in human and in vitro samples. The assembled networks are presented in figures A.22 and A.23.

3.6 Experimental Setup and Execution Settings

The model was implemented in R [85], and executed using R 2.15.3 on an Intel® Xeon X5460 @ 3.16GHz with 2×6MB of L2 cache and 32GB or RAM. For simulated data, each model was executed at least 30 times in order to obtain reliable results and, when possible, a 100 times. In each time a new observation matrix was generated from the same parameters. These runs were usually executed in different CPU cores, not necessarily belonging to the same cluster node. The R packages and respective versions used in the execution of the lpNet and lpNet-dyn/2 models are listed in table 3.3a, and the R packages used for DDEPN execution are listed in table 3.3b.

To solve the LP problems stated in sections 2.2-2.3 the R package lpSolve v5.6.6 [**17**] was used, whereas the ddepn R package version 2.1.2 was used to run the respective model.

For the lpNet and lpNet-dyn/2 models the CV method used was the LOOCV for the five-node networks, and k-fold CV for the ten-node networks with $k = 10$. The CV step was repeated a 100 times for the LOOCV and 5 times for k-fold CV, resulting in 50 networks inferred for each possible value of λ for k-fold CV and a number of networks dependent on the observation matrix dimension for the LOOCV (more precisely: equal to the number of entries for all time points except the first

Package name	Version
lpSolve	5.6.6

(a)

Package name	Version
gam	1.08
coda	0.16-1
gtools	2.7.1
bitops	1.0-5
caTools	1.13
gplots	2.11.0.1
igraph0	0.5.7
ddepn	2.1.2
MASS	7.3.23

(b)

Table 3.3: R packages, and respective versions, used to execute (a) lpNet and lpNet-dyn/2] and (b) DDEPN.

one). The value for λ ranged from 0 to $\Xi = L \times \sigma^2 (x_{ik})$ with intervals of 0.05. For the experimental datasets, LOOCV was used for the ERBB G1/S dataset which had few observations, and k-fold CV for the HCC1954 dataset with $k = 10$. For both datasets the CV step was repeated a 100 times, and the interval between each successive value of λ was set to 0.01.

Regarding DDEPN's execution, the following parameters for the genetic algorithm were used: a population of 500 networks, $p = 500$, a crossover (selection) rate of $q(1-q)$, where $q = 0.3$, a mutation rate of 0.8, $m = 0.8$, and the maximum number of iterations was set to a 1000. These parameters were chosen according to [6]. To run the simulations with no knockdown experiments, only one stimuli experiment is considered, in which one stimuli node is added to the true network. This node is connected to all source nodes by an activating edge. For simulations with $K = n + 1$ knockdown experiments, K stimuli experiments are considered, one in which the network is stimulated by a single node connected by a positive edge to all source nodes, and an experiment per knockdown in which the network is stimulated by: the stimuli node again, and further perturbed by an inhibiting node, which is connected to its target – the node to be silenced – by an inhibiting edge. Hence, when K knockdown experiments are performed K stimuli/inhibiting nodes and respective edges are added to the true network. However, when assessing the network inference results, we only consider the edges inferred among the original network nodes, and not the edges inferred from the stimuli/inhibiting nodes to their targets, as the latter nodes and edges do not exist in our model.

Chapter 4

Results and discussion

In this chapter we present and discuss the results obtained with the models lpNet-dyn and lpNet-dyn2, both when using artificial networks with simulated data, generated as described in sections 3.2 and 3.3, and experimental datasets, in particular the ERBB G1/S and HCC1954 datasets. Furthermore, we compare our results to the ones obtained with lpNet and DDEPN.

4.1 Simulated data

4.1.1 Prediction of Motifs in Biological Networks

In this section, we evaluate the performance of lpNet-dyn/2 when inferring the 5-node networks from figures 4.1a-4.1f, which contain motifs usually found in biological networks [78, 81]. To assess this, we present, for each model, whether the network non-null edges are correctly inferred or not, and which edges are incorrectly inferred (false positives). Tests were performed with a single knockdown experiment ($K = 1$), in which no nodes are silenced, and with six knockdown experiments ($K = 6$), in which a different node is silenced in each experiment and in the last one no nodes are silenced. When knockdown experiments are performed, we compare the results obtained with lpNet-dyn/2 and lpNet. Each model was executed a 100 times, resulting in a 100 inferred networks, which are summed up by considering an edge w_{ji} to exist if its median value is greater than its MAD. In the end, we also discuss the motivation for this criterion.

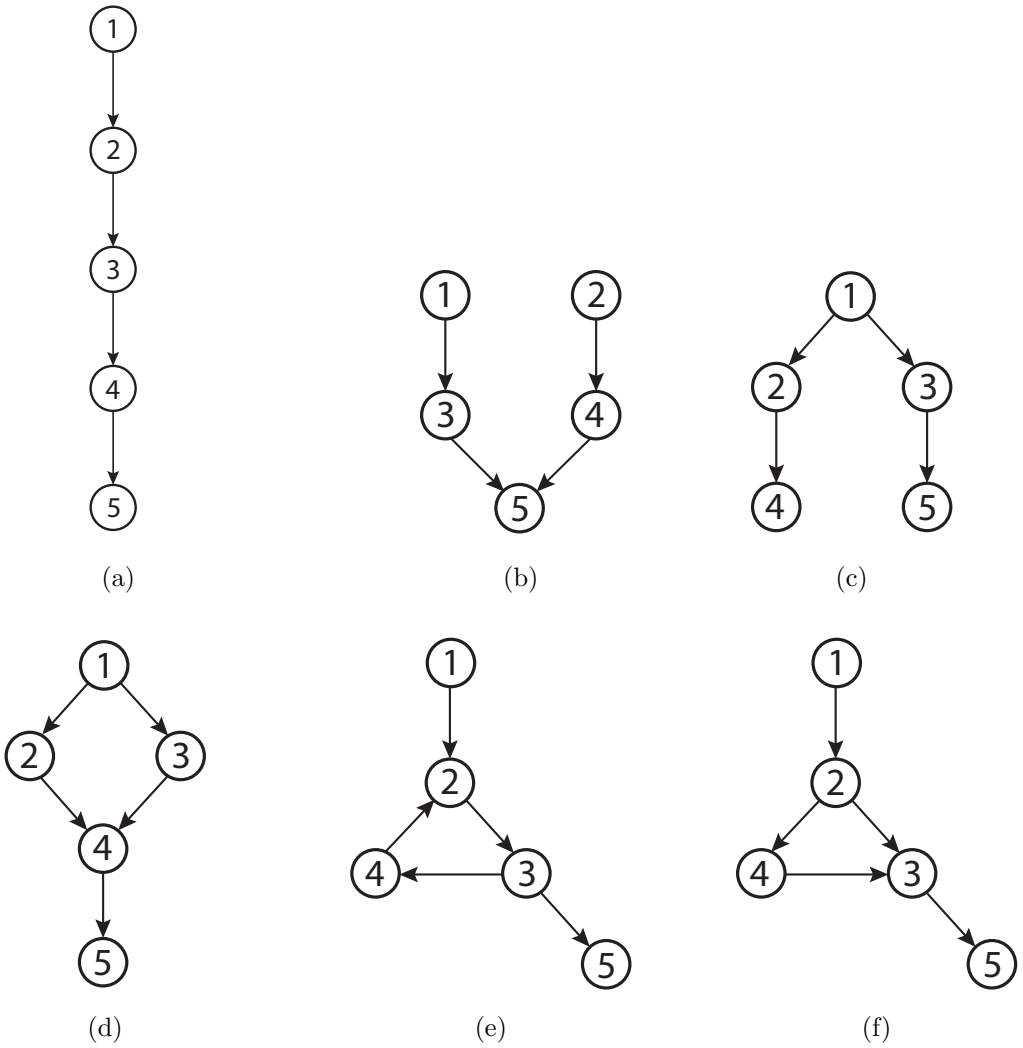


Figure 4.1: In this figure 7 networks containing motifs found in biological networks are shown: (a) cascade, each node activates the next one sequentially; (b) fan-in, both nodes 3 and 4 activate node 5; (c) fan-out, node 1 activates both nodes 2 and 3; (d) fan-in + fan-out, node 1 activates both nodes 2 and 3, which in turn activate node 4; (e) feedback loop, node 2 activates node 3, node 3 activates node 4, and node 4 activates node 2 again; (f) feed-forward loop, node 2 activates both nodes 3 and 4, and node 4 activates node 3 again.

Cascade

The first five-node network to be inferred is a linear signaling cascade, fig. 4.1a, where each node activates the next one sequentially. The results are presented in table 4.1. lpNet-dyn2 always infers all edges correctly and no false positives, while lpNet-dyn needs 6 knockdown experiments to infer all edges correctly, otherwise it misses one edge and infers a false edge. The original lpNet with 6 knockdown experiments

misses one edge, $3 \rightarrow 4$.

The network inferred by lpNet-dyn for $K = 1$ explains the data as well as the true network. The reason it is inferred is that, by adding edge $1 \dashv 5$, the sum of the slack variables ξ_{lt} to be added to the objective function is smaller, and the constraints $w_5^0 + \sum_{j \neq 5} w_{j5} x_{jkt-1} \leq 0 + \xi_{lt}$ are still satisfied. These constraints need to be satisfied 4 out of 5 time points, when the sum $\sum_{j \neq 5} w_{j5} x_{jkt-1}$ is smaller than at the last time point, when all nodes $j \neq 5$ are active. Hence, it is less expensive to infer an extra edge $1 \dashv 5$ than to add the necessary slack variables to the objective function, even if edge w_{45} needs to have an higher value to satisfy $w_5^0 + \sum_{j \neq 5} w_{j5} x_{jkt=4} \geq \delta_5$. This situation should be avoided by setting λ value high enough, however, for this simulation its maximum value is 0.7, turning the addition of slack variables more expensive than inferring extra edges.

	$K = 1$		$K = 6$		
Edges	lpNet-dyn	lpNet-dyn2	lpNet-dyn	lpNet-dyn2	lpNet
$1 \rightarrow 2$	YES	YES	YES	YES	YES
$2 \rightarrow 3$	YES	YES	YES	YES	YES
$3 \rightarrow 4$	NO	YES	YES	YES	NO
$4 \rightarrow 5$	YES	YES	YES	YES	YES
FP	$1 \dashv 5$	-	-	-	-

Table 4.1: The above table shows for each model, without ($K = 1$) and with knock-down experiments ($K = 6$), which edges are correctly inferred for a linear signaling cascade, fig. 4.1a. Inferred false positive (FP) edges are shown in the last line, where $\dashv$ refers to an inhibiting edge.

Fan-in

The network from figure 4.1b contains a fan-in motif: two nodes activate a third node at the same time. Yet, it also contains another structure that is hard to infer: nodes 1 and 3 activate nodes 4 and 5, respectively, at the same time point. This structure can be explained by 4 different sets of edges: 1) $1 \rightarrow 4$, $1 \rightarrow 5$, 2) $3 \rightarrow 4$, $3 \rightarrow 5$, 3) $1 \rightarrow 4$, $3 \rightarrow 5$, 4) $1 \rightarrow 5$, $3 \rightarrow 4$; thus the difficulty in inferring such a structure. Due to this situation no edges are inferred for $K = 1$, see table 4.2, the LP problem constraints being explained by setting $w_i^0 \geq \delta_i$. Only for $K = 6$, can lpNet-dyn/2 infer all edges correctly and no false positive edges, while the original model lpNet fails to infer both edges ending in node 5.

	$K = 1$		$K = 6$		
Edges	lpNet-dyn	lpNet-dyn2	lpNet-dyn	lpNet-dyn2	lpNet
$1 \rightarrow 3$	NO	NO	YES	YES	YES
$2 \rightarrow 4$	NO	NO	YES	YES	YES
$3 \rightarrow 5$	NO	NO	YES	YES	NO
$4 \rightarrow 5$	NO	NO	YES	YES	NO
FP	-	-	-	-	-

Table 4.2: The above table shows for each model, without ($K = 1$) and with knockdown experiments ($K = 6$), which edges are correctly inferred for the network in figure 4.1b, that contains a fan-in motif. No false positive edges were inferred.

Fan-out

The network depicted in figure 4.1c contains a fan-out motif: one node activates two other nodes at the same time point, and again contains 2 nodes activating 2 other nodes at the same time point. This is most probably why, for $K = 1$, lpNet-dyn/2 cannot infer edges $2 \rightarrow 4$ and $3 \rightarrow 5$, only for $K = 6$ can all models infer every edge correctly and no false positive edges, see table 4.3.

	$K = 1$		$K = 6$		
Edges	lpNet-dyn	lpNet-dyn2	lpNet-dyn	lpNet-dyn2	lpNet
$1 \rightarrow 2$	YES	YES	YES	YES	YES
$1 \rightarrow 3$	YES	YES	YES	YES	YES
$2 \rightarrow 4$	NO	NO	YES	YES	YES
$3 \rightarrow 5$	NO	NO	YES	YES	YES
FP	-	-	-	-	-

Table 4.3: The above table shows for each model, with ($K = 6$) and without knockdown experiments ($K = 1$), which edges are correctly inferred for the network in figure 4.1c, that contains a fan-out motif. No false positive edges were inferred.

Fan-in plus Fan-out

The network presented in figure 4.1d, contains both a fan-in and a fan-out motif. For this network, only lpNet-dyn2 is able to infer all edges correctly and no false positive edges for $K = 6$, as can be seen in table 4.4. lpNet-dyn, in the same situation, infers all edges correctly but also infers four false positive edges, and lpNet infers no edges at all. When no knockdown experiments are performed, lpNet-dyn only infers both edges starting at node 1 correctly, and fails to infer the remaining 3 edges, whereas lpNet-dyn2 only fails to infer $3 \rightarrow 4$.

Regarding the inferred false positive edges, the reason for this is the same presented for the cascade motif: by inferring extra inhibiting edges, the sum of slack variables added to the objective function is smaller, and the function is minimized.

	$K = 1$		$K = 6$		
Edges	lpNet-dyn	lpNet-dyn2	lpNet-dyn	lpNet-dyn2	lpNet
$1 \rightarrow 2$	YES	YES	YES	YES	NO
$1 \rightarrow 3$	YES	YES	YES	YES	NO
$2 \rightarrow 4$	NO	YES	YES	YES	NO
$3 \rightarrow 4$	NO	NO	YES	YES	NO
$4 \rightarrow 5$	NO	YES	YES	YES	NO
FP	-	-	$1 \dashv 5$, $5 \dashv 2$, $5 \dashv 3$, $5 \dashv 4$	-	-

Table 4.4: The above table shows for each model, with ($K = 6$) and without knock-down experiments ($K = 1$), which edges are correctly inferred for a network containing both a fan-in and a fan-out motif, see fig. 4.1d. Inferred false positive (FP) edges are shown in the last line, where $\dashv$ refers to an inhibiting edge.

Feedback loop

The network shown in figure 4.1e contains a feedback loop: node 2 activates node 3, node 3 activates node 4, and node 4 activates 2 again. Without the right perturbation experiments this type of structure is very hard to infer, since there is no need to infer the edge $4 \rightarrow 2$ to explain the value of node 2, and it would only lead to an higher value of the objective function, which has to be minimized. The results shown in table 4.5 show exactly that, all models, for both $K = 1$ and $K = 6$, infer all edges correctly and no false edges, except for edge $4 \rightarrow 2$, where node 4 activates node 2, which is already active. The only way to infer this edge would be to silence node 1, so that node 2 is inactive when the connection from 4 to 2 occurs, however, node 1 is the only source of signal in the network and when it is silenced all nodes are inactive. Therefore, an extra source node would have to exist for the edge $4 \rightarrow 2$ to be inferred, for instance, a second source node connected to node 4.

Feedforward loop

Figure 4.1f shows a network containing a feedforward loop: node 2 activates both nodes 3 and 4, and node 4 activates node 3, which is already active. Note that edges $2 \rightarrow 4$, $4 \rightarrow 3$ and $2 \rightarrow 3$ are an example of a situation where both indirect and direct signaling occur, and it is generally hard to infer all the edges correctly, as edge $4 \rightarrow 3$ does not need to be inferred to explain the data. The results in table 4.6 show

	$K = 1$		$K = 6$		
Edges	lpNet-dyn	lpNet-dyn2	lpNet-dyn	lpNet-dyn2	lpNet
$1 \rightarrow 2$	YES	YES	YES	YES	YES
$2 \rightarrow 3$	YES	YES	YES	YES	YES
$3 \rightarrow 4$	YES	YES	YES	YES	YES
$3 \rightarrow 5$	YES	YES	YES	YES	YES
$4 \rightarrow 2$	NO	NO	NO	NO	NO
FP	-	-	-	-	-

Table 4.5: The above table shows for each model, without ($K = 1$) and with knockdown experiments ($K = 6$), which edges are correctly inferred for the network in figure 4.1e, that contains a feedback loop. No false positive edges were inferred.

exactly that. This happens because nodes 3 and 4 are active immediately after node 2 is activated, thus increasing the probability that edges $2 \rightarrow 4$ and $2 \rightarrow 3$ are inferred. But since node 4 becomes active at the same time point as node 3, and node 3 is already active when the edge $4 \rightarrow 3$ occurs, it is highly unlikely for this edge to be inferred when no knockdown experiments are done. When knockdown experiments are performed and node 2 is silenced, it does not activate node 3, but since node 4 is also never active, it cannot activate node 3 and thus the edge is not inferred. One way to infer the edge $4 \rightarrow 3$, would be to have a second source node that would activate node 4 when node 2 is silenced, this way node 4 would activate node 3 and the edge could be inferred.

Summing up, apart from lpNet which infers no edges at all and lpNet-dyn which fails to infer edge $3 \rightarrow 5$ for $K = 1$, all models fail to infer the edge from node 4 to 3.

	$K = 1$		$K = 6$		
Edges	lpNet-dyn	lpNet-dyn2	lpNet-dyn	lpNet-dyn2	lpNet
$1 \rightarrow 2$	YES	YES	YES	YES	NO
$2 \rightarrow 3$	YES	YES	YES	YES	NO
$2 \rightarrow 4$	YES	YES	YES	YES	NO
$3 \rightarrow 5$	NO	YES	YES	YES	NO
$4 \rightarrow 3$	NO	NO	NO	NO	NO
FP	-	-	-	-	-

Table 4.6: The above table shows for each model, without ($K = 1$) and with knockdown experiments ($K = 6$), which edges are correctly inferred for a network containing a feedforward loop, fig. 4.1f. No false positive edges were inferred.

Assessing different criteria to classify an edge as non-null

At last, we also used these networks to assess the impact of considering an edge to be non-null when using two different criteria: 1) an edge is non-null if its median value is greater than its MAD; 2) an edge is non-null when it is inferred at least in 50% of the inferred networks.

The results obtained for criterion 2 are shown in section B.1, together with the results already presented for the first criterion. Overall, when using criterion 2, more true positive edges are considered as existing, yet, this happens at the cost of more existing false positive edges. On the other hand, when using criterion 1 most of the true positive edges are still considered as existing, while few false positive edges are considered to be non-null. Still, when a node has two incoming edges, such as in the feedback and feedforward loops, one of the edges is always considered as null, whatever criterion is used. Thus, we opt to use criterion 1 to classify an edge as existing or not, and have more confidence on the non-null inferred edges.

Finally, it should be noted that the absolute value of each edge $|w_{ji}|$ is not necessarily correlated to how much evidence there is in the data for the inference of a particular edge. On the inference of the above networks, we noticed that the edge values $|w_{ji}|$ depend mostly on the node state and on which incoming edges were inferred. For instance, there were situations in which, for an active node with a single incoming positive edge, the value of this edge was $|w_{ji}| = 0.8$. On the other hand, when an active node had both one incoming positive edge and one incoming negative edge, the value of the positive edge was $|w_{ji}| = 1.6$, in order to balance the effect of the inhibiting edge.

General discussion

In general, our model fails to infer the following two types of edges: 1) an edge that activates an already active node, and 2) "parallel" edges, e.g. edges that start at nodes 1 and 2 and activate nodes 3 and 4 respectively.

Under our assumptions, that the expression of an active node does not change with the number of incoming activating edges from active parents, the first type of edges are difficult to infer with most models. The reason is that, given an active node, there is no evidence in the data for further incoming activating edges from active parents. To solve this situation, the appropriate knockdown experiments need to be done, i.e., silencing one parent node while the child node is still activated by another parent node.

Regarding the inference of "parallel" edges, the difficulty is in which nodes influence which, as in terms of protein expression, when node 1 and 2 are active at $t = 1$ and nodes 3 and 4 are active at $t = 2$, several combinations of edges can explain the situation. It may be either node 1 or 2 that activates both nodes 3 and 4, it may be node 1 and 2 that activate node 3 and 4, respectively, or vice-versa. All the four alternatives are able to correctly explain the data and, to resolve the situation, the appropriate knockdown experiments need to be performed.

The above mentioned problems can be summed up in a single well known problem in network inference: in general, for a given dataset, there are several network topologies that can explain the data equally well.

Interestingly, there is an higher tendency for lpNet-dyn to classify null edges as non-null than for lpNet-dyn2. This happens because, in lpNet-dyn2, when a parent node j is inactive, the product $w_{ji}x_{jkt-1}$ is set to 0 and will have no influence on the problem's constraints, meaning there is less incentive for the edge w_{ji} to be inferred. Whereas, in lpNet-dyn, inferring an inhibiting edge w_{ji}^{-} might increase the value of the objective function by a lower amount than adding the necessary slack variables to satisfy constraints $w_i^0 + \sum_{j \neq i} w_{ji}x_{jkt-1} \leq 0 + \xi_{lt}$.

As a final note, please keep in mind that the inference results for the motifs presented in figures 4.1a-4.1f may not apply to all cases, as these depend mostly on the networks in which the motif is embedded and the time-frame of the signal propagation.

4.1.2 Artificial Ten-Node networks

In this section we use ten-node networks with simulated data to assess the performance of lpNet-dyn/2 in different conditions.

First, we study the performance of lpNet-dyn/2 with an increasing number of time points and knockdown experiments. To do this, we use the network shown in fig. A.3, and execute the model 100 times using always the same parameters for data generation, $\mu_{act} = 0.95$, $\mu_{inact} = 0.56$, $\sigma = 0.01$.

Second, we assess our models performance for data with different levels of noise, since a common problem when inferring biological networks from experimental data is the amount of noise in the measurements. To do this, we use the set of ten ten-node networks with positive edges plus 20% of negative edges, figures A.11-A.20. Furthermore, we compare our results to the results obtained with the original lpNet and with DDEPN. For this study the models were executed 30 times due to time

constraints, however, the overall results for 100 runs and 30 runs are similar, as shown in figures B.1-B.4 .

Finally, we use ddepn R package `makedata` function to generate data for both lpNet-dyn/2 and DDEPN, and assess the impact of using different assumptions to generate data on the comparison between these models. The motivation is that the method used to generate data in all other tests follows our assumptions, while the `makedata` function follows DDEPN assumptions, which are not the same and may introduce some bias in the results obtained. For this test the models were executed 50 times, and a single noise level was used ($\sigma = 0.01$).

Increasing number of time points

In figures 4.2 and 4.3, we present the results from lpNet-dyn/2, for $K = 1$ and $K = 11$, in terms of SP, SN, and PR for an increasing number of time points, more specifically from 2 time points (the minimum required by the model) to 12. The results represent median and respective MAD values calculated over 100 inferred networks.

The signal propagation in the network (fig. A.3) used for this test takes 6 time points to reach the end nodes, i.e., there are 6 node state vectors, one per time point. Hence, for a number of time points less than 6, the required number of time points and corresponding node state vectors was deliberately removed, typically for every other time point. For a number of time points greater than 6, the required number of time points and respective node state vectors were repeated. To choose the time points to be repeated, these were sampled with replacement. The specific time points and respective node state vectors used for each number of time points are shown in table A.1.

Before proceeding into the results analysis, one should note that the node state vectors at $t = 5$ and $t = 6$ are the same, all nodes are active, however, the signal propagation only stops at $t = 6$.

Overall, from figures 4.2 and 4.3 we verify that SP, SN, and PR values attain their maximum values between $T = 5$ and $T = 8$, with few exceptions. This happens most likely because at $t = 6$ the signal only reaches nodes that are already active, thus, it is hard to extract any more information from this last time point. Regarding the results for $T = 7$, 8, the repetition of only a few node state vectors appears to have a positive impact on overall results. Yet, if a greater number of node state vectors are repeated, the models performance tends to decrease. One reason for this is that, by repeating only some vectors, a bias is introduced towards the edges that become

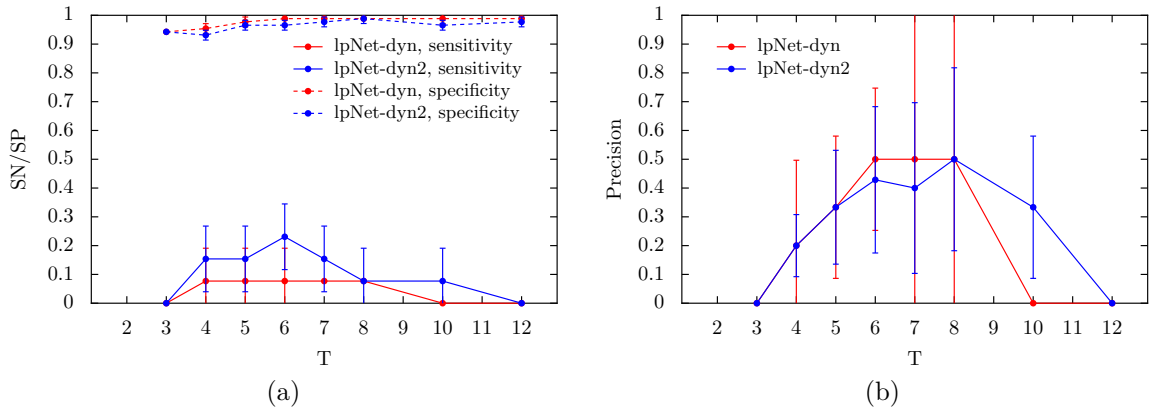


Figure 4.2: Results for an increasing number of time points when no nodes are silenced, $K = 1$, for one ten-node network. Both in (a) and (b) red refers to results from lpNet-dyn and blue to results from lpNet-dyn2. In (a) normal lines show the evolution of the median sensitivity and corresponding MAD, whereas dashed lines represent the evolution of specificity median values and respective MAD with increasing number of time points. In (b) precision median and respective MAD values with increasing number of time points are shown.

active at the respective time points. However, this impact is not the same for $K = 1$ and $K = 11$. For $K = 11$ the repetition of more node state vectors does not appear to have such a negative impact. This is should be because, as each gene is silenced once at each time point, the bias towards the inference of edges that become active at the repeated time points is reduced. Yet, one should note that, due to this bias, the results presented in here may change depending on the network to be inferred and which node state vectors are repeated.

Finally, note that lpNet-dyn/2 is not able to infer the network for $T = 2$, the resulting network being the empty network, i.e., $w_{ji} = 0 \forall \{i, j\}$. For $T = 2$, most nodes are inactive at the first time point and all nodes are active at the last time point, thus, the reason for this behavior is obvious when using lpNet-dyn2: if the parent nodes are inactive at $t = 0$, they do not contribute to their children's *activity* at $t = 1$ and, in principle, no edges w_{ji} are inferred, as these do not influence the LP problem constraints. Instead, the constraints are satisfied by setting $w_i^0 \approx \delta_i$. For lpNet-dyn, what probably happens is that the x_{jkt-1} values are so low that, in terms of minimizing the objective function, it compensates to set $w_i^0 \approx \delta_i$ instead of setting $w_{ji} > 0$. However, for this particular network, it can be inferred for $T = 2$ if knockdown experiments are performed and the data for the first time point corresponds to $t = 5$, when all nodes are active. These results are presented in section B.3.

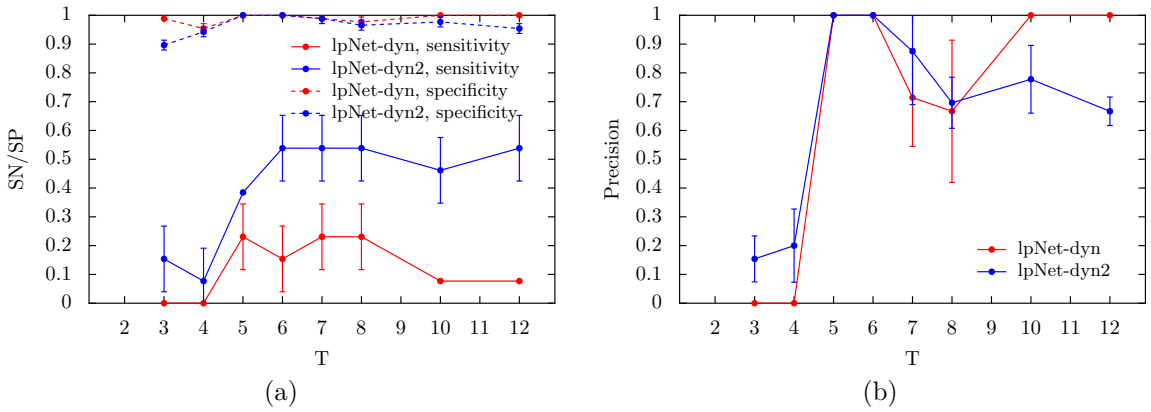


Figure 4.3: Results for an increasing number of time points when 11 knockdown experiments are performed, $K = 11$, for one ten-node network. Red and blue refer to results from lpNet-dyn and lpNet-dyn2, respecetively. In (a) normal lines represent the evolution of the median sensitivity and corresponding MAD, whereas dashed lines show the evolution of specificity median values and respective MAD with increasing number of time points. In (b) precision median and respective MAD values with increasing number of time points are shown.

Increasing number of knockdown experiments

To study the influence of the number of knockdowns on the performance of lpNet-dyn/2 , we varied the number of knockdown experiments from 1, $K = 1$, (no silenced nodes) to 16, $K = 16$, (one gene silenced in each experiment, 5 experiments with double knockdowns, plus one experiment with no silenced nodes) while keeping the number of time points, T, constant. For each value of K, the set of nodes silenced for $K - 1$ knockdown experiments was kept, and a new node(s) was sampled from the set of nodes that were not chosen for silencing yet. Thus, the same node is never silenced in two different experiments of the same type (single or double knockdowns). The list of silenced nodes for each value of K is shown in table A.2.

The results obtained are shown in figure 4.4, these represent SP, SN, and PR median and respective MAD values calculated over 100 inferred networks. There is a general increasing trend in SP, SN, and PR values for a growing number of knockdowns. Yet, this trend is not steady, for some K values SP, SN, and/or PR actually decrease in comparison to previous K values, which might be due to the particular nodes that were silenced in the respective experiments. For instance, lpNet-dyn SN and PR decrease for $K = 4$ – when nodes 1, 7, and 8 are silenced –, comparing to $K = 1$. Node 8 is an end node with a single incoming edge and, since the models say nothing about incoming connections for silenced nodes, this edge is most probably not inferred for $K = 4$ while it is probably be inferred for $K = 1$, when no nodes

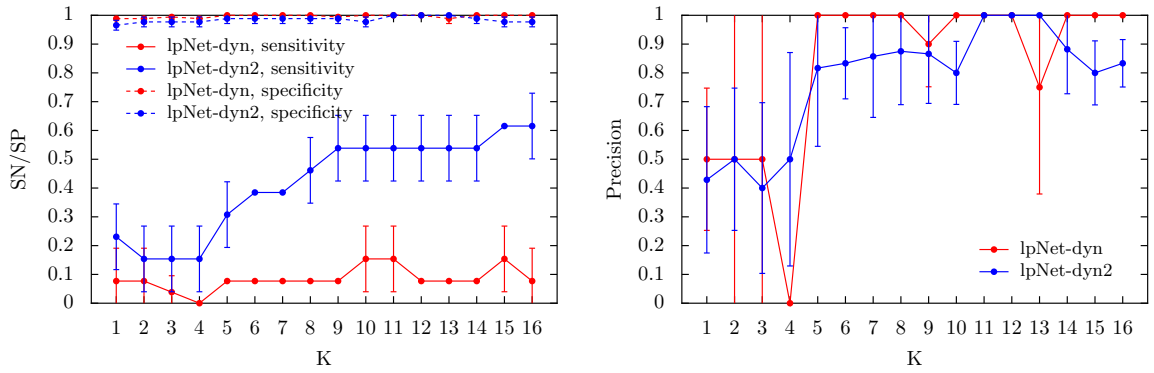


Figure 4.4: Sensitivity, Specificity, and Precision values for an increasing number of knockdown experiments, $K = 1, 16$. Red and blue lines refer to results from lpNet-dyn and lpNet-dyn2, respectively. Normal lines in (a) represent the evolution of median Sensitivity values and respective MAD with growing number of knockdown experiments, dashed lines represent the evolution of Specificity median values and respective MAD, while in (b) lines represent evolution of Precision median values and respective MAD.

are silenced. Concerning nodes 1 and 7, these are nodes with a single incoming edge and with outgoing edges that activate already active nodes. Hence, by silencing these nodes, their incoming edges are probably not inferred in comparison to $K = 1$, while their outgoing edges are hard to infer in any case, as they are not crucial for the activation of their child nodes.

In short, the more knockdown experiments are performed, the better are the results. Yet, including few knockdown experiments is not always beneficial, it depends on which nodes are silenced. In general, if the silenced nodes have a single incoming edge and outgoing edges that activate already active nodes, silencing them may actually lead to worse results.

It is also interesting to note that lpNet-dyn and lpNet-dyn2 SN, SP, and PR values, while displaying the same tendencies, do not exhibit the same exact behavior.

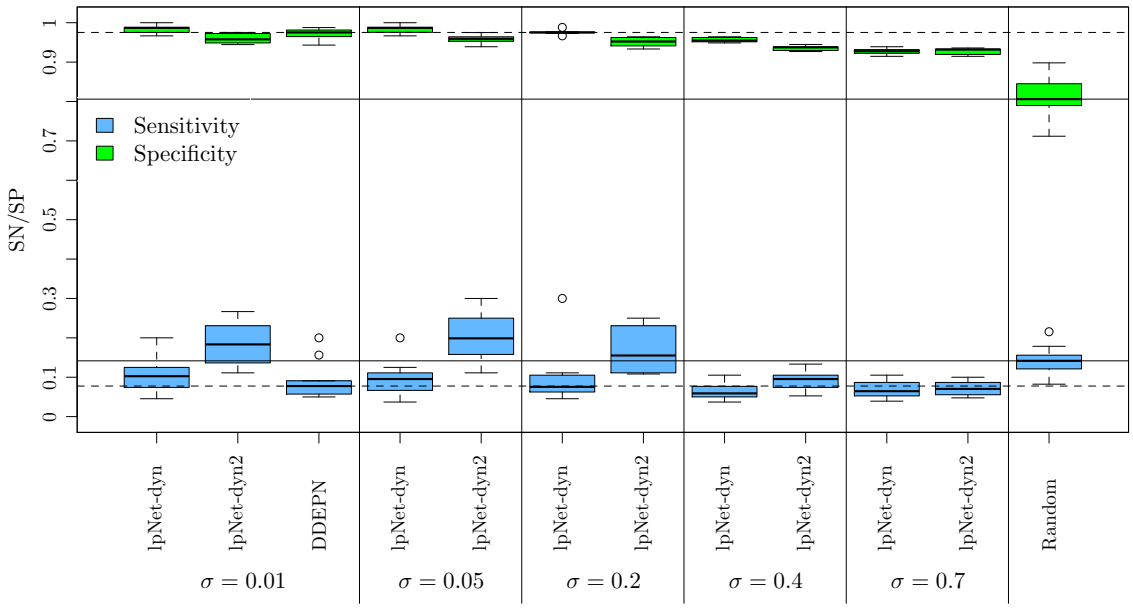
Increasing level of noise in the data

We now assess the impact of noisy data on lpNet-dyn/2, by measuring its performance in terms of SP, SN, and PR values for increasing levels of noise in the data, from $\sigma = 0.01$ to $\sigma = 0.7$ for $K = 1$, and from $\sigma = 0.01$ to $\sigma = 0.2$ for $K = 11$. These results are compared to lpNet results for all noise values and $K = 11$, and to DDEPN results for $\sigma = 0.01$ and $K = \{1, 11\}$.

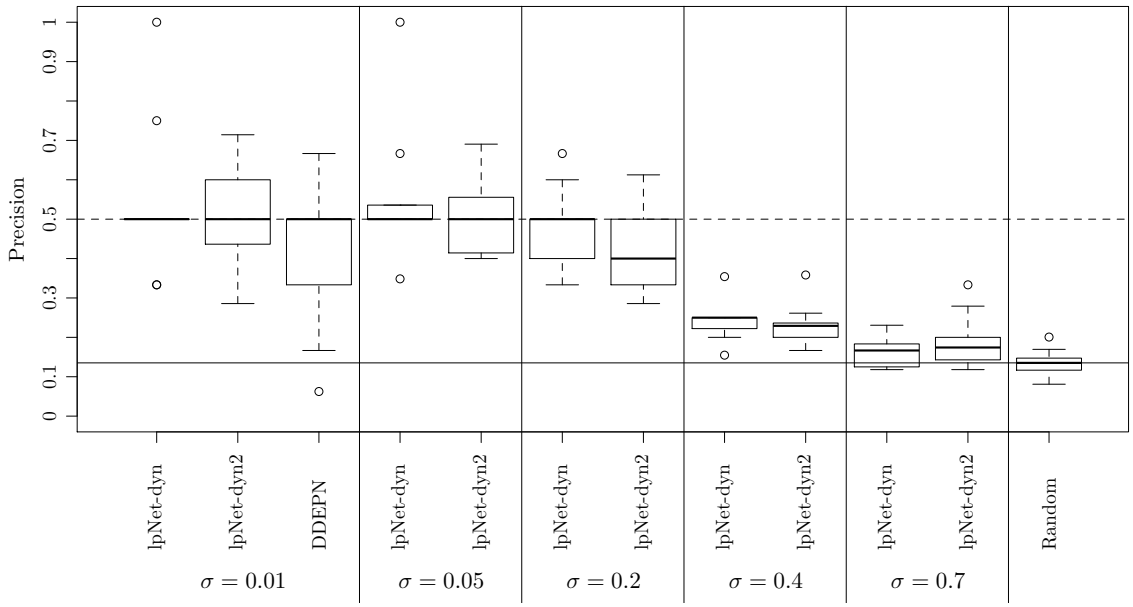
Before proceeding to the results analysis, one must note that the way to include knockdown experiments in DDEPN is very different from ours. In DDEPN, knockdown experiments are encoded as stimuli experiments, in which inhibiting stimuli nodes are added to the network, as well as edges to their target nodes, which will be inhibited. These edges are then inferred by the model, together with the edges from the original network (the network without stimuli nodes). Additionally, one or more stimuli nodes, which activate the network source nodes, must exist. Hence, when knockdown experiments are included, $K = 11$, DDEPN needs to infer an extra 10 edges from inhibiting stimuli to target nodes and the edges from stimuli node to source nodes, in addition to the original network edges. On the other hand, in lpNet-dyn/2 the knockdown experiments are modeled by an activation matrix, which states whether the node is active or not.

One way to include prior information about silenced nodes in DDEPN is to include prior knowledge regarding the stimuli to target edges, and say nothing about the remaining edges. Still, besides the prior knowledge itself one must also supply its "strength", whose value depends on the dataset used. Taking the results in Bender *et al* [7] as a reference, we chose a value of 0.01. However, the obtained results indicated that the "strength" of the prior was too large, and little non-null edges were inferred, the results being actually worse in terms of sensitivity and precision than when using no prior. Therefore, the results here presented refer to DDEPN with no prior.

Overall, when no nodes are silenced, $K = 1$, the performance of lpNet-dyn/2 is relatively robust for noise values up to 0.2, while it drops significantly for higher values, see fig. 4.5. One reason for this is that, when $\sigma = 0.4$, active nodes can have values as low as inactive nodes and vice-versa. In addition, the δ value also depends on σ, thus it becomes hard to distinguish an active node from an inactive one, and the results deteriorate. In general, lpNet-dyn performs better than lpNet-dyn2 in terms of specificity and precision. Yet, lpNet-dyn2 is the only model able to attain sensitivity values significantly above random, while DDEPN performs worse in terms of sensitivity than both lpNet-dyn and lpNet-dyn2, and performs roughly as well as both models in regard to specificity and precision values. For the highest value of noise, $\sigma = 0.7$, lpNet-dyn/2 still performs better than random in terms of PR and SP, performing below random in terms of sensitivity.

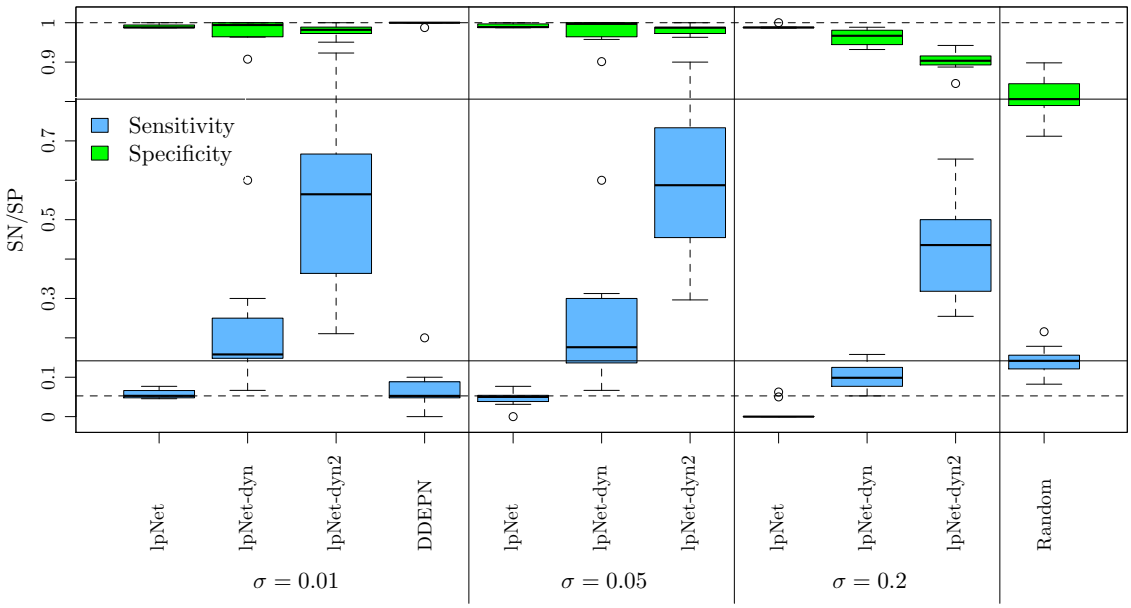


(a)

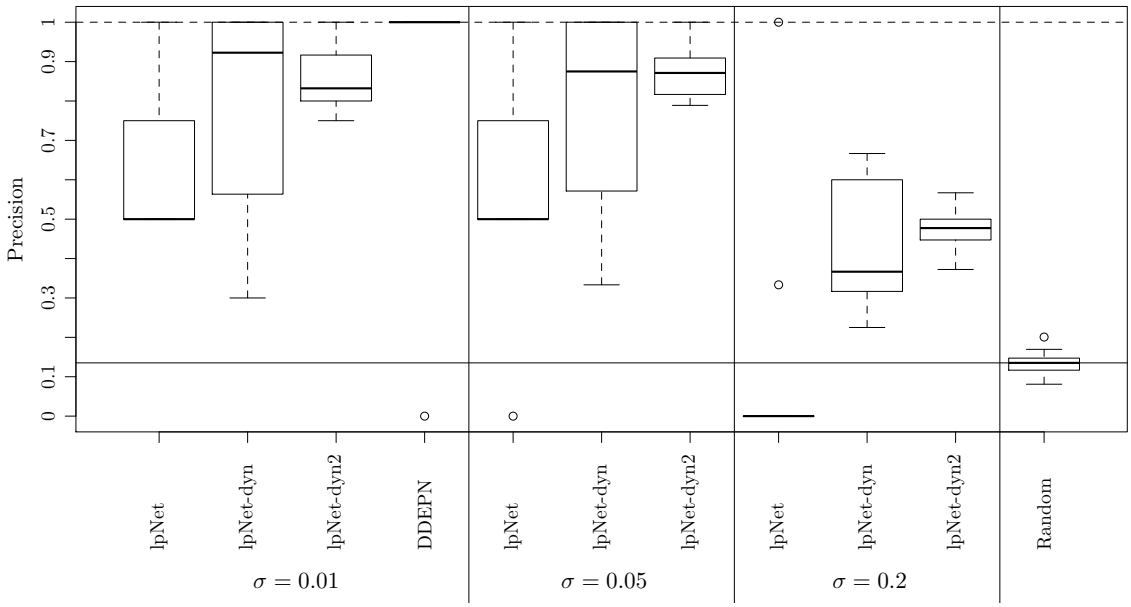


(b)

Figure 4.5: Sensitivity, Specificity, and Precision values for different noise values, from 0.01 to 0.7, for $K = 1$. A set of ten ten-node networks was used. (a) Shows Sensitivity (blue) and Specificity (green) values for lpNet-dyn/2, lpNet, and DDEPN. (b) Shows Precision values for the same models. The horizontal solid black line and the dashed black line represent the median value of SN/SP/PR obtained for random prediction and with DDEPN, respectively.



(a)



(b)

Figure 4.6: Sensitivity, Specificity, and Precision values for different noise values, from 0.01 to 0.2, for $K = 11$. A set of ten ten-node networks was used. (a) Shows Sensitivity (blue) and Specificity (green) values for lpNet-dyn/2, lpNet, and DDEPN. (b) Shows Precision values for the same models. The horizontal solid black line and the dashed black line represent the median value of SN/SP/PR obtained for random prediction and with DDEPN, respectively.

For $K = 11$, 4.6 it is noteworthy the increase in lpNet-dyn2 sensitivity values

relative to $K = 1$, which is larger than the increase in lpNet-dyn SN values, showing that the extra constraints added to lpNet-dyn2 fulfill their purpose: inferring more TP edges and less false negatives. Overall, lpNet-dyn/2 is robust to noise values up to $\sigma = 0.05$, showing a significant decrease for $\sigma = 0.2$, as this noise value allows for an overlap of values for active and inactive nodes. Plus, since there are more inactive nodes for greater values of K, the overlap probability increases and the decrease in SN, SP, PR values is observed for lower noise levels than when $K = 1$. Regarding SP values, these are constantly high for lpNet-dyn/2 when $\sigma = 0.01, 0.05$, showing a significant drop for $\sigma = 0.2$, to which lpNet-dyn is more robust. As for Precision median values, lpNet-dyn performs slightly better than lpNet-dyn2 for $\sigma = 0.01$ but its performance gets worse than lpNet-dyn2 for $\sigma = 0.2$.

Regarding DDEPN performance alone, it attains very high SP and PR values but very low SN values (below random), showing that all positive edges it infers are actually positive edges. However, it infers little positive edges in comparison to false negative edges. As for Knapp and Kaderali model, lpNet, except for SP, it performs worse than lpNet-dyn/2. Yet, this model cannot take advantage of time-series data, which provides more information on the node states than steady-state data with single knockdown experiments. Plus, it has also been shown to perform better when double knockdowns are used [45]. Therefore, the comparison is biased, and the results were expected.

Using data generated with ddepn R-package

To assess the impact of generating data based on different assumptions on the comparison between lpNet-dyn/2 and DDEPN performance, we used the ddepn R package function `makedata` to generate data for both models and compare the results. The key difference between this function and our method to generate data, is in the assumptions for the signal propagation through the network. In particular, on the influence of incoming inhibiting connections. Given a node, in DDEPN it is assumed that a single incoming inhibiting edge from an active parent is enough to render the node inactive, no matter how many incoming activating edges from active parents there are. On the other hand, we assume that the given node is rendered inactive only if there is a greater or equal number of incoming inhibiting edges from active parents than incoming activating edges from active parents.

The function `makedata` generates a dataset with 9 replicates and 10 time points, which are used without modifications for DDEPN, while the replicate values are averaged into a single measurement for lpNet-dyn/2.

To do this study, we chose one network that possesses 30% of negative edges (fig. A.21). This choice was made because, due to the different signal propagation assumptions implicit in each model, we expect a greater impact in the results for networks containing negative edges. Furthermore, we performed tests with a single knockdown/stimuli experiment, $K = 1$, and with 11 knockdown/stimuli experiments, $K = 11$. For DDEPN we also tested the use of a prior for edges from stimuli to target nodes, with $\lambda_p = 0.1$, and setting the value for the remaining edges to 0.

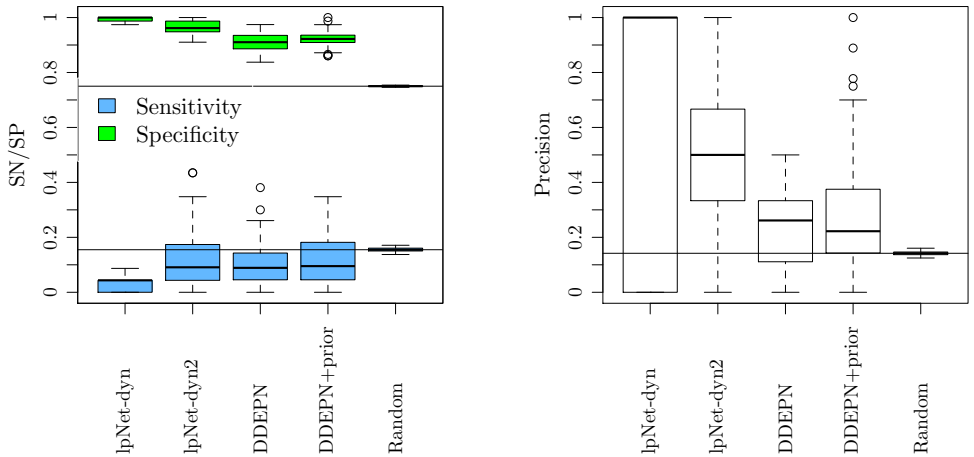
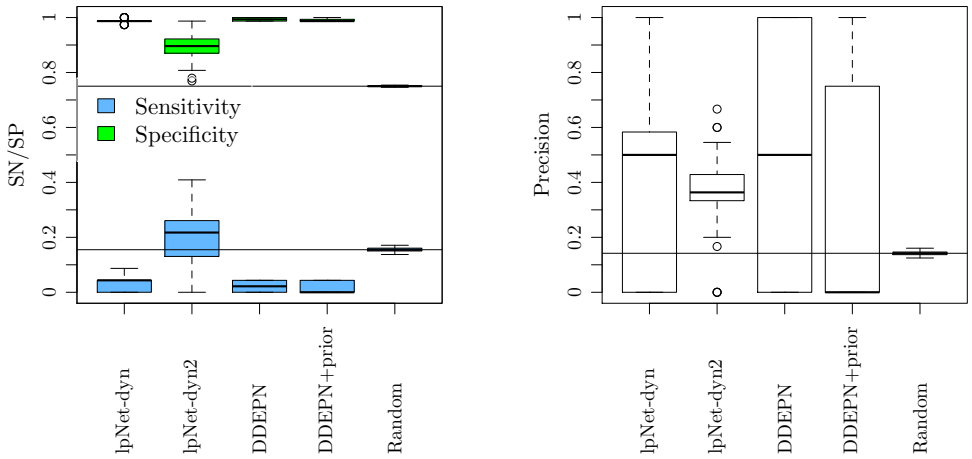


Figure 4.7: (a) Sensitivity (blue) and Specificity (green) values for lpNet-dyn, lpNet-dyn2, DDEPN, and DDEPN when uing a prior. (b) Precision values for the same models. The results refer to the inference of network in fig. A.21, when no nodes are silenced. The black line represents the SN/SP/PR median value for random prediction.

For $K = 1$, fig. 4.7, lpNet-dyn2 generally performs better than the other models, although its median SN value is worse than random, and DDEPN results with and without prior are similar. This may be because there are only two edges from stimuli to target nodes.

For $K = 11$, fig. 4.8, lpNet-dyn2 is the only model that attains median values for SN, SP, PR above random, however, both DDEPN and lpNet-dyn achieve better SP and PR values, at the cost of very low sensitivity values. Thus, lpNet-dyn2 shows a more balanced performance. Regarding the introduction of a prior for DDEPN, it leads to worse results than when using no prior. Taking into account that it achieves very high SP values but almost null SN and PR values, this is probably because the value chosen for λ_p leads to a prior that is too strong, and little positive edges on the original network are inferred.



(a) (b)

Figure 4.8: (a) Sensitivity (blue) and Specificity (green) values for lpNet-dyn, lpNet-dyn2, DDEPN, and DDEPN when uing a prior. (b) Precision values for the same models. The results refer to the inference of network in fig. A.21 for $K = 11$. The black line represents the SN/SP/PR median value for random prediction.

Summing up, even by using data that fits DDEPN assumptions instead of our model assumptions, in general, lpNet-dyn2 shows a more balanced performance than the other models, and sensitivity values for $K = 11$ are especially noteworthy.

At last, we briefly analyzed the running times of lpNet-dyn/2 and DDEPN models.

lpNet-dyn/2 running time increases with the number of nodes n, number of time points T, and number of knockdown experiments K, as all of these lead to the inclusion of more constraints in the LP problem, which then takes more time to be solved. Moreover, its running time also increases with the noise in the data. This happens because the upper value for λ increases, and the defined interval for possible values of λ is constant. Yet, this can be adapted to reduce the run time with no significant impact in the results.

To infer each network 10 times for each value of $\sigma = \{0.01, 0.05, 0.2, 0.4, 0.7\}$ with $K = 1$ lpNet-dyn/2 took approximately 1h, while DDEPN took ∼2h to infer each network 3 times for $\sigma = 0.01$. For $K = 11$, it takes ∼13h for lpNet-dyn/2 to infer 3 networks for each value of $\sigma = \{0.01, 0.05, 0.2\}$, and it takes ∼10h for DDEPN to infer 3 networks for $\sigma = 0.01$. Therefore, although no precise measurements were performed on running times, one can conclude that lpNet-dyn/2 executes much faster and with comparable, if not better, results than DDEPN, especially when

no knockdown experiments are performed. This is mostly due to the low practical complexity of the simplex algorithm used to solve the LP problem, which is the core of our model.

4.1.3 General discussion

Overall, lpNet-dyn2 achieves better SN values than lpNet-dyn, and similar SP values. One reason for the difference in SN values is that, by introducing the extra constraints in lpNet-dyn2, the probability of classifying non-null edges as null decreases. This happens because the product $w_{ji}x_{jkt-1}$ is set to zero also when the parent node j is inactive ($x_{jkt-1} < \delta_j$) and not only when $w_{ji} = 0$. Thus, assuming $w_i^0 = 0$, it is possible to satisfy the constraint that requires the activity of a node i to be $\leq 0 + \xi_{lt}$, when it is inactive, without the need to set $w_{ji} = 0$ or add slack variables ξ_{lt} .

Usually, an higher SN value reflects in lower SP values, yet, we have observed that the SP values are mostly above 0.9 and are similar for both lpNet-dyn and lpNet-dyn2. In addition, these values are fairly constant, which is probably due to the network sparsity: many true negative edges are inferred, and even a moderate change in the number of true negative or false negative inferred edges will result in little change in the SP value.

Regarding precision values, these are highly variable and both models perform similarly well. The PR values high variability is probably due to the low number of edges classified as non-null. Thus, a little change in the number of TP or FP leads to an high change in PR values.

On the comparison between lpNet-dyn/2 and DDEPN. Generally, lpNet-dyn/2 performs better than DDEPN, but we need to take into account that the information about the nodes to be silenced is included in different ways for each model, which results in the need to infer extra edges for DDEPN, thus making the comparison biased. By using a prior, the comparison can be done in a less biased way, but the strength of this prior needs to be adapted for each dataset, which would not be possible in a real experimental scenario. Summing up, although DDEPN can take advantage of perturbation experiments in general, the way these are included in the model makes it difficult to obtain good results for an high number of experiments.

As for the comparison between the original lpNet and lpNet-dyn/2, the objective was to confirm our expectations, that lpNet-dyn/2 is able to take advantage of time-series data, producing better results. Otherwise the comparison is biased, since these models were designed to use different types of data.

4.2 Experimental data

4.2.1 ERBB G1/S dataset

In this section, we present the results obtained for the ERBB G1/S dataset with lpNet-dyn/2, and quantitatively compare its performance to lpNet and random prediction. The models were executed 5 times, and always inferred the same network.

To infer the network relative to this dataset with lpNet, replicate measurements were summed up by calculating their mean values, and the δ value was set as the protein expression for the MOCK control at the first time point, according to [45]. As for μ_{act} and μ_{inact}, these were set to 0.95 and 0.56, respectively, while σ was set to 0.01.

lpNet-dyn only inferred four non-null edges, which are not part of the network assembled with Ingenuity, thus this results will not be further analyzed. As for lpNet-dyn2, it classified 16 edges as non-null, which are represented in figure 4.9, where the value of each edge w_{ji} is not necessarily correlated to how much evidence from the data there is. We will now enumerate the edges for which we found support from literature:

- ERK1/2 $\dashv$ pRB1, ERK has been shown to phosphorylate Cyclin D1 and lead to its degradation [64], while an active form of Cyclin D1 is needed to form the Cyclin D1-CDK4/6 complex that activates pRB [91]. Thus, ERK is able to, indirectly, inhibit pRB1.
- ERK $\rightarrow$ p27, ERK signaling activates MYC [63], which in turn inhibits p27 activity [52].
- p21 $\rightarrow$ ERK1/2, p21 is known to increase the phosphorylation of cFos and MBP by ERK1 and ERK2 [61].
- p21 $\rightarrow$ pRB1, p21 is known to inactivate cyclins and prevent the phosphorylation of pRB1 during the cell cycle [82].
- ERBB2 $\rightarrow$ ERBB1, these receptors are known to form heterodimers [94].
- ERBB2 $\rightarrow$ p27, evidence for an indirect inhibitory connection was found, since ERBB2 stimulates ERK activation [94], and ERK activates MYC [63] which inhibits p27 [52].

- ERBB1 → ERK1/2, the activation of ERBB1 homodimers leads to the stimulation of ERK activation [94]
- Cyclin D1 → pRB1 and CDK4→pRB1, the active form of the CyclinD1-CDK4/6 complex is known to phosphorylate pRB1 [91].
- CDK2 → pRB1, an increase in levels of the complex Cyclin E–CDK2 is linked to the completion of pRb phosphorylation [2, 50, 60].

Summing up, we found literature support for 11 out of 16 inferred connections. However, 3 of these connections were inferred with an opposite effect to the one found in literature, e.g. an indirect inhibitory connection between ERBB2 and p27 was found, while lpNet-dyn2 inferred an activating edge. Although no literature evidence was found for an activating connection from CDK4 to CDK2, these proteins are both involved in the cell cycle regulation. The complexes formed by Cyclin D-CDK4 and Cyclin E-CDK2 are known to phosphorylate pRB sequentially [82].

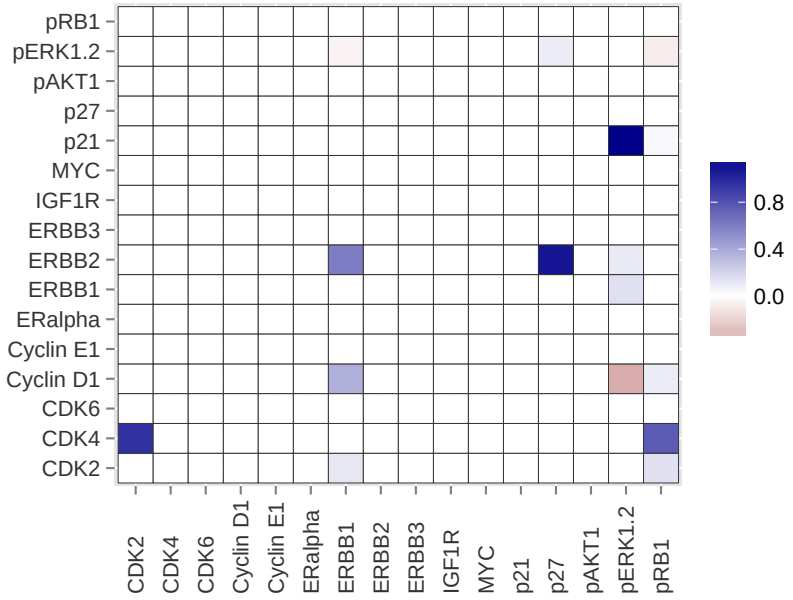


Figure 4.9: Heat plot of the median value of inferred edge weights $w_{i,j}$ with lpNet-dyn2, for the ERBB G1/S dataset. Blue represents positive edges, red represents negative edges, and white represents null edges. The less transparent the colors are, the greater the value of the inferred edge $w_{i,j}$.

We further evaluate our results quantitatively. By using the network assembled using Ingenuity IPA (see fig. A.22), we calculate SN, SP, PR, and accuracy (AC) values. The results are presented in table 4.7, which shows that both lpNet and lpNet-dyn2

perform significantly better than random prediction for all measures (p-value $< 10^{-4}$ using a one-sided Mann–Whitney–Wilcoxon test [53]) except for sensitivity, meaning that both models have a difficulty on inferring true non-null edges. Comparing lpNet-dyn2 and lpNet, the former performs slightly better than lpNet in terms of sensitivity and precision values, and equally well in terms of specificity and accuracy values.

	lpNet-dyn2	lpNet	Random
SN	0.1	0.08	0.36
SP	0.96*	0.96*	0.64
PR	0.56*	0.50*	0.36
AC	0.64*	0.64*	0.54

Table 4.7: Results in terms of sensitivity, specificity, precision, and accuracy values, calculated using a network assembled using Ingenuity IPA. Results are presented for both lpNet-dyn2 and the original lpNet, as well as for random prediction. * means the results are significantly better than random, with a p-value $< 10^{-4}$ for a one-sided Mann-Whitney-Wilcoxon test.

Note that the ERBB G1/S dataset includes 16 proteins, but there are no measurements for 6 of these proteins, i.e. there are 6 latent nodes out of 16. Thus, a brief study on the performance of lpNet-dyn2 was done in order to assess the impact of a growing number of latent nodes, up to 50%. The results are presented in section B.4 for $K = 11$, and show relatively robust results in terms of specificity and precision values, with a steady decrease in sensitivity values with growing number of latent nodes. Therefore, one reason for the low sensitivity values may be the number of latent nodes. Additionally, this dataset contains only 2 time points, for which lpNet-dyn2 is not guaranteed to work well, as shown in section 4.1.2. Another reason for the obtained results is the almost certain presence of noise in the data, which, if high enough, leads to lower values of SN, SP, and PR, as shown in section 4.1.2.

Finally, we should note that literature support was found for a few connections that are not part of the network assembled with Ingenuity, e.g. the activation of MYC by ERK. Thus, this network is used as a reference, but is not considered a gold standard. This represents one of the difficulties of assessing network inference results from experimental datasets, even for the most well studied networks there exists no gold standard, and not finding literature evidence for a given connection does not necessarily mean the edge does not exist.

4.2.2 ERBB signalling cascade - HCC1954 dataset

In this section, we present the results obtained for the HCC1954 dataset with lpNet-dyn/2, and compare its quantitative performance with DDEPN and random prediction.

To infer the network using DDEPN, we formatted the data using the function `format_ddepn` from the ddepn R package and defined the following settings: population size 500, maximum number of iterations 1000, crossover rate 0.3, mutations rate 0.8, which are in agreement with the settings used in [6]. Plus, only edges that are inferred in at least half of the networks are considered as non-null. We then executed the model 5 times, yet, we were not able to reproduce the results presented in [6]. Since we used exactly the same parameters as in [6], one probable reason for this lies in the use of a Genetic Algorithm to find the set of networks that explain the data best. A genetic algorithm is a search heuristic, thus it may not always converge to the best result and may deliver different results each time it is executed. Besides, when evaluating the likelihood of a network, similar results can be obtained for different networks, since in general a given dataset can be explained equally well by different network topologies.

lpNet-dyn/2 were executed also 5 times, for which lpNet-dyn did not infer more than 3 positive edges in each execution, thus its results are not further analyzed. Regarding lpNet-dyn2, the inferred edges are represented in figure 4.10.

We will now enumerate the edges inferred by lpNet-dyn2 for which we found literature evidence:

- SRC $\rightarrow$ PDK1, in vascular smooth muscle cells the phosphorylation of PDK1 was shown to depend on SRC [84].
- PDK1 $\rightarrow$ PKCα, in vitro studies have shown PDK1 to phosphorylate the activation loop of PKCα and βII [15].
- PDK1 $\rightarrow$ PRAS, PDK1 is known to phosphorylate AKT [13, 51], which in turn has been shown to phosphorylate PRAS [46].
- mTOR $\rightarrow$ GSK3, in dendritic cells GSK3 activity has been shown to be regulated by signaling linked to mTOR [88].
- ERBB4 $\rightarrow$ ERBB2 + ERBB2 $\rightarrow$ ERBB4, Tzahar *et al* [89], demonstrated the existence of the ERBB2-ERBB4 heterodimer.

- ERBB4 → ERBB3, Riese *et al* [70] have shown that ERBB3 and ERBB4 interact.
- ERBB1 → ERBB2, Tzahar *et al* [89], demonstrated the existence of the ERBB1-ERBB2 heterodimer .
- ERBB1 → ERBB3 + ERBB3 → ERBB1, Tzahar *et al* [89], demonstrated the existence of the ERBB1-ERBB3 heterodimer.
- ERBB1 → ERBB4, Tzahar *et al* [89], demonstrated the existence of the ERBB1-ERBB4 heterodimer.
- ERBB2 → PRAS, activation of ERBB2 has been linked to an increase in AKT expression [79], which in turn phosphorylates PRAS [46].

Summing up, literature support was found for 12 of the 16 inferred connections, and 7 of these connections refer to the heterodimers formed among the ERBB receptors. Although no evidence for a connection from ERBB1 to PRAS was found, its existence is plausible when taking into account that ERBB1 and ERBB2 are known to form heterodimers, and there is evidence of an indirect connection from ERBB2 to PRAS. Concerning the edges from AKT to PDK1 and from PDK1 to SRC, interestingly, literature support was found for connections in the opposite direction [13, 51, 84].

We now evaluate the results quantitatively by using the network assembled using Ingenuity IPA to calculate SN, SP, PR, and AC values for each model's results.

In this assessment we also include the results obtained with lpNet-dyn2 when using a different criterion to define δ_{ik} value. Assuming the distribution of the expression values for the active and inactive nodes in each stimuli experiment can be approximated by a normal distribution for each state, we used the R package mclust[1] [19, 20] to find the parameters of these distributions, by forcing the number of distributions to be 2. Then we set δ_{ik} as the intersection of the two distribution's curves. In case the curves do not intersect, we subtract the curve corresponding to the active state from the other curve and set δ_{ik} as the root of the function. When these curves are superposed, we consider the node to be always in the same state and set the value for δ according to whether the node corresponds to a receptor protein or not: if yes, $\delta_{ik} = 0$ and the node is always active, otherwise $\delta_{ik} = \max_t \{x_{ikt}\} + 1$ and the node is considered to be always inactive. Regarding the parameters of the normal

[1] briefly describe mclust

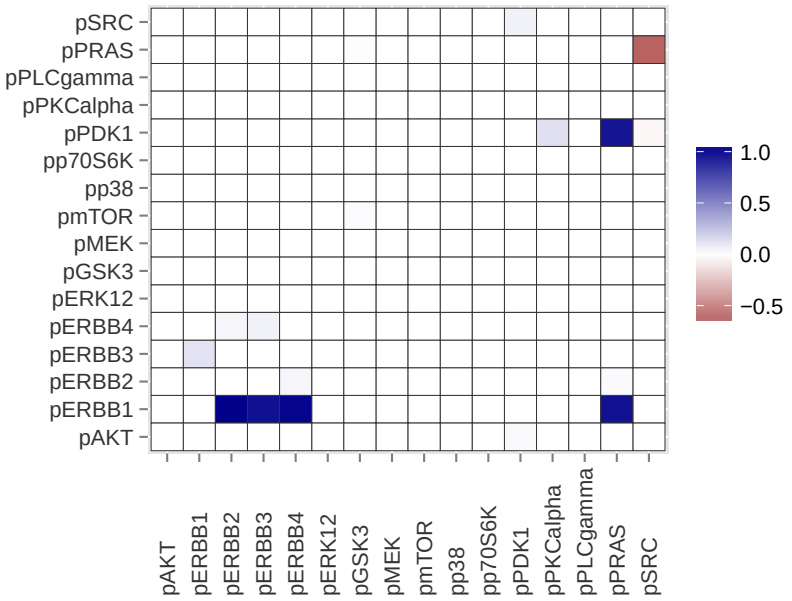


Figure 4.10: Heat plot of the median value of inferred edge weights $w_{i,j}$ with lpNet-dyn2 for the HCC1954 dataset. Blue represents positive edges, red represents negative edges, and white represents null edges. The less transparent the colors are, the greater the value of the inferred edge $w_{i,j}$.

distribution $\mathcal{N}(\mu, \sigma)$ used to predict the value of a node in the CV step, $\mu_{act,i,k}$ and $\mu_{inact,i,k}$ are the means of the respective normal distributions and $\sigma = 0.01$. We will refer to δ_{ik} defined according to this criterion as δ^2 and to δ_{ik} defined in section 3.4.2 as δ^1.

The obtained results are shown in table 4.8, these values represent the average value over the results obtained for each of the 5 runs of each model. For random prediction the values are averaged over the 1000 networks generated by permuting each row and column of the true network 1000 times.

Both criteria used to set δ's value lead to similar values of SN, SP, PR, and AC, which are higher than the same values for DDEPN. Quantitatively, the main difference between using δ^1 and δ^2 is the number of true positive edges inferred. No model is able to achieve a sensitivity value better than random, while lpNet-dyn2 achieves better values for all other measures, and DDEPN achieves values better than random only for specificity and accuracy. After using a Shapiro-Wilk test [80] to verify the normality of the values obtained for each measure and model, and obtaining p-values above 0.2, a one sided Student's t-test [83] was performed to check whether the results obtained with lpNet-dyn2 and DDEPN are statistically significantly better than random prediction. For lpNet-dyn2, a p-value of less than 10^{-6} and 10^{-3} was

obtained for SP + AC, and PR values, respectively, whereas for DDEPN a p-value of less than 0.005 was obtained for SP and AC values.

Note that the results obtained with lpNet-dyn2 are more stable than the ones obtained with DDEPN, as can be seen from the higher standard deviations. The same conclusion is achieved after analyzing the set of inferred edges for each run in both models (results not shown here).

	lpNet-dyn2, δ^1	lpNet-dyn2, δ^2	DDEPN	Random
SN	0.13 ± 0.01	0.113 ± 0.007	0.07 ± 0.04	0.24
SP	0.973**± 0.004	0.973** ± 0.006	0.88*** ± 0.04	0.76
PR	0.61*± 0.04	0.52* ± 0.05	0.15 ± 0.09	0.24
AC	0.773**± 0.003	0.767** ± 0.005	0.70*** ± 0.03	0.64

Table 4.8: Results in terms of Sensitivity, Specificity, Precision, and Accuracy are shown for the network inference on the experimental dataset regarding the ERBB signalling cascade in HCC1954 breast cancer cell lines. These results are presented for the lpNet-dyn2 and DDEPN models, as well as for random prediction. For lpNet-dyn2, results are shown for δ values calculated according to 2 different criteria, δ^1 and δ^2. Values statistically significantly higher than the random prediction values are marked as follows: * p-value $< 10^{-3}$, ** p-value $< 10^{-6}$, *** p-value < 0.005.

On the execution time of lpNet-dyn2 and DDEPN for this dataset, lpNet-dyn2 execution time for each run of the model is of $\approx$ 6h30min, while DDEPN run time is of $\approx$ 32h for each run.

Overall, lpNet-dyn2 is not able to infer all non-null edges, however, from the set of non-null edges it infers, more than half are true positives. Thus, this model provides reliable results, and which are highly reproducible. Quantitatively, lpNet-dyn2 performs better than DDEPN and better than random prediction, except in terms of sensitivity, as the model tends to infer a small number of true positive edges. One major reason for this, is that lpNet-dyn2 is not able to achieve sensitivity values significantly higher than random when no knockdown experiments are performed, which is the case here. Another reason, may be the existence of edges that activate nodes which are already active, and which are difficult to infer in general, as shown for the inference of a feedforward loop in section 4.1.1.

Interestingly, when using different criteria to define δ_{ik} value, δ^1 and δ^2, although the quantitative results were similar, the set of edges inferred when using each criteria were very different (the set of edges inferred for δ^2 is shown in fig. B.7). This behavior

highlights the importance of the δ value, as the obtained results depend mostly on the ability to correctly distinguish an active state from an inactive one.

With this dataset, we also executed lpNet-dyn2 with prior knowledge included, by defining the source nodes to be ERBB1/2/3/4. Qualitatively, more edges were inferred (see fig. B.8), and quantitatively, the precision values were slightly worse (results not shown here). A reason for this is that, by including this type of knowledge we are forcing all nodes that are not source nodes to have incoming edges, which leads to the inference of edges that are not part of the network assembled with Ingenuity IPA.

Note that, as with the G1/S dataset, literature evidence was found for edges that are not included in the network assembled with Ingenuity IPA, while no conclusive support was found for a few edges included in this network. Thus, such a network cannot be considered as a gold standard but only as a reference.

Chapter 5

Final Remarks

In this thesis, two extensions to Knapp and Kaderali [45] model lpNet were developed and tested, lpNet-dyn and lpNet-dyn2. The key difference between these extensions is that in lpNet-dyn2 we assume that inactive nodes cannot influence other nodes.

Both simulated data and experimental data were used to test the performance of lpNet-dyn/2 in terms of sensitivity, specificity, and precision values.

For simulated data, the models were tested in four different situations:

- inference of motifs found in biological networks, where both models are shown to fail to infer edges that activate already active nodes. Additionally, lpNet-dyn is shown to infer more FP edges than lpNet-dyn2, because the λ value is not high enough, while lpNet-dyn2 infers less FN edges due to the assumption that inactive nodes cannot influence other nodes;

- inference results for an increasing number of time points, where it is shown that better results are achieved for a number of time points similar to the number of different node state vectors. Yet, when knockdown experiments are performed, more time points can be repeated without introducing a bias towards the inference of edges that become active at the repeated time points;

- inference results for a growing number of knockdown experiments, where it is shown that more knockdown experiments are generally beneficial, although when few experiments are performed the obtained results depend on the specific nodes that are silenced;

- inference results for increasing levels of noise, where it is shown that the results are stable until the noise in the data causes active and inactive node values to

overlap. Yet, even for high noise levels, lpNet-dyn2 is able to perform generally better than random prediction;

For the last test, the performance of lpNet-dyn/2 was also compared to DDEPN and the original lpNet. Moreover, simulated data was generated under DDEPN assumptions and the two models were compared once more. In both situations, lpNet-dyn/2 had a better overall performance. However, this comparison is biased, since the way to include knockdown experiments in DDEPN leads to the inclusion of extra edges that need to be inferred by the model, and the use of prior knowledge for these edges is not straightforward. In short, lpNet-dyn/2 and DDEPN are suited for different scenarios. lpNet-dyn/2 is more appropriate for scenarios in which known nodes are inhibited, while DDEPN is better suited for scenarios in which the perturbations applied to the network are unknown and should be inferred by the model. Furthermore, the choice of one model over the other also depends on the assumptions about the network to be inferred, e.g., if we assume that a single incoming inhibiting connection from an active parent is enough to inhibit a node, then DDEPN should be chosen. If we assume that for a node to be inhibited by its parents, it needs to have more incoming inhibiting connections from active parents than activating connections, then lpNet-dyn/2 is the model to use. At last, DDEPN allows for the explicit inclusion of replicate measurements and stimuli experiments, which can be included in lpNet-dyn/2 only by encoding them as different knockdown experiments in which no nodes are silenced, e.g. if 3 stimuli experiments are performed, these can be encoded as 3 knockdown experiments, in which no nodes are silenced in any of them. Yet, lpNet-dyn/2 performance does not usually benefit from replicate experiments, which also increase the model's execution time.

A common problem to both lpNet-dyn and lpNet-dyn2 is the general difficulty in inferring edges that activate already active nodes. Yet, under our assumptions: that the expression of a protein does not depend on the number of incoming activating edges, this problem is a common problem in network inference and, in principle, can be solved by performing the appropriate knockdown experiments, e.g., given a node, silence all of its parent nodes except one, and repeat the procedure for all parent nodes.

In general, lpNet-dyn2 presents a more balanced performance, with noteworthy values of sensitivity in comparison to other models, especially when knockdown experiments are performed. Relative to lpNet-dyn in particular, the high sensitivity levels are due to a lower number of false negative edges, resulting from the assumption that

an inactive node cannot influence other nodes.

On the experimental datasets ERBB G1/S and HCC1954. lpNet-dyn performance was very poor, with no more than 4 non-null edges inferred for each dataset. On the other hand, lpNet-dyn2 inferred 16 non-null edges for each dataset, and literature support was found for most of these edges. Interestingly, some of the edges for which no literature support was found are still plausible, for instance, the inference of the edge CDK4 $\rightarrow$ CDK2. Since both of these proteins are involved in the phosphorylation of pRB during the cell cycle, it might be worth to investigate such connection.

Quantitatively, lpNet-dyn2 performed slightly better than lpNet on ERBB G1/S, and better than DDEPN on HCC1954. Comparing to random prediction, it performed significantly better in terms of specificity, precision, and accuracy measurements (maximum p-value of 10^{-3}), but worse in terms of sensitivity values.

A major difficulty of using lpNet-dyn2 to infer a network from experimental data is the definition of the threshold δ, that distinguishes an active node from an inactive one. This value is inferred from the data and, as shown in this dissertation, different criteria can be used to define it. Yet, different criteria can lead to similar quantitative results but different sets of inferred edges. Additionally, due to noisy data it might not be possible to determine the "optimal" value for δ.

Finally, note that only edges whose median value is greater than its MAD were considered as non-null, therefore only edges with more evidence from the data are considered, which leads to higher precision values. However, if the goal of inferring a network from an experimental dataset is to explore the existence of edges not yet confirmed experimentally, one can relax the median $>$ MAD criterion, or even remove it, and more non-null edges are considered to exist, which can be experimentally tested.

lpNet-dyn2 possesses two key advantages relatively to other network inference methods, and in particular to DDEPN, its execution time is short and the results are highly reproducible. The short running time is due to the efficiency of the simplex method, the core of our model. As for the results reproducibility, the only source of randomness in our model is the k-fold cross validation step. Since for each run of the model different sets of entries are removed from the observation matrix, this can lead to the inference of different networks. Yet, if the value of k is small enough compared to the dimension of the observation matrix, the results are highly reproducible, as the solution of the LP problem is always the same if the problem does not change.

In conclusion, from all the analyzed models, lpNet-dyn2 is the one that shows a more balanced performance – with especially higher sensitivity values when knock-down experiments are performed – , and produces reproducible results in a relatively short amount of time.

Appendices

Appendix A

Supplemental information

A.1 Artificial ten-node networks

A.1.1 Positive edges only

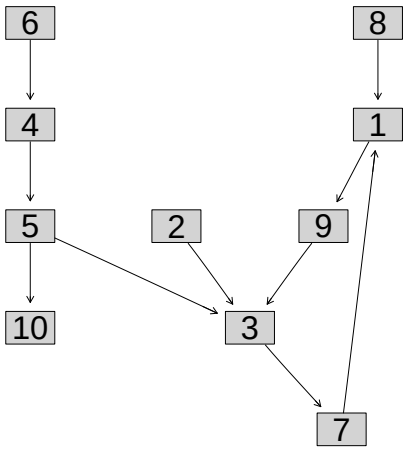


Figure A.1: Network 1 from the set of networks with positive edges only. Arrows represent activating edges.

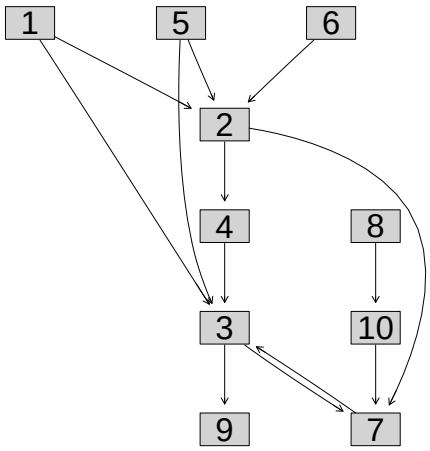


Figure A.2: Network 2 from the set of networks with positive edges only. Arrows represent activating edges.

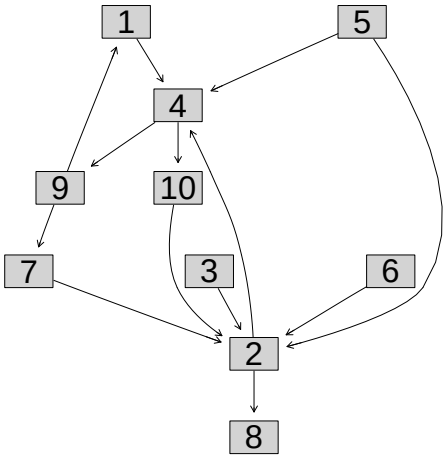


Figure A.3: Network 3 from the set of networks with positive edges only. Arrows represent activating edges.

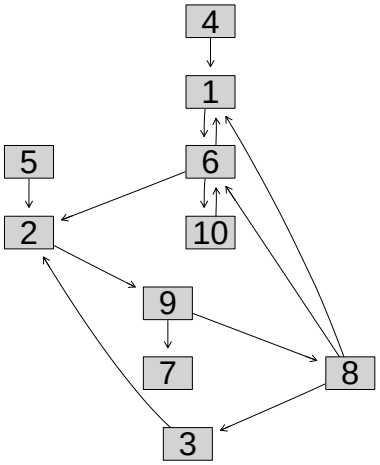


Figure A.4: Network 4 from the set of networks with positive edges only. Arrows represent activating edges.

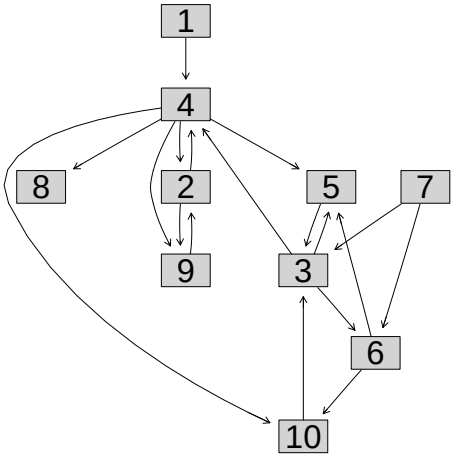


Figure A.5: Network 5 from the set of networks with positive edges only. Arrows represent activating edges.

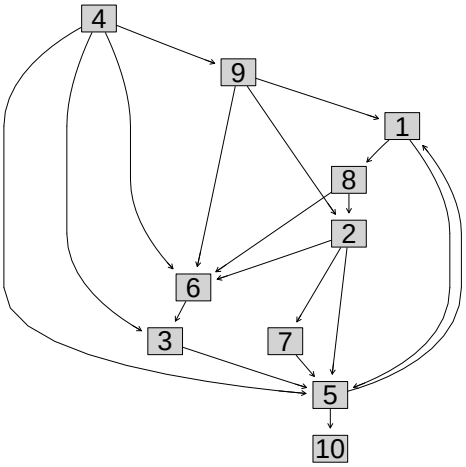


Figure A.6: Network 6 from the set of networks with positive edges only. Arrows represent activating edges.

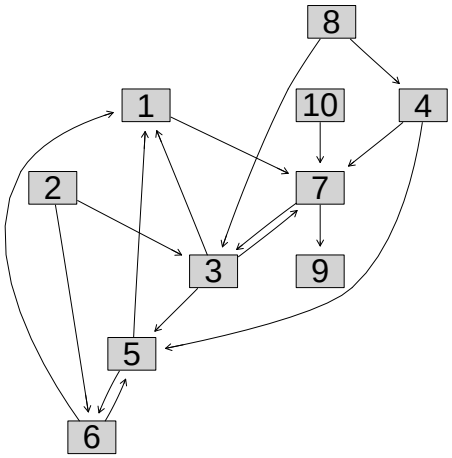


Figure A.7: Network 7 from the set of networks with positive edges only. Arrows represent activating edges.

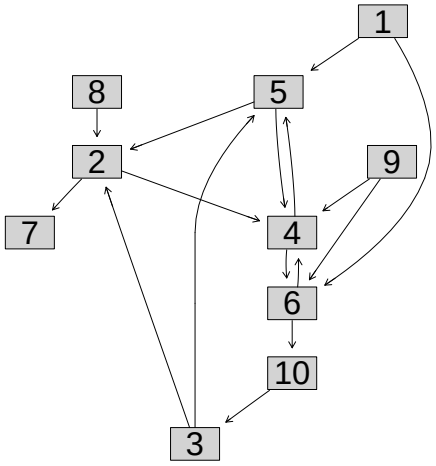


Figure A.8: Network 8 from the set of networks with positive edges only. Arrows represent activating edges.

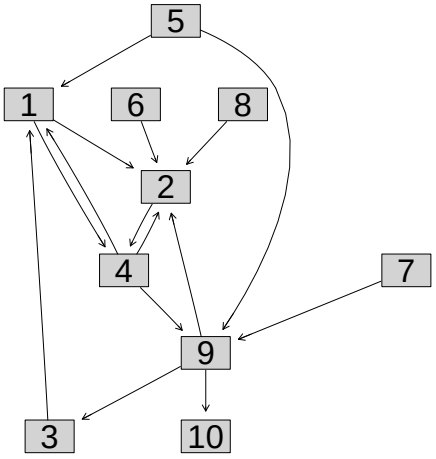


Figure A.9: Network 9 from the set of networks with positive edges only. Arrows represent activating edges.

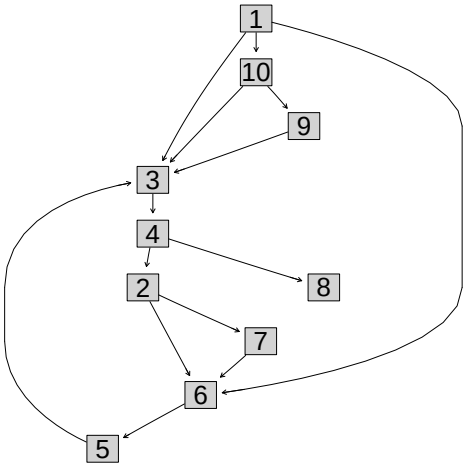


Figure A.10: Network 10 from the set of networks with positive edges only. Arrows represent activating edges.

A.1.2 Positive and negative edges

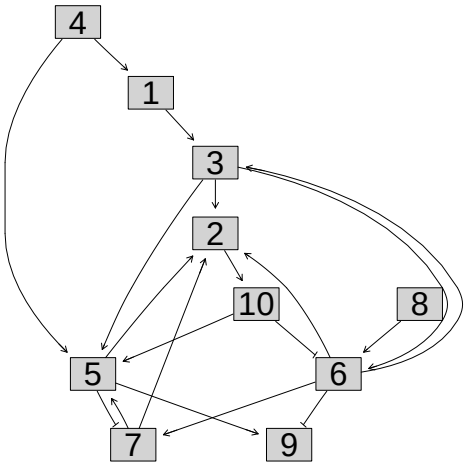


Figure A.11: Network 1 from the set of networks with positive and negative edges only. Arrows represent activating edges, while ⊣ represents inhibiting edges.

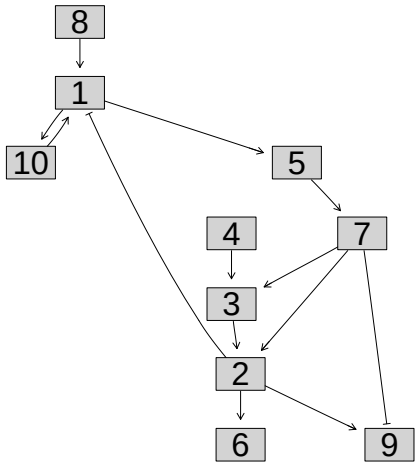


Figure A.12: Network 2 from the set of networks with positive and negative edges only. Arrows represent activating edges, while ⊣ represents inhibiting edges.

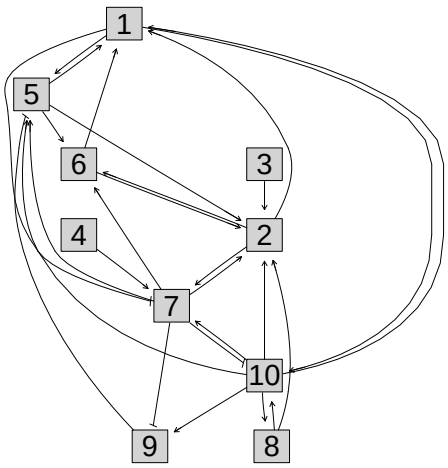


Figure A.13: Network 3 from the set of networks with positive and negative edges only. Arrows represent activating edges, while ⊣ represents inhibiting edges.

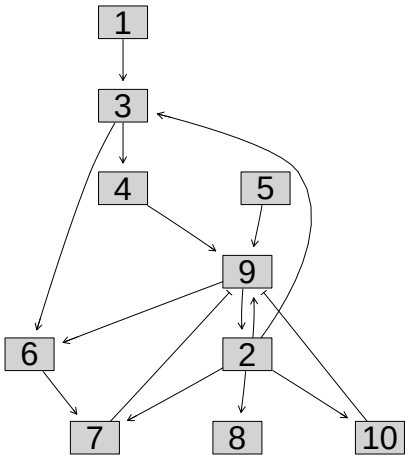


Figure A.14: Network 4 from the set of networks with positive and negative edges only. Arrows represent activating edges, while ⊣ represents inhibiting edges.

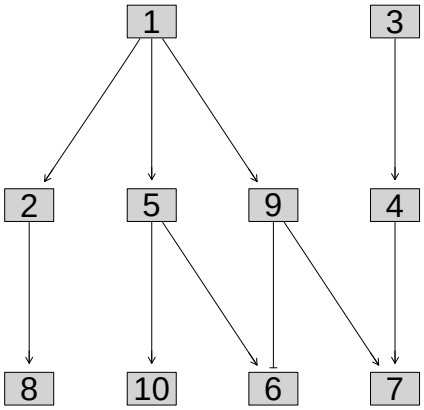


Figure A.15: Network 5 from the set of networks with positive and negative edges only. Arrows represent activating edges, while ⊣ represents inhibiting edges.

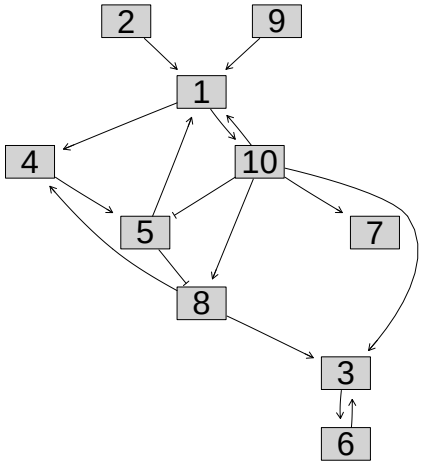


Figure A.16: Network 6 from the set of networks with positive and negative edges only. Arrows represent activating edges, while ⊣ represents inhibiting edges.

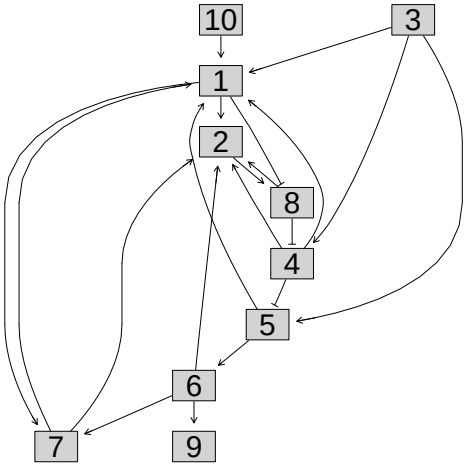


Figure A.17: Network 7 from the set of networks with positive and negative edges only. Arrows represent activating edges, while ⊣ represents inhibiting edges.

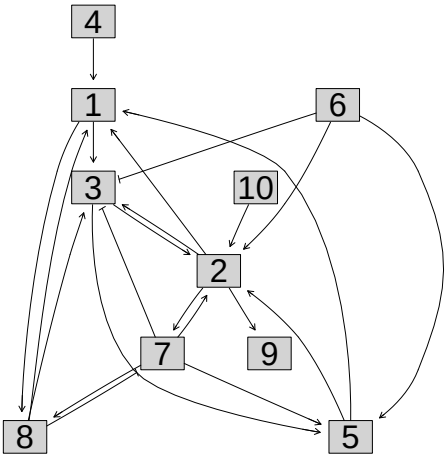


Figure A.18: Network 8 from the set of networks with positive and negative edges only. Arrows represent activating edges, while ⊣ represents inhibiting edges.

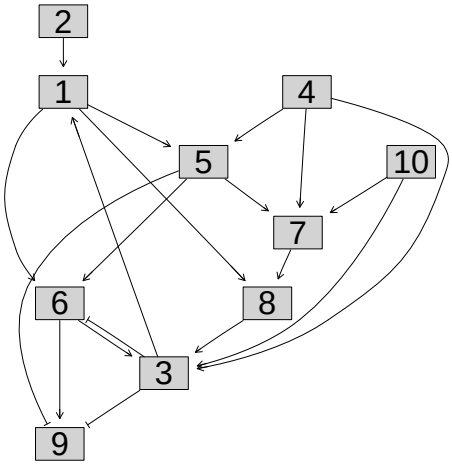


Figure A.19: Network 9 from the set of networks with positive and negative edges only. Arrows represent activating edges, while ⊣ represents inhibiting edges.

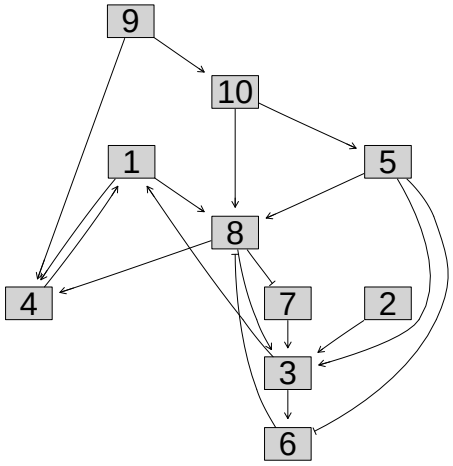


Figure A.20: Network 10 from the set of networks with positive and negative edges only. Arrows represent activating edges, while ⊣ represents inhibiting edges.

A.1.3 Test network

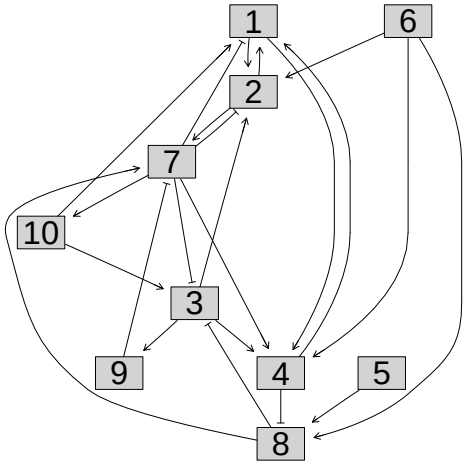


Figure A.21: Test network with positive and 30% negative edges.

A.2 Networks assembled with Ingenuity IPA

A.2.1 ERBB G1/S

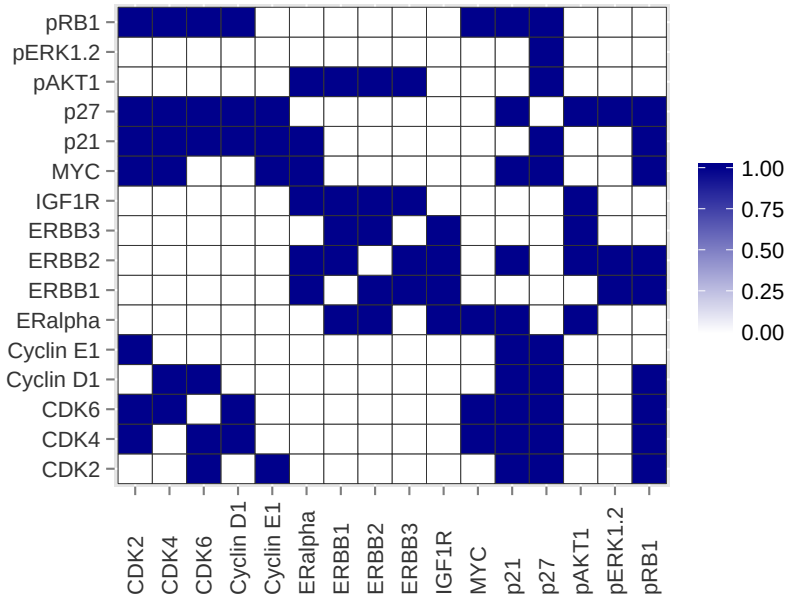


Figure A.22: Network assembled with Ingenuity IPA for ERBB G1/S dataset. Only direct edges for which experimental evidence exists in humans or *in vitro* were considered. Blue stands for existing edges, and white for non-existing edges

A.2.2 HCC1954

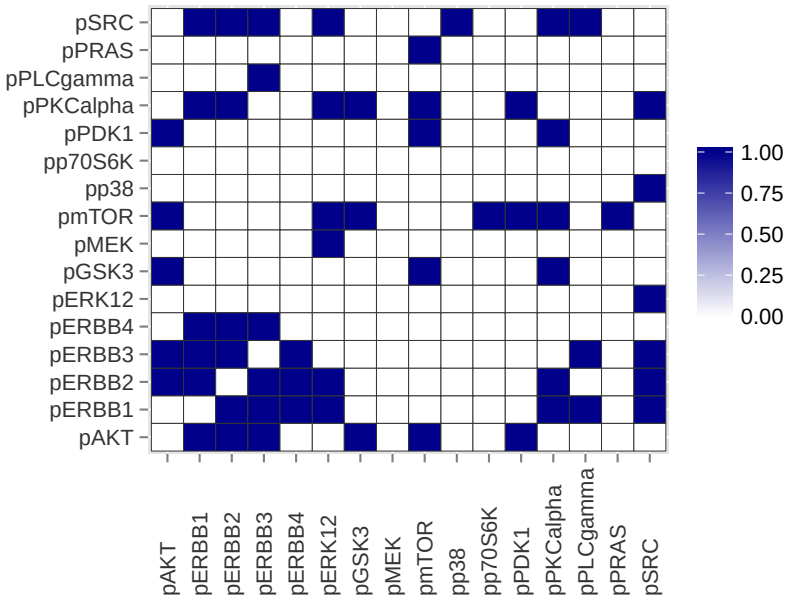


Figure A.23: Network assembled with Ingenuity IPA for HCC1954 dataset. Only direct edges for which experimental evidence exists in humans or *in vitro* were considered. Blue stands for existing edges, and white for non-existing edges

A.3 Other information

A.3.1 List of time points

T	Time points
2	(0, 6) / (1, 6)
3	1, 3, 6
4	1, 2, 4, 6
5	2, 3, 4, 5, 6
6	1, 2, 3, 4, 5, 6
7	1, 2, 3, 4, 5, 6, 6
8	1, 2, 3, 4, 4, 5, 5, 6
10	1, 2, 2, 3, 3, 3, 4, 5, 6, 6
12	1, 2, 3, 3, 3, 3, 4, 4, 4, 5, 6, 6

Table A.1: Time points t used for each number of time points T for which the performance of lpNet-dyn/2 was studied.

A.3.2 List of silenced nodes

K	Silenced nodes in single knockdowns	Silenced nodes in double knockdowns
1	-	-
2	8	-
3	7, 8	-
4	1, 7, 8	-
5	1, 2, 7, 8	-
6	1, 2, 7, 8, 10	-
7	1, 2, 4, 7, 8, 10	-
8	1, 2, 4, 5, 7, 8, 10	-
9	1, 2, 4, 5, 6, 7, 8, 10	-
10	1, 2, 3, 4, 5, 6, 7, 8, 10	-
11	1, 2, 3, 4, 5, 6, 7, 8, 9, 10	-
12	1, 2, 3, 4, 5, 6, 7, 8, 9, 10	(1, 4)
13	1, 2, 3, 4, 5, 6, 7, 8, 9, 10	(1, 4), (7, 10)
14	1, 2, 3, 4, 5, 6, 7, 8, 9, 10	(1, 4), (7, 10), (2, 3)
15	1, 2, 3, 4, 5, 6, 7, 8, 9, 10	(1, 4), (7, 10), (2, 3), (6,8)
16	1, 2, 3, 4, 5, 6, 7, 8, 9, 10	(1, 4), (7, 10), (2, 3), (6,8)

Table A.2: Silenced nodes for each number of knockdown experiments K. Pairs of nodes in between parenthesis refer to the nodes silenced in each double knockdown experiment, e.g., for K=15, the pair (7,10) means these two nodes were both silenced in the 13th knockdown experiment. For all values of K, no nodes are silenced in the 1st knockdown experiment.

A.3.3 List of latent nodes

nL	Latent nodes
1	8
2	1, 7
3	1, 9, 10
5	2, 3, 6, 7, 8

Table A.3: Nodes set as latent for each number of latent nodes nL for which the performance of lpNet-dyn2 is tested.

Appendix B

Supplemental Results

B.1 Prediction of Motifs in Biological Networks - extra results

In this section, we show the results obtained for the inference of the networks depicted in figures 4.1a-4.1f, both when considering an edge w_{ji} to exist if it is inferred at least 50% of the total times a network is inferred (100 in this case), no matter what its value is, and when w_{ji} is considered to exist if its median value is greater than its MAD. Although the results for the latter situation have already been presented in section 4.1.1, we present them again so that it is easier for the reader to see the differences.

Cascade

	$K = 1$		$K = 6$		
Edges	lpNet-dyn	lpNet-dyn2	lpNet-dyn	lpNet-dyn2	lpNet
$1 \rightarrow 2$	YES	YES	YES	YES	YES
$2 \rightarrow 3$	YES	YES	YES	YES	YES
$3 \rightarrow 4$	NO	YES	YES	YES	NO
$4 \rightarrow 5$	YES	YES	YES	YES	YES
FP	$1 \dashv 5$	-	-	-	-

Table B.1: The above table shows for each model, without ($K = 1$) and with knock-down experiments ($K = 6$), which edges are correctly inferred for the network in figure 4.1a, a cascade motif. Inferred false positive (FP) edges are shown in the last line, where $\dashv$ refers to an inhibiting edge. A non-null edge is considered as such when its median value is less than its MAD.

	$K = 1$		$K = 6$		
Edges	lpNet-dyn	lpNet-dyn2	lpNet-dyn	lpNet-dyn2	lpNet
$1 \to 2$	YES	YES	YES	YES	YES
$2 \to 3$	YES	YES	YES	YES	YES
$3 \to 4$	YES	YES	YES	YES	YES
$4 \to 5$	YES	YES	YES	YES	YES
FP	$1 \dashv 5$, $5 \dashv 4$	-	-	-	$1 \dashv 5$

Table B.2: The above table shows for each model, without ($K = 1$) and with knock-down experiments ($K = 6$), which edges are correctly inferred for the network in figure 4.1a, a cascade motif. Inferred false positive (FP) edges are shown in the last line, where $\dashv$ refers to an inhibiting edge. A non-null edge is considered as such if it is inferred at least in 50% of all inferred networks.

Fan-in

	$K = 1$		$K = 6$		
Edges	lpNet-dyn	lpNet-dyn2	lpNet-dyn	lpNet-dyn2	lpNet
$1 \to 3$	NO	NO	YES	YES	YES
$2 \to 4$	NO	NO	YES	YES	YES
$3 \to 5$	NO	NO	YES	YES	NO
$4 \to 5$	NO	NO	YES	YES	NO
FP	-	-	-	-	-

Table B.3: The above table shows for each model, without ($K = 1$) and with knock-down experiments ($K = 6$), which edges are correctly inferred for the network in figure 4.1b, which contains a fan-in motif. No false positive edges were inferred. A non-null edge is considered as such when its median value is less than its MAD.

	$K = 1$		$K = 6$		
Edges	lpNet-dyn	lpNet-dyn2	lpNet-dyn	lpNet-dyn2	lpNet
$1 \to 3$	NO	YES	YES	YES	YES
$2 \to 4$	YES	YES	YES	YES	YES
$3 \to 5$	NO	NO	YES	YES	NO
$4 \to 5$	NO	NO	YES	YES	NO
FP	$2 \to 3$	$1 \to 4$, $2 \to 3$	-	-	$3 \dashv 4$, $4 \dashv 3$

Table B.4: The above table shows for each model, without ($K = 1$) and with knock-down experiments ($K = 6$), which edges are correctly inferred for the network in figure 4.1b, which contains a fan-in motif. No false positive edges were inferred. A non-null edge is considered as such if it is inferred at least in 50% of all inferred networks.

Fan-out

	$K = 1$		$K = 6$		
Edges	lpNet-dyn	lpNet-dyn2	lpNet-dyn	lpNet-dyn2	lpNet
$1 \to 2$	YES	YES	YES	YES	YES
$1 \to 3$	YES	YES	YES	YES	YES
$2 \to 4$	NO	NO	YES	YES	YES
$3 \to 5$	NO	NO	YES	YES	YES
FP	-	-	-	-	-

Table B.5: The above table shows for each model, without ($K = 1$) and with knock-down experiments ($K = 6$), which edges are correctly inferred for the network in figure 4.1c, which contains a fan-out motif. No false positive edges were inferred. A non-null edge is considered as such when its median value is less than its MAD.

	$K = 1$		$K = 6$		
Edges	lpNet-dyn	lpNet-dyn2	lpNet-dyn	lpNet-dyn2	lpNet
$1 \to 2$	YES	YES	YES	YES	YES
$1 \to 3$	YES	YES	YES	YES	YES
$2 \to 4$	NO	NO	YES	YES	YES
$3 \to 5$	NO	NO	YES	YES	YES
FP	-	-	-	-	$4 \dashv 5$, $5 \dashv 4$

Table B.6: The above table shows for each model, without ($K = 1$) and with knock-down experiments ($K = 6$), which edges are correctly inferred for the network in figure 4.1c, which contains a fan-out motif. No false positive edges were inferred. A non-null edge is considered as such if it is inferred at least in 50% of all inferred networks.

Fan-in plus Fan-out

	$K = 1$		$K = 6$		
Edges	lpNet-dyn	lpNet-dyn2	lpNet-dyn	lpNet-dyn2	lpNet
$1 \rightarrow 2$	YES	YES	YES	YES	NO
$1 \rightarrow 3$	YES	YES	YES	YES	NO
$2 \rightarrow 4$	NO	YES	YES	YES	NO
$3 \rightarrow 4$	NO	NO	YES	YES	NO
$4 \rightarrow 5$	NO	YES	YES	YES	NO
FP	-	-	$1 \dashv 5$, $5 \dashv 2$, $5 \dashv 3$, $5 \dashv 4$	-	-

Table B.7: The above table shows for each model, without ($K = 1$) and with knock-down experiments ($K = 6$), which edges are correctly inferred for the network in figure 4.1d, which contains both a fan-in and a fan-out motif. Inferred false positive (FP) edges are shown in the last line, where $\dashv$ refers to an inhibiting edge. A non-null edge is considered as such when its median value is less than its MAD.

	$K = 1$		$K = 6$		
Edges	lpNet-dyn	lpNet-dyn2	lpNet-dyn	lpNet-dyn2	lpNet
$1 \rightarrow 2$	YES	YES	YES	YES	YES
$1 \rightarrow 3$	YES	YES	YES	YES	YES
$2 \rightarrow 4$	YES	YES	YES	YES	NO
$3 \rightarrow 4$	YES	YES	YES	YES	NO
$4 \rightarrow 5$	YES	YES	YES	YES	YES
FP	-	-	$1 \rightarrow 4$, $1 \dashv 5$, $3 \dashv 5$, $5 \dashv 2$, $5 \dashv 3$, $5 \dashv 4$	$3 \rightarrow 2$	$1 \rightarrow 4$

Table B.8: The above table shows for each model, without ($K = 1$) and with knock-down experiments ($K = 6$), which edges are correctly inferred for the network in figure 4.1d, which contains both a fan-in and a fan-out motif. Inferred false positive (FP) edges are shown in the last line, where $\dashv$ refers to an inhibiting edge. A non-null edge is considered as such if it is inferred at least in 50% of all inferred networks.

Feedback loop

	$K = 1$		$K = 6$		
Edges	lpNet-dyn	lpNet-dyn2	lpNet-dyn	lpNet-dyn2	lpNet
$1 \rightarrow 2$	YES	YES	YES	YES	YES
$2 \rightarrow 3$	YES	YES	YES	YES	YES
$3 \rightarrow 4$	YES	YES	YES	YES	YES
$3 \rightarrow 5$	YES	YES	YES	YES	YES
$4 \rightarrow 2$	NO	NO	NO	NO	NO
FP	-	-	-	-	-

Table B.9: The above table shows for each model, without ($K = 1$) and with knockdown experiments ($K = 6$), which edges are correctly inferred for the network in figure 4.1e, which contains a feedback loop. No false positive edges were inferred. A non-null edge is considered as such when its median value is less than its MAD.

	$K = 1$		$K = 6$		
Edges	lpNet-dyn	lpNet-dyn2	lpNet-dyn	lpNet-dyn2	lpNet
$1 \rightarrow 2$	YES	YES	YES	YES	YES
$2 \rightarrow 3$	YES	YES	YES	YES	YES
$3 \rightarrow 4$	YES	YES	YES	YES	YES
$3 \rightarrow 5$	YES	YES	YES	YES	YES
$4 \rightarrow 2$	NO	NO	NO	NO	NO
FP	-	-	-	-	-

Table B.10: The above table shows for each model, without ($K = 1$) and with knockdown experiments ($K = 6$), which edges are correctly inferred for the network in figure 4.1e, which contains a feedback loop. No false positive edges were inferred. A non-null edge is considered as such if it is inferred at least in 50% of all inferred networks.

Feedforward loop

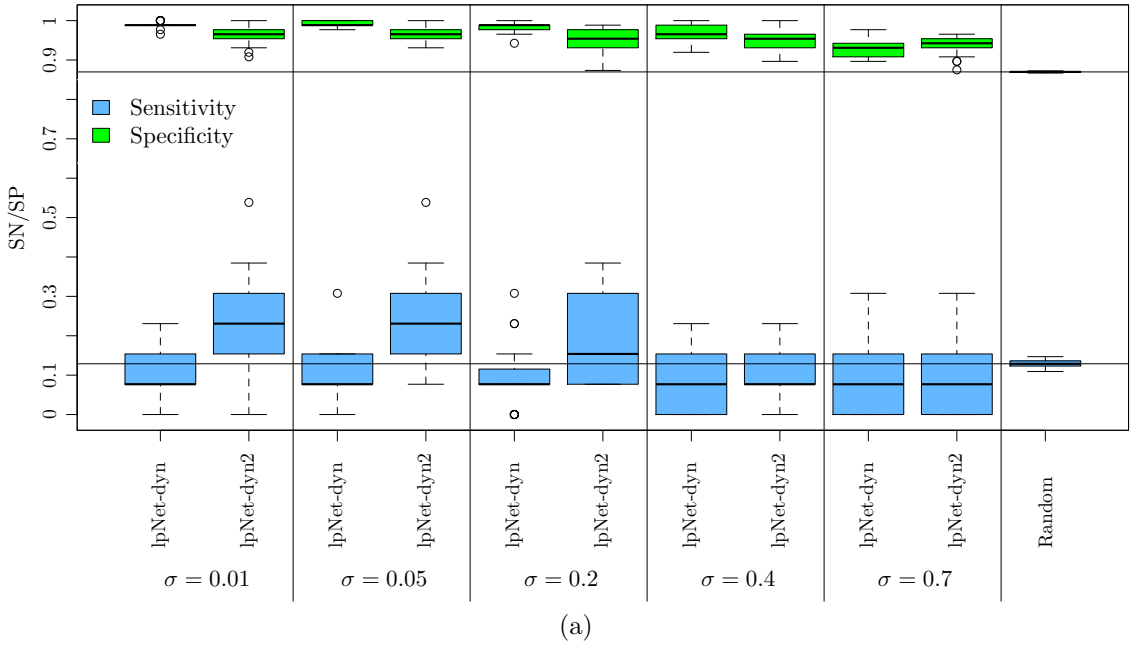
	$K = 1$		$K = 6$		
Edges	lpNet-dyn	lpNet-dyn2	lpNet-dyn	lpNet-dyn2	lpNet
$1 \to 2$	YES	YES	YES	YES	NO
$2 \to 3$	YES	YES	YES	YES	NO
$2 \to 4$	YES	YES	YES	YES	NO
$3 \to 5$	NO	YES	YES	YES	NO
$4 \to 3$	NO	NO	NO	NO	NO
FP	-	-	-	-	-

Table B.11: The above table shows for each model, without ($K = 1$) and with knockdown experiments ($K = 6$), which edges are correctly inferred for the network in figure 4.1f, which contains a feedforward loop. No false positive edges were inferred. A non-null edge is considered as such when its median value is less than its MAD.

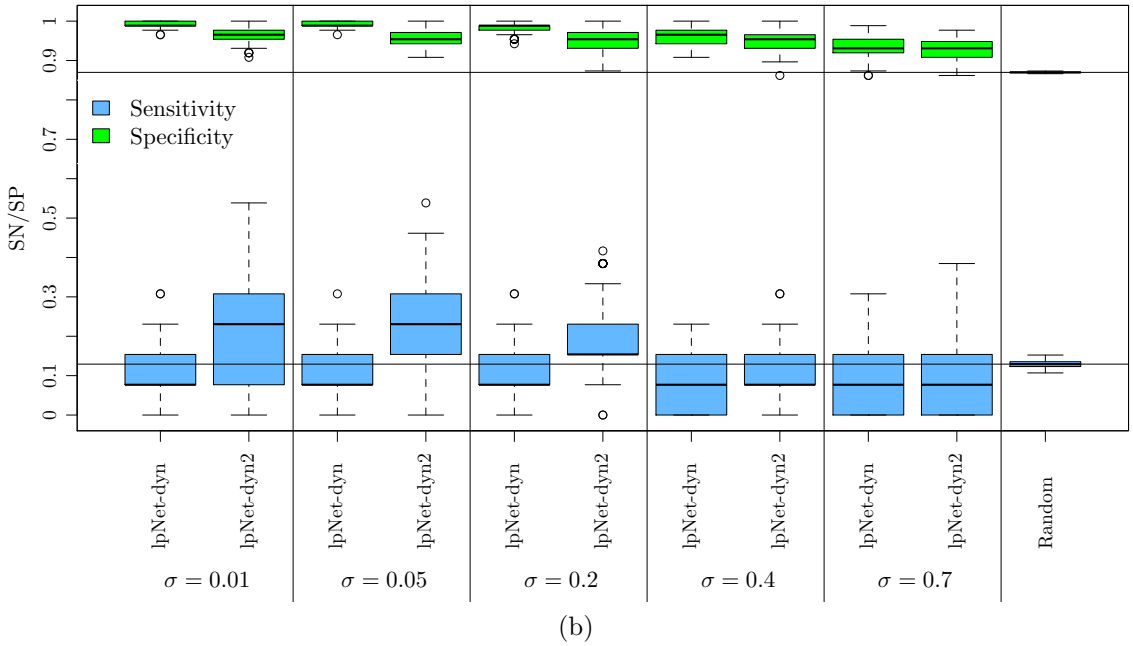
	$K = 1$		$K = 6$		
Edges	lpNet-dyn	lpNet-dyn2	lpNet-dyn	lpNet-dyn2	lpNet
$1 \to 2$	YES	YES	YES	YES	YES
$2 \to 3$	YES	YES	YES	YES	YES
$2 \to 4$	YES	YES	YES	YES	YES
$3 \to 5$	NO	YES	YES	YES	YES
$4 \to 3$	NO	NO	NO	NO	NO
FP	-	-	-	-	-

Table B.12: The above table shows for each model, without ($K = 1$) and with knockdown experiments ($K = 6$), which edges are correctly inferred for the network in figure 4.1f, which contains a feedback loop. No false positive edges were inferred. A non-null edge is considered as such if it is inferred at least in 50% of all inferred networks.

B.2 Influence of number of inferred networks on final result

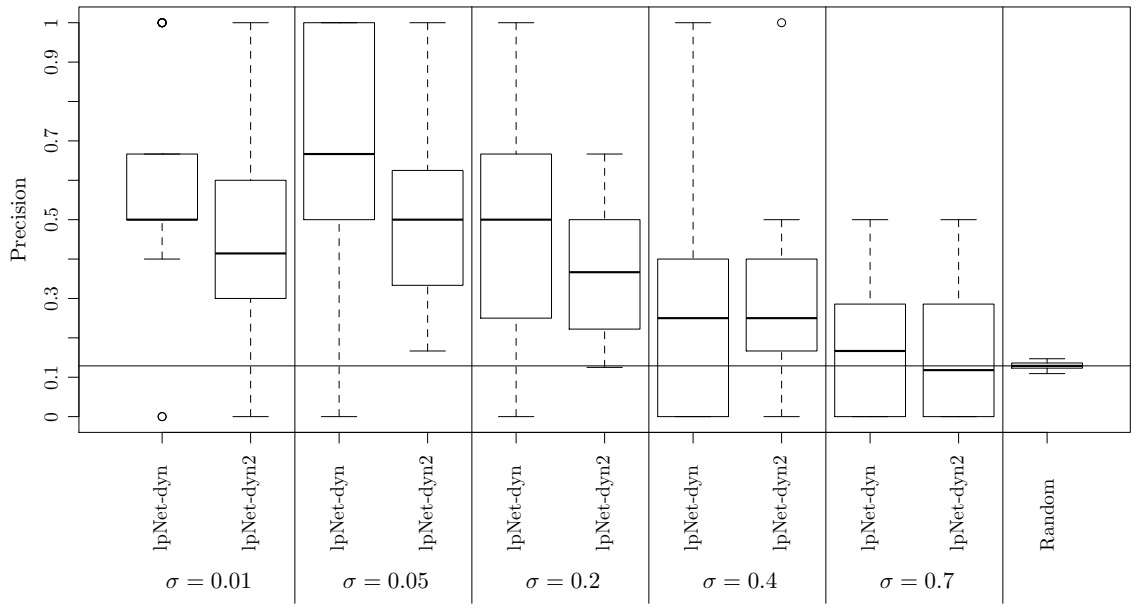


(a)

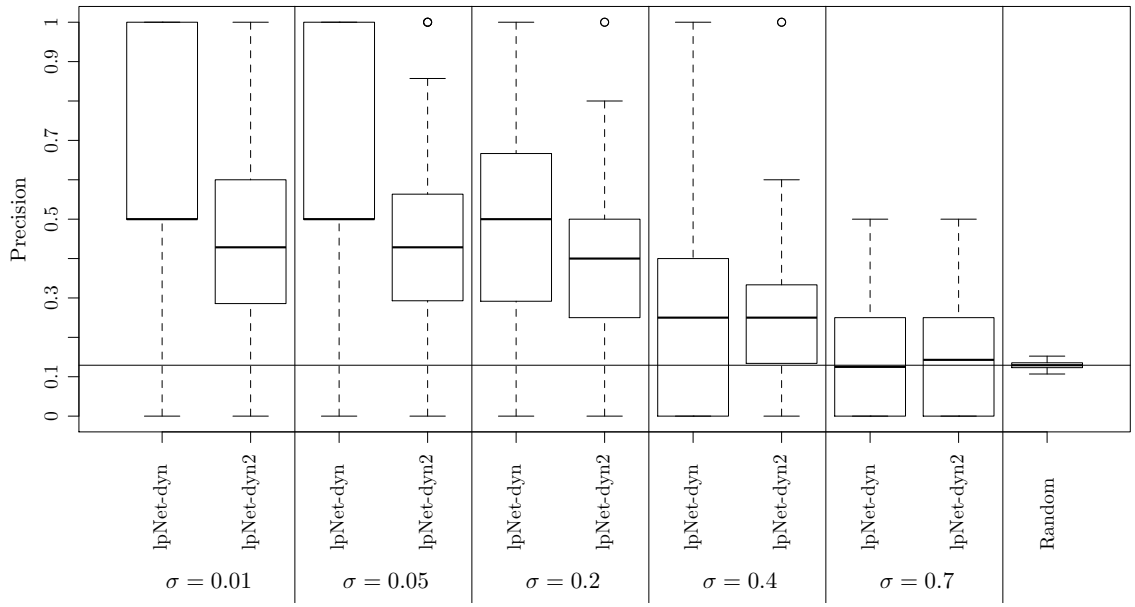


(b)

Figure B.1: Impact of the number of executions on SN/SP values for $K = 1$. SN/SP values for (a) 30 and (b) 100 executions of lpNet-dyn/2. The black horizontal line represents the median SP and SN value for random prediction.

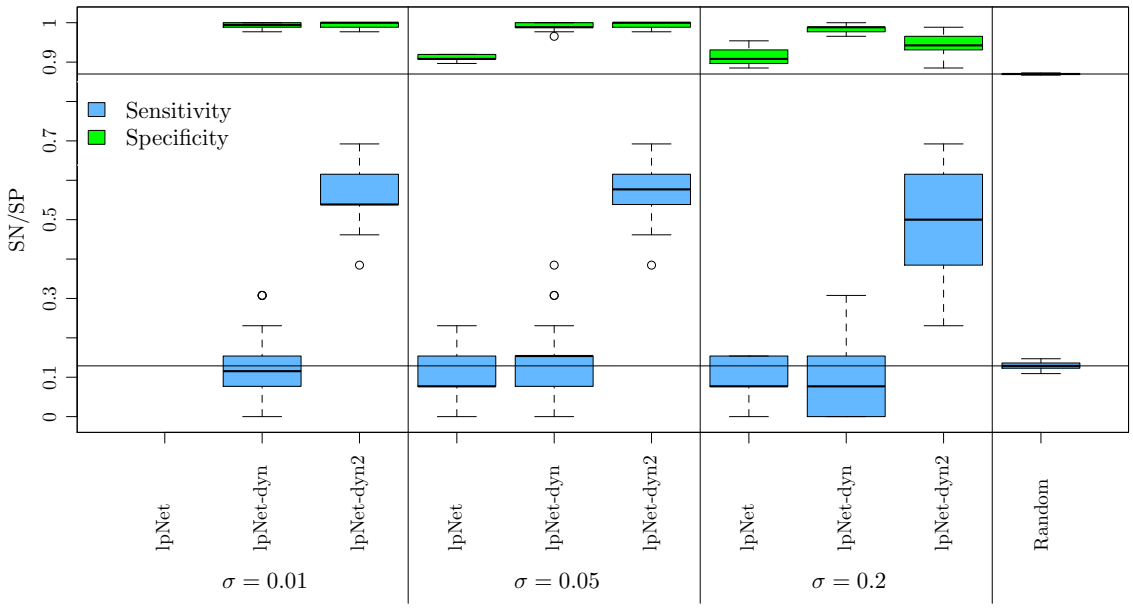


(a)

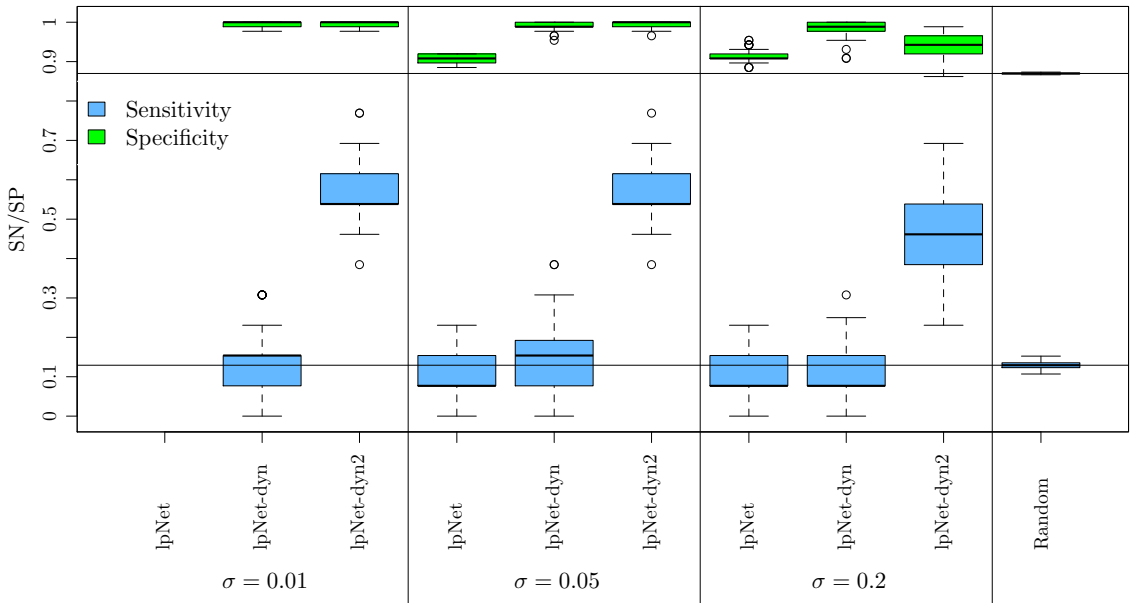


(b)

Figure B.2: Impact of the number of executions on PR values for $K = 1$. PR values for (a) 30 and (b) 100 executions of lpNet-dyn/2. The black horizontal line represents the median PR value for random prediction.

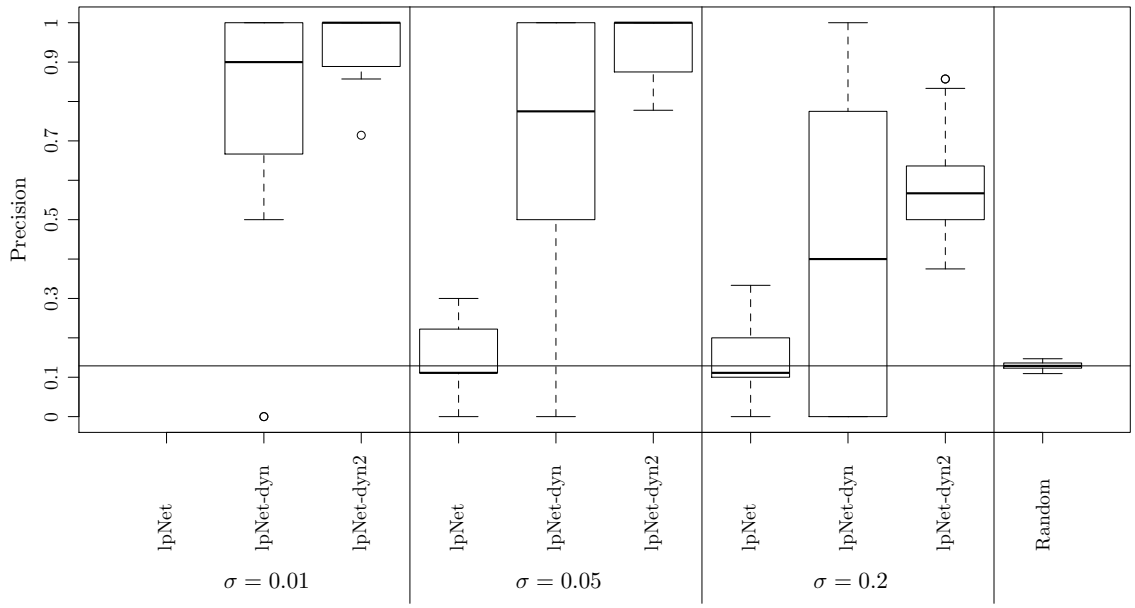


(a)

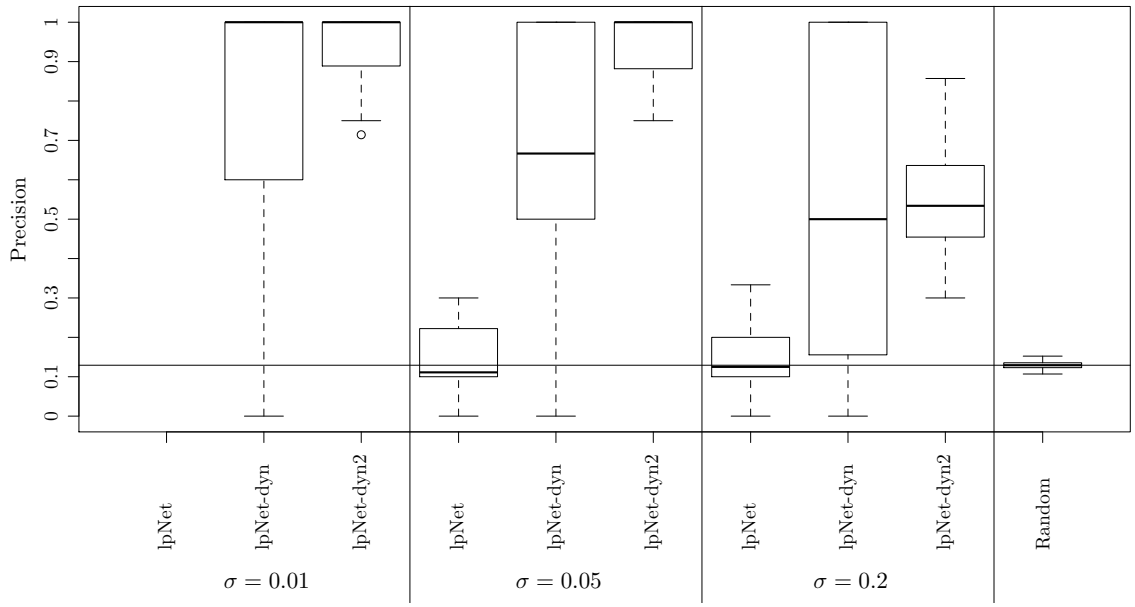


(b)

Figure B.3: Impact of the number of executions on SN/SP values for $K = 11$. SN/SP values for (a) 30 and (b) 100 executions of lpNet and lpNet-dyn/2. The black horizontal line represents the median SP and SN value for random prediction.



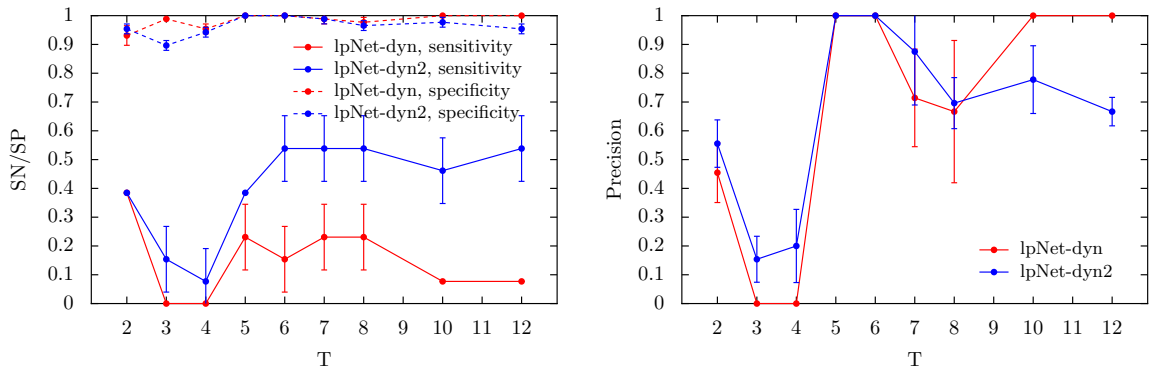
(a)



(b)

Figure B.4: Impact of the number of executions on PR values for $K = 11$. PR values for (a) 30 and (b) 100 executions of lpNet and lpNet-dyn/2. The black horizontal line represents the median PR value for random prediction.

B.3 Results for data with $T = 2$

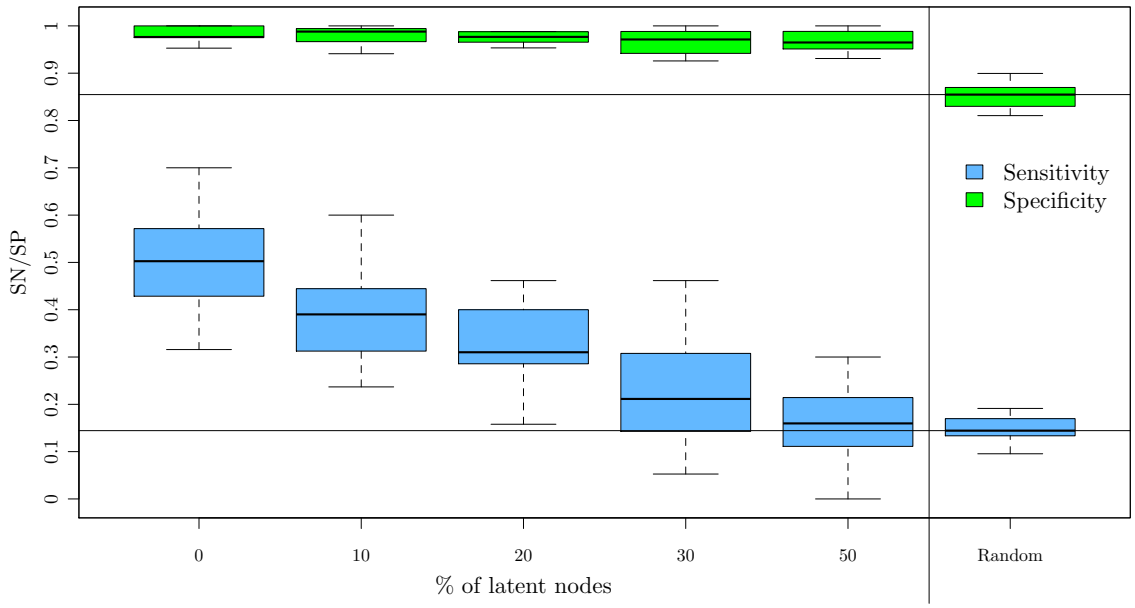


(a) (b)

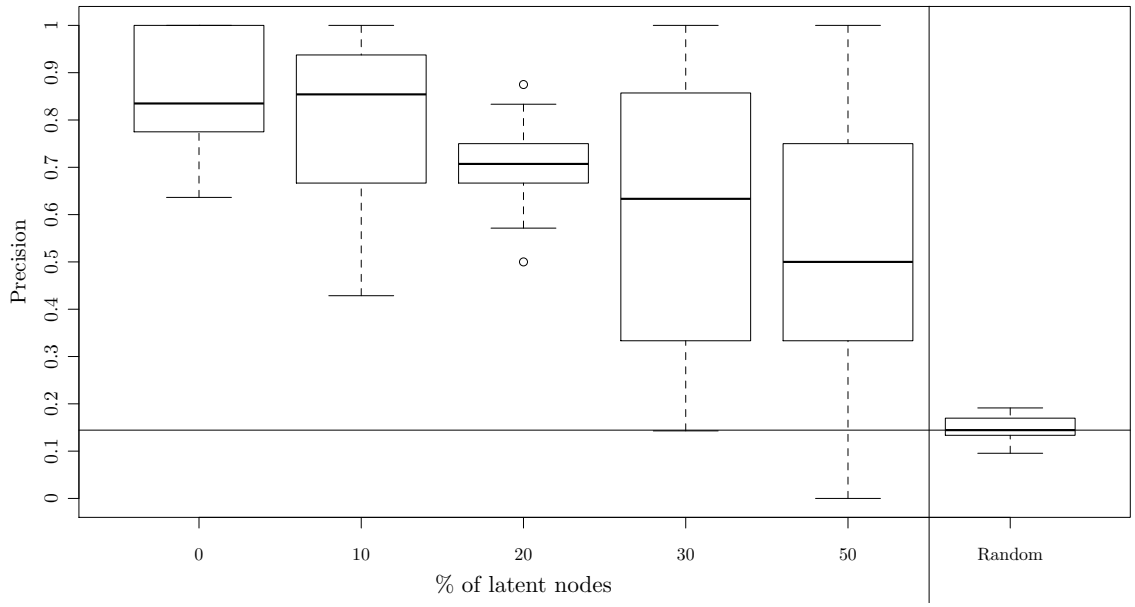
Figure B.5: Results with increasing number of time points for $K = 11$, when the time points used for $T = 2$ are $t = 5, 6$.

B.4 Results for an increasing number of latent nodes

In this section, we present the change in lpNet-dyn2 performance for a growing number of latent nodes, when $K = 11$: from no latent nodes up to 5 latent nodes, which corresponds to 50% of network's nodes. For each number of latent nodes (nL), the nodes to set as latent were sampled with no replacement. The list of latent nodes for each nL value is shown in table A.3. The results displayed in figure B.6 were obtained from executing each model for each network 30 times. The set of ten ten-node networks used is depicted in figures A.1-A.10. These networks contain only positive edges.



(a)



(b)

Figure B.6: (a) Shows the change in sensitivity (blue) and specificity (green) values with increasing number of latent nodes in 10 ten-node networks for each model, lpNet-dyn and lpNet-dyn2, when $K = 11$, while (b) shows the evolution of the precision values. The black horizontal line represents the median SN, SP, and PR values for random prediction.

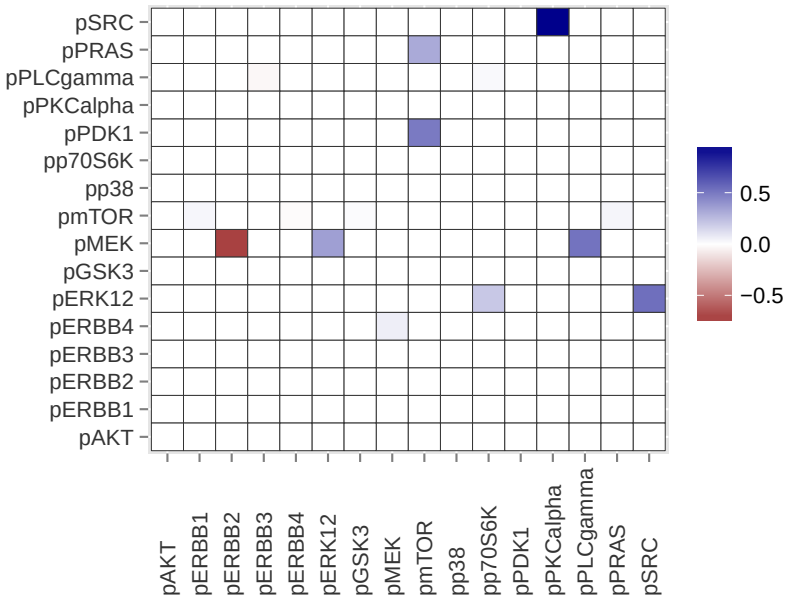
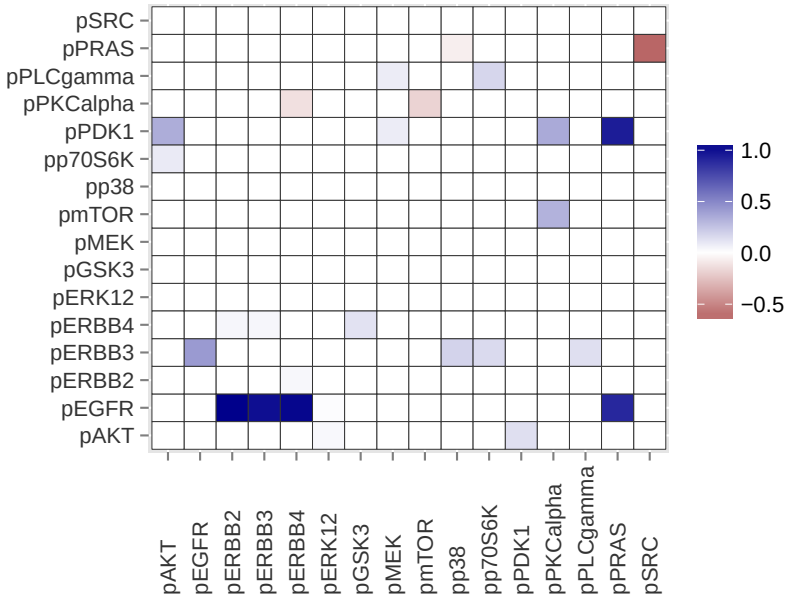


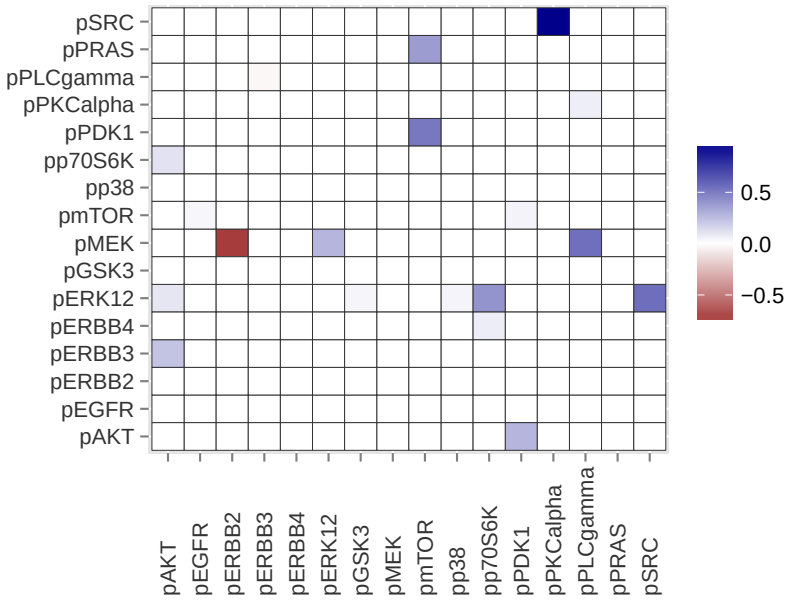
Figure B.7: Heat plot of the median value of inferred edge weights $w_{i,j}$ with lpNetdyn2 for the HCC1954 dataset, when using mclust to define the δ value. Blue represents positive edges, red represents negative edges, and white represents null edges.

B.5 Results for the HCC1954 with a different δ value

B.6 Results for the HCC1954 with prior knowledge



(a)



(b)

Figure B.8: Heat plot of the median value of inferred edge weights $w_{i,j}$ with lpNetdyn2 for the HCC1954 dataset, when using prior knowledge. Blue represents positive edges, red represents negative edges, and white represents null edges. (a) Shows the inferred edges for δ^1, and (b) shows the inferred edges for δ^2

References

[1] http://www.ingenuity.com/products/ipa, June 2013.

[2] Edward E Allen, Jacquelyn S Fetrow, Larry W Daniel, Stan J Thomas, and David J John. Algebraic dependency models of protein signal transduction networks from time-series data. *Journal of theoretical biology*, 238(2):317–30, January 2006.

[3] Benedict Anchang, Mohammad J Sadeh, Juby Jacob, Achim Tresch, Marcel O Vlad, Peter J Oefner, and Rainer Spang. Modeling the temporal interplay of molecular signaling and gene expression by using dynamic nested effects models. *Proceedings of the National Academy of Sciences of the United States of America*, 106(16):6447–52, April 2009.

[4] Mukesh Bansal, Giusy Della Gatta, and Diego di Bernardo. Inference of gene regulatory networks and compound mode of action from time course gene expression profiles. *Bioinformatics (Oxford, England)*, 22(7):815–22, April 2006.

[5] Christian Bender. *ddepn: Dynamic Deterministic Effects Propagation Networks: Infer signalling networks for timecourse RPPA data.*, 2012. R package version 2.1.2.

[6] Christian Bender, Frauke Henjes, Holger Fröhlich, Stefan Wiemann, Ulrike Korf, and Tim Beissbarth. Dynamic deterministic effects propagation networks: learning signalling pathways from longitudinal protein array data. *Bioinformatics (Oxford, England)*, 26(18):i596–602, September 2010.

[7] Christian Bender, Silvia Vd Heyde, Frauke Henjes, Stefan Wiemann, Ulrike Korf, and Tim Beissbarth. Inferring signalling networks from longitudinal data using sampling based approaches in the R-package 'ddepn'. *BMC bioinformatics*, 12(1):291, January 2011.

[8] B.M. Bolstad, R.A Irizarry, M. Astrand, and T.P. Speed. A comparison of normalization methods for high density oligonucleotide array data based on variance and bias. *Bioinformatics*, 19(2):185–193, 2003.

[9] K.-H. Borgwardt. The average number of pivot steps required by the simplex-method is polynomial. *Zeitschrift fÃŒr Operations Research*, 26(1):157–177, 1982.

[10] Jiguo Cao and Hongyu Zhao. Estimating dynamic models for gene regulation networks. *Bioinformatics (Oxford, England)*, 24(14):1619–24, July 2008.

[11] G.B. Dantzig and M.N. Thapa. *Linear Programming: 1: Introduction*. Linear Programming. Springer, 1997.

[12] Riet De Smet and Kathleen Marchal. Advantages and limitations of current network inference methods. *Nature reviews. Microbiology*, 8(10):717–29, October 2010.

[13] Zhiyong Ding, Jiyong Liang, Jin Li, Yiling Lu, Vathsala Ariyaratna, Zhimin Lu, Michael A. Davies, John K. Westwick, and Gordon B. Mills. Physical association of pdk1 with akt1 is sufficient for pathway activation independent of membrane localization and phosphatidylinositol 3 kinase. *PLoS ONE*, 5(3):e9910, 03 2010.

[14] Norbert Dojer, Anna Gambin, Andrzej Mizera, Bartek Wilczynski, and Jerzy Tiuryn. Applying dynamic bayesian networks to perturbed gene expression data. *BMC Bioinformatics*, 7(1):249, 2006.

[15] Erica M. Dutil, Alex Toker, and Alexandra C. Newton. Regulation of conventional protein kinase c isozymes by phosphoinositide-dependent kinase 1 (pdk-1). *Current Biology*, 8(25):1366 – 1375, 1998.

[16] H.A. Eiselt and C.L. Sandblom. *Linear programming and its Applications*. Springer-Verlag Berlin Heidelberg, 2007.

[17] Michel Berkelaar et al. *lpSolve: Interface to Lp_solve v. 5.5 to solve linear/integer programs*, 2011. R package version 5.6.6.

[18] Andrew Fire, SiQun Xu, Mary K. Montgomery, Steven A. Kostas, Samuel E. Driver, and Craig C. Mello. Potent and specific genetic interference by double-stranded rna in caenorhabditis elegans. *Nature*, 391:806–811, 1998.

[19] C Fraley, A E Raftery, T B Murphy, and Scrucca L. mclust version 4 for r: Normal mixture modeling for model-based clustering, classification, and density estimation. Technical Report 597, Department of Statistics, University of Washington, 2012.

[20] Chris Fraley and Adrian E. Raftery. Model-based clustering, discriminant analysis, and density estimation. *JOURNAL OF THE AMERICAN STATISTICAL ASSOCIATION*, 97:611–631, 2000.

[21] N Friedman, M Linial, I Nachman, and D Pe'er. Using Bayesian networks to analyze expression data. *Journal of computational biology : a journal of computational molecular cell biology*, 7(3-4):601–20, January 2000.

[22] Nir Friedman, Kevin Murphy, and S Russell. Learning the structure of dynamic probabilistic networks. *Proceedings of the Fourteenth ...*, 1998.

[23] Holger Froehlich, Mark Fellmann, Holger Sueltmann, Annemarie Poustka, and Tim Beissbarth. Large scale statistical inference of signaling pathways from RNAi and microarray data. *BMC bioinformatics*, 8:386, January 2007.

[24] Holger Froehlich, Florian Markowetz, Achim Tresch, Theresa Niederberger, Christian Bender, Matthias Maneck, Claudio Lottaz, and Tim Beissbarth. *nem: Nested Effects Models to reconstruct phenotypic hierarchies*. R package version 2.32.1.

[25] H Frohlich, M Fellmann, H Sultmann, A Poustka, and T BeiSZbarth. Large scale statistical inference of signaling pathways from rnai and microarray data. *BMC Bioinformatics*, 8:386, 2007.

[26] H Frohlich, M Fellmann, H Sultmann, A Poustka, and T BeiSZbarth. Estimating large scale signaling networks through nested effect models with intervention effects from microarray data. *Bioinformatics*, 24:2650–2656, 2008.

[27] Holger Fröhlich, Paurush Praveen, and Achim Tresch. Fast and efficient dynamic nested effects models. *Bioinformatics (Oxford, England)*, 27(2):238–44, January 2011.

[28] Holger Fröhlich, Ozgür Sahin, Dorit Arlt, Christian Bender, and Tim Beissbarth. Deterministic Effects Propagation Networks for reconstructing protein signaling networks from multiple interventions. *BMC bioinformatics*, 10(1):322, January 2009.

[29] Holger Frohlich, Ozgur Sahin, Dorit Arlt, Christian Bender, and Tim BeiSZbarth. Deterministic effects propagation networks for reconstructing protein signaling networks from multiple interventions. *BMC Bioinformatics*, 10(1):322, 2009.

[30] T Futamura, K Toyooka, S Iritani, K Niizato, R Nakamura, K Tsuchiya, T Someya, a Kakita, H Takahashi, and H Nawa. Abnormal expression of epidermal growth factor and its receptor in the forebrain and serum of schizophrenic patients. *Molecular psychiatry*, 7(7):673–82, January 2002.

[31] Timothy S Gardner, Diego di Bernardo, David Lorenz, and James J Collins. Inferring genetic networks and identifying compound mode of action via expression profiling. *Science (New York, N.Y.)*, 301(5629):102–5, July 2003.

[32] M Gassmann, F Casagranda, D Orioli, H Simon, C Lai, R Klein, and G Lemke. Aberrant neural and cardiac development in mice lacking the erbb4 neuregulin receptor. *Nature*, 378:390–394, 1995.

[33] Marco Grzegorczyk. A Non-Homogeneous Dynamic Bayesian Network with Sequentially Coupled Interaction Parameters for Applications in Systems and Synthetic Biology A Non-Homogeneous Dynamic Bayesian Network with Sequentially Coupled Interaction Parameters for Applications in S. 11(4), 2012.

[34] Marco Grzegorczyk and Dirk Husmeier. Non-stationary continuous dynamic bayesian networks. In *Advances in Neural Information Processing Systems (NIPS)*, 2009.

[35] Marco Grzegorczyk and Dirk Husmeier. Non-homogeneous dynamic bayesian networks forÂ continuous data. *Machine Learning*, 83(3):355–419, February 2011.

[36] Saad Haider and Ranadip Pal. Boolean network inference from time series data incorporating prior biological knowledge. *BMC Genomics*, 13(Suppl 6):S9, 2012.

[37] Seiya Imoto, Tomoyuki Higuchi, Takao Goto, Kousuke Tashiro, Satoru Kuhara, and Satoru Miyano. Combining microarrays and biological knowledge for estimating gene networks via bayesian networks. *Journal of Bioinformatics and Computational Biology*, 02(01):77–98, 2004.

[38] Yuriko Iwakura and Hiroyuki Nawa. Erbb1-4-dependent egf/neuregulin signals and their cross talk in the central nervous system: pathological implications in schizophrenia and parkinson's disease. *Frontiers in cellular neuroscience*, 7(February):4, January 2013.

[39] Yuriko Iwakura, Ying-shan Piao, Makoto Mizuno, Nobuyuki Takei, Akiyoshi Kakita, Hitoshi Takahashi, and Hiroyuki Nawa. Influences of dopaminergic lesion on epidermal growth factor-erbb signals in parkinson's disease and its model: neurotrophic implication in nigrostriatal neurons. *Journal of Neurochemistry*, 93(4):974–983, 2005.

[40] Lars Kaderali, Eva Dazert, Ulf Zeuge, Michael Frese, and Ralf Bartenschlager. Reconstructing signaling pathways from rnai data using probabilistic boolean threshold networks. *Bioinformatics*, 25(17):2229–2235, 2009.

[41] S a Kauffman. Metabolic stability and epigenesis in randomly constructed genetic nets. *Journal of theoretical biology*, 22(3):437–67, March 1969.

[42] Hans a Kestler, Christian Wawra, Barbara Kracher, and Michael Kühl. Network modeling of signal transduction: establishing the global view. *BioEssays : news and reviews in molecular, cellular and developmental biology*, 30(11-12):1110–25, November 2008.

[43] Shuhei Kimura, Kaori Ide, Aiko Kashihara, Makoto Kano, Mariko Hatakeyama, Ryoji Masui, Noriko Nakagawa, Shigeyuki Yokoyama, Seiki Kuramitsu, and Akihiko Konagaya. Inference of s-system models of genetic networks using a cooperative coevolutionary algorithm. *Bioinformatics*, 21(7):1154–1163, 2005.

[44] Bettina Knapp. *RNA Interference Data: from a Statistical Analysis to Network Inference*. PhD thesis, Ruprecht Karl University of Heidelberg, 2012.

[45] Bettina Knapp and Lars Kaderali. Reconstruction of cellular signal transduction networks using perturbation assays and linear programming. *PLoS ONE*, 8(7):e69220, 07 2013.

[46] Kristina S. Kovacina, Grace Y. Park, Sun Sik Bae, Andrew W. Guzzetta, Erik Schaefer, Morris J. Birnbaum, and Richard A. Roth. Identification of a proline-rich akt substrate as a 14-3-3 binding partner. *Journal of Biological Chemistry*, 278(12):10189–10194, 2003.

[47] Sophie Lèbre, Jennifer Becq, Frédéric Devaux, Michael P H Stumpf, and Gaëlle Lelandais. Statistical inference of the time-varying structure of gene-regulation networks. *BMC systems biology*, 4:130, January 2010.

[48] Robert D Leclerc. Survival of the sparsest: robust gene networks are parsimonious. *Molecular systems biology*, 4:213, January 2008.

[49] Kuo-Fen Lee, H Simon, H Chen, B Bates, Mien-Chie Hung, and C Hauser. Requirement for neuregulin receptor erbb2 in neural and cardiac development. *Nature*, 378:394–398, 1995.

[50] Jüri Lember and Alexey Koloydenko. The adjusted Viterbi training for hidden Markov models. *Bernoulli*, 14(1):180–206, February 2008.

[51] L. Liu, Y. Xie, and L. Lou. Cyclic amp inhibition of proliferation of hepatocellular carcinoma cells is mediated by akt. *Cancer Biol Ther*, 4(11):1240–7, 2005.

[52] Subbareddy Maddika, Sudharsana Rao Ande, Soumya Panigrahi, Ted Paranjothy, Kazimierz Weglarczyk, Anne Zuse, Mehdi Eshraghi, Kamala D. Manda, Emilia Wiechec, and Marek Los. Cell survival, cell death and cell cycle pathways are interconnected: Implications for cancer therapy. *Drug Resistance Updates*, 10:13–29, 2007.

[53] H. B. Mann and D. R. Whitney. On a test of whether one of two random variables is stochastically larger than the other. *Annals of Mathematical Statistics*, 18:50–60, 1947.

[54] Florian Markowetz, Jacques Bloch, and Rainer Spang. Non-transcriptional pathway features reconstructed from secondary effects of rna interference. *Bioinformatics (Oxford, England)*, 21(21):4026–32, November 2005.

[55] Florian Markowetz, Dennis Kostka, Olga G Troyanskaya, and Rainer Spang. Nested effects models for high-dimensional phenotyping screens. *Bioinformatics (Oxford, England)*, 23(13):i305–12, July 2007.

[56] Florian Markowetz and Rainer Spang. Inferring cellular networks–a review. *BMC bioinformatics*, 8 Suppl 6:S5, January 2007.

[57] Shawn Martin, Zhaoduo Zhang, Anthony Martino, and Jean-Loup Faulon. Boolean dynamics of genetic regulatory networks inferred from microarray time series data. *Bioinformatics*, 23(7):866–874, 2007.

[58] D Meyer and C Birchmeier. Multiple essential functions of neuregulin in development. *Nature*, 378:386–390, 1995.

[59] D.O. Morgan. *The Cell Cycle: Principles of Control*. Primers in biology. New Science Press, 2007.

[60] L Morris, K E Allen, and N B La Thangue. Regulation of e2f transcription by cyclin e-cdk2 kinase mediated through p300/cbp co-activators. *Nature cell biology*, 2(4):232–9, April 2000.

[61] D Neise, D Sohn, W Budach, and RU Janicke. Evidence for a differential modulation of p53-phosphorylating kinases by the cyclin-dependent kinase inhibitor p21waf1/cip1. *Cell Cycle*, 9:3575–3583, 2010.

[62] Sven Nelander, Weiqing Wang, Björn Nilsson, Qing-Bai She, Christine Pratilas, Neal Rosen, Peter Gennemark, and Chris Sander. Models from experiments: combinatorial drug perturbations of cancer cells. *Molecular systems biology*, 4(216):216, January 2008.

[63] Manuel Nieto-Sampedro, Beatriz Valle-Argos, Diego Gomez-Nicola, Alfonso Fernandez-Mayoralas, and Manuel Nieto-DÃaz. Inhibitors of glioma growth that reveal the tumour to the immune system. *Clinical Medicine Insights: Oncology*, 5:265–314, 09 2011.

[64] Hiroshi Okabe, Sang-Hyun Lee, Janyaporn Phuchareon, Donna G. Albertson, Frank McCormick, and Osamu Tetsu. A critical role for fbxw8 and mapk in cyclin d1 degradation and cancer cell proliferation. *PLoS ONE*, 1(1):e128, 12 2006.

[65] C P Paweletz, L Charboneau, V E Bichsel, N L Simone, T Chen, J W Gillespie, M R Emmert-Buck, M J Roth, E F Petricoin III, and L a Liotta. Reverse phase protein microarrays which capture disease progression show activation of pro-survival pathways at the cancer invasion front. *Oncogene*, 20(16):1981–9, April 2001.

[66] D. Pe'er, A. Regev, G. Elidan, and N. Friedman. Inferring subnetworks from perturbed expression profiles. *Bioinformatics*, 17(Suppl 1):S215–S224, June 2001.

[67] Dana Pe'er. Bayesian network analysis of signaling networks: A primer. *Sci. STKE*, 2005(281):pl4, 2005.

[68] R S Pellish, a Nasir, B Ramratnam, and S F Moss. Review article: Rna interference–potential therapeutic applications for the gastroenterologist. *Alimentary pharmacology & therapeutics*, 27(9):715–23, May 2008.

[69] Laura Pentassuglia and Douglas B. Sawyer. Erbb/integrin signaling interactions in regulation of myocardial cell-cell and cell-matrix interactions. *Biochimica et Biophysica Acta (BBA) - Molecular Cell Research*, 1833(4):909 – 916, 2013. <ce:title>Cardiomyocyte Biology: Cardiac Pathways of Differentiation, Metabolism and Contraction</ce:title>.

[70] D J Riese, T M Van Raaij, G D Plowman, G C Andrews, F Stern, David J Riese Ii, and M Tom. The cellular response to neuregulins is governed by complex interactions of the erbB receptor family . The Cellular Response to Neuregulins Is Governed by Complex Interactions of the erbB Receptor Family Downloaded from http://mcb.asm.org/ on August 31 ,. *Molecular and Cellular Biology*, 15(10):5570–5776, 1995.

[71] Joshua W Robinson and Alexander J Hartemink. Learning Non-Stationary Dynamic Bayesian Networks. *Journal of Machine Learning Research*, 11:3647–3680, 2010.

[72] Joshua W. Robinson and Er J. Hartemink. Non-stationary dynamic bayesian networks. In *Advances in Neural Information Processing Systems 21*, pages 1369–1376. Morgan Kaufmann Publishers, 2009.

[73] Karen Sachs, Omar Perez, Dana Pe'er, Douglas a Lauffenburger, and Garry P Nolan. Causal protein-signaling networks derived from multiparameter single-cell data. *Science (New York, N.Y.)*, 308(5721):523–9, April 2005.

[74] Ozgür Sahin, Holger Fröhlich, Christian Löbke, Ulrike Korf, Sara Burmester, Meher Majety, Jens Mattern, Ingo Schupp, Claudine Chaouiya, Denis Thieffry, Annemarie Poustka, Stefan Wiemann, Tim Beissbarth, and Dorit Arlt. Modeling ERBB receptor-regulated G1/S transition to find novel targets for de novo trastuzumab resistance. *BMC systems biology*, 3(1):1, January 2009.

[75] Francesco Sambo, Marco a Montes de Oca, Barbara Di Camillo, Gianna Toffolo, and Thomas Stützle. MORE: mixed optimization for reverse engineering–an application to modeling biological networks response via sparse systems of nonlinear differential equations. *IEEE/ACM transactions on computational biology and bioinformatics / IEEE, ACM*, 9(5):1459–71, 2012.

[76] Michael A. Savageau. Biochemical systems analysis: I. some mathematical properties of the rate law for the component enzymatic reactions. *Journal of Theoretical Biology*, 25(3):365 – 369, 1969.

[77] Michael A. Savageau. Biochemical systems analysis: Ii. the steady-state solutions for an n-pool system using a power-law approximation. *Journal of Theoretical Biology*, 25(3):370 – 379, 1969.

[78] Thomas Schaffter, Daniel Marbach, and Dario Floreano. GeneNetWeaver: in silico benchmark generation and performance profiling of network inference methods. *Bioinformatics (Oxford, England)*, 27(16):2263–70, August 2011.

[79] Lawrence A. Scheving, Mary C. Stevenson, Xiuqi Zhang, and William E. Russell. Cultured rat hepatocytes upregulate akt and erk in an erbb-2-dependent manner. *American Journal of Physiology - Gastrointestinal and Liver Physiology*, 295(2):G322–G331, 2008.

[80] S. S. Shapiro and M. B. Wilk. An analysis of variance test for normality (complete samples). *Biometrika*, 52(3-4):591–611, 1965.

[81] Shai S Shen-Orr, Ron Milo, Shmoolik Mangan, and Uri Alon. Network motifs in the transcriptional regulation network of escherichia coli. *Nature genetics*, 31(1):64–8, May 2002.

[82] Charles J. Sherr and James M. Roberts. Cdk inhibitors: positive and negative regulators of g1-phase progression. *Genes & Development*, 13(12):1501–1512, 1999.

[83] Student. The probable error of a mean. *Biometrika*, VI:1–25, 1908.

[84] Yoshihiro Taniyama, David S Weber, Petra Rocic, Lula Hilenski, Marjorie L Akers, Jongsun Park, Brian A Hemmings, R Wayne Alexander, and Kathy K Griendling. Pyk2- and src-dependent tyrosine phosphorylation of pdk1 regulates focal adhesions. *Molecular and Cellular Biology*, 23(22):8019–8029, 2003.

[85] R Development Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2012. ISBN 3-900051-07-0.

[86] John F Timms, Sarah L White, Michael J O Hare, and Michael D Waterfield. Effects of erbb-2 overexpression on mitogenic signalling and cell cycle progression in human breast luminal epithelial cells. pages 6573–6586, 2002.

[87] A Tresch and F Markowetz. Structure learning in nested effects models. *Statistical Applications in Genetics and Molecular Biology*, 7:Article 9, 2008.

[88] Heth R Turnquist, Jon Cardinal, Camila Macedo, Brian R Rosborough, Tina L Sumpter, A David, Diana Metes, Angus W Thomson, Washington Dc, Diana Metes, and Angus W Thomson. mtor and gsk-3 shape the cd4+ t-cell stimulatory and differentiation capacity of myeloid dcs after exposure to lps. *Blood*, 115:4758–4769, 2010.

[89] E Tzahar, H Waterman, X Chen, G Levkowitz, D Karunagaran, B J Ratzkin, Y Yarden, Eldad Tzahar, Hadassa Waterman, Xiomei Chen, G I L Levkowitz, Devarajan Karunagaran, Sara Lavi, and Barry J Ratzkin. A hierarchical network of interreceptor interactions determines signal transduction by neu differentiation factor / neuregulin and epidermal growth factor . a hierarchical network of interreceptor interactions determines signal transduction by neu differe. *Molecular and Cellular Biology*, 16:5276–5287, 1996.

[90] CJ Vaske, C House, T Luu, B Frank, CH Yeang, NH Lee, and JM Stuart. A factor graph nested effects model to identify networks from genetic perturbations. *PLoS Comput Biol*, 5:el000274, 2009.

[91] Bo Yu, Maureen E. Lane, Richard G. Pestell, Chris Albanese, and Scott Wadler. Downregulation of cyclin d1 alters cdk 4- and cdk 2-specific phosphorylation of retinoblastoma protein. *Molecular Cell Biology Research Communications*, 3(6):352 – 359, 2000.

[92] Le Yu, Steven Watterson, Stephen Marshall, and Peter Ghazal. Inferring boolean networks with perturbation from sparse gene expression data: a general model applied to the interferon regulatory network. *Mol. BioSyst.*, 4:1024–1030, 2008.

[93] C Zeller, H Frohlich, and A Tresch. A bayesian network view on nested effects models. *EURASIP Journal on Bioinformatics and Systems Biology*, 195272:8, 2009.

[94] Lixing Zhan, Bin Xiang, and Senthil K. Muthuswamy. Controlled activation of erbb1/erbb2 heterodimers promote invasion of three-dimensional organized epithelia in an erbb1-dependent manner: Implications for progression of erbb2-overexpressing tumors. *Cancer Research*, 66(10):5201–5208, 2006.