Integrating phosphoproteomic time series data into prior knowledge networks

Misbah Razzaq

UNIVERSITE BRETAGNE LOIRE / MATHSTIC

CENTRALE NANTES

THESE DE DOCTORAT DE

L'ÉCOLE CENTRALE DE NANTES
COMUE UNIVERSITE BRETAGNE LOIRE

ECOLE DOCTORALE N° 601
Mathématiques et Sciences et Technologies de l'Information et de la Communication
Spécialité : *Informatique*

Par

« Misbah RAZZAQ »

« Integrating Phosphoproteomic Time Series Data into Prior Knowledge Networks »

Thèse présentée et soutenue à « Nantes », le « 05/12/2018 »
Unité de recherche : Laboratoire des Sciences du Numérique de Nantes

Rapporteurs avant soutenance :

Céline ROUVEIROL	Professeur des universités, Université Paris 13
Thomas SAUTER	Professeur, Université du Luxembourg

Composition du Jury :

Président :	Frédéric SAUBION	Professeur des universités, Université d'Angers
Examinateurs :	Sara-Jane DUNN	Directrice de recherche, Microsoft Research Cambridge
	Sabine PERES	Maître de conférences, Université de Paris Sud
Dir. de thèse :	Jérémie BOURDON	Professeur des universités, Université de Nantes
Co-encadrant de thèse :	Carito GUZIOLOWSKI	Maître de conférences, École Centrale de Nantes

Acknowledgment

First and foremost, I want to thank *God* for blessing me with this life and for being there in good and bad times.

I would like to thank my supervisor, *Dr. Carito Guziolowksi*, who I met for the first time when I arrived to Nantes by train at 11 o clock at night. She was there to pick me up at this late hour. She won my heart right there by this simple act of kindness. I knew at that moment that I am going to learn science as well as humanity from her. She has played a significant role in my life in these 3 years of PhD. Her constant support and guidance has helped me to understand my thesis subject properly.

I would also like to thank my supervisor, *Dr. Jeremie Bourdon*, who I get to know in the mid of my PhD. He has provided me with immense support and guidance during my thesis work. I have also learned a great deal of politeness from him.

I am also grateful to my parents, for not following the typical traditions and helping me to get the education, which allowed me to write this thesis. My mother, *Rubeena*, encouraged me to be an independent woman. My father, *Razzaq*, appreciated my individuality instead of making me follow the crowd. My brothers, *Adnan* and *Munsfeen*, provided constant loving support even during the times when we disagreed.

I would also like to thank my special friend, *Roland*, with whom I enjoyed an 80km cycling trip and 1000m hike (record of my life, almost died). He also annoyed me a lot with his ambition of teaching me how to use articles. He taught me the importance of persistence and confidence.

I am thankful to my friends, *Julia*, *Abhilash*, and *Khaoula* for taking care of me when I got sick which happened quite a lot during first two years of my PhD. Thanks to *Julien* for taking care of my plants and listening to me during stressful times. Thanks to *Simon* for trying to teach me how to swim which I still cannot do. I am also grateful to *Bodgan*, *Konstatin*, *Sylvain*, *Deep*, *Tahir*, *Sebestien* and (annoying) *Saman* for great moments and discussions.

I would like to end this section with this beautiful quote:

"Your task is not to seek for love, but merely to seek and find all the barriers within yourself that you have built against it." Rumi.

Contents

List of Tables

List of Figures

1

Introduction

Systems biology is the study of biological functions and mechanisms consisting of DNA, RNA, and proteins in a systematic way. It is also concerned with understanding the complex and dynamic biological organization within the living organism in order to explain experimentally observed behaviors and do future predictions [Alo06]. Cells are the fundamental units of a living organism and share the same building blocks consisting of DNA, RNA and proteins with different complexity and structure. The central dogma of molecular biology explains the continuous reproduction of cells and flow of genetic information. It states that the blueprint of the DNA is carried by the mRNA (messenger RNA) to make a "protein" as a functional product, which performs the cellular work. Figure 1.1 explains the processes involved in the central dogma of biology: (1) Replication, (2) Transcription, and (3) Translation. Replication is the process of generating new DNA as functional product. Transcription produces new RNA as a result. Translation results in a protein as final product. Even after translation, proteins have to go through many modifications in order to become fully functional [Alo06].

These modifications involve the folding into three dimensional structure and moving to a specific location in the cell. Each cell contains different types of proteins and each protein can be expressed differently under different circumstances. Proteins play an integral role in all cellular functions of living organisms, i.e., creating a signaling cascade and regulating various biological mechanisms. Proteins are made up of the primary sequence of 20 amino acids. Usually proteins interact with other proteins to create complexes of varying sizes. These complexes work with other proteins or complexes to create pathways or modules to carry out tasks in a cell such as signal transduction, metabolism, duplication, DNA transcription, and DNA damage repair, etc. Understanding the interactions among proteins and complexes is crucial to fully

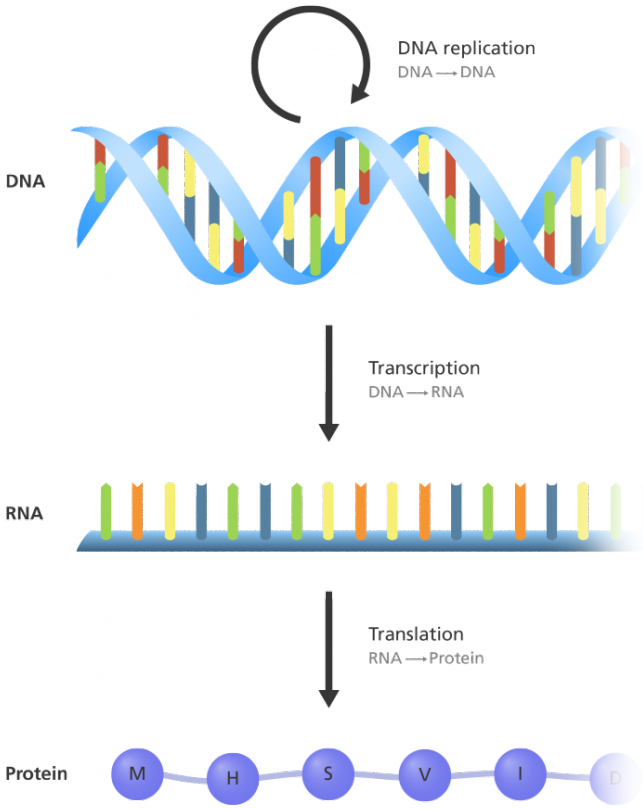


Figure 1.1 – Central dogma of biology (Image source: Genome Research Limited).

identify and characterize the structure and functions of the cell machinery.

There have been a lot of advances in experimental, analytical and computational techniques in the last couple of decades. These advances helped to generate large amounts of proteomics data, such as the yeast two-hybrid system and mass spectrometry. Large scale protein to protein interaction databases contain a large number of experimentally verified protein to protein interactions. These advances have made things less clear and at the same time they helped to understand the protein functions, interactions, and structure at varying levels. For example, linear dynamics were used to model signal transduction inside cells; now it seems that signals follow the more complex nonlinear dynamics. Understanding the mechanisms and interactions among proteins helps to unravel how information propagates within cells in different diseases such as cancer [Alo06, DW15]. In this thesis, I am focused on inferring and understanding the protein signaling networks in four breast cancer cell lines (BT20, BT549, MCF7, UACC812).

1.1 Motivation

Protein signaling networks are not static in nature since they respond to stimuli and perturbation. They constitute complex regulatory systems controlled by crosstalk and feedback mechanisms. These networks are often regulated in diseases. Discovering the precise mecha-

nisms of signal transduction may provide a better fundamental understanding of disease behavior. For instance, a main difficulty in cancer treatment is that different signaling networks reveal that cell populations specialize upon treatment and therefore patient responses may be heterogeneous. Computational models of signaling control for different patient groups could guide cancer research towards a better drug targeting system. In this work, we propose a methodological framework to discriminate among the regulatory mechanisms of four breast cancer cell lines by building predictive computational models.

Boolean network (BN) modeling is a simple yet powerful framework to study biological models such as signaling pathways or regulatory networks. BNs are based on qualitative approaches, allowing to model large scale biological networks [OPS+15]. Part of the research about this paradigm focuses on the topology of the network, searching for interesting characteristics such as cycles and hubs. Some researchers focus on identifying the influences among the components of the network, knowing which gene or protein activates or inhibits the others. There is also research about studying the dynamics of the model, how the combinations of inherent influences make the system evolve and how this evolution changes under different conditions or situations.

While many BN approaches exist to model biological systems, they focus mainly on system properties, and few exist to integrate experimental data in them. In this thesis, we use the caspo time series (*caspo-ts*) method to learn cell line specific BNs by integrating protein signaling networks with experimental data. The *caspo-ts* method uses Answer Set Programming (ASP) and Model Checking techniques to solve the combinatorial optimization problem of enumerating a family of Boolean networks (BNs) optimally explaining time-series data [OPS+16]. Figure 1.2 shows the overall process of *caspo-ts*, a publicly available software at [RPO18].

This thesis is focused on applying and ameliorating the *caspo-ts* method to large scale experimental data of four breast cancer cell lines to identify cell line specific Boolean models. More precisely, I apply the *caspo-ts* method to multi-perturbation time series data of four breast cancer cell lines (BT20, BT549, MCF7, UACC812) along with the traditional breast signaling network. I study how these cell lines derive different signaling behaviors under the same conditions. This behavior is important to study to unravel the remarkable heterogeneity among breast cancer types. It may help to design effective therapeutic strategies by having a better understanding of the underlying behavior of the system.

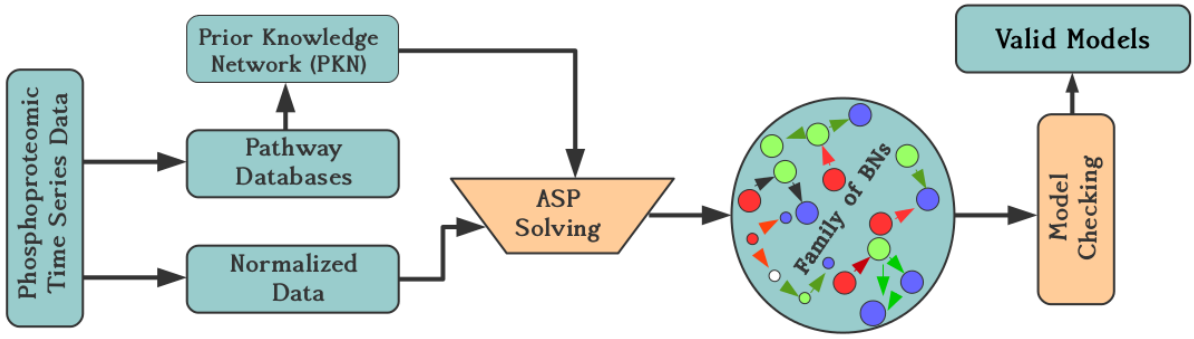


Figure 1.2 – Caspo-ts workflow. Prior Knowledge Networks (PKNs) are extracted from literature curated databases containing information about interactions between different proteins or genes. PKNs are available in different databases such as Reactome, PID, etc. Phosphoproteomic time-series data show the measurement of different proteins at different time points under multiple perturbations. A BN consists of a set of nodes where a Boolean function is assigned to each node. The state of each node is updated by evaluating the Boolean function. Given phosphoproteomic time series data we construct a PKN by querying pathway databases. After normalizing the time series data, we use it together with the PKN as input of *caspo-ts* (ASP component) for learning BNs. Finally, *caspo-ts*, uses a model checking step to filter false positive BNs. In this figure, the two main components of caspo-ts are shown in orange.

1.2 Contributions

1.2.1 The *caspo-ts* system applied to breast cancer

Protein signaling networks are static views of dynamic processes where proteins go through many biochemical modifications such as ubiquitination and phosphorylation to propagate signals that regulate cells and can act as feedback systems. Understanding the precise mechanisms underlying protein interactions can elucidate how signaling and cell cycle progression occur within cells in different diseases, such as cancer. This knowledge may guide better drug designs.

In this work, I focused on computational identification of BNs representing protein signaling behavior using the *caspo-ts* [OPS+15] method. I used four breast cancer cell lines of the HPN-DREAM challenge dataset [HHC+16, HNJC+17]. This dataset contains multi-perturbation time series data of four breast cancer cell lines (BT20, BT549, MCF7, UACC812). I thoroughly constructed and refined the prior knowledge network (PKN) from public databases, such as Reactom [WDD+14a], to cover the maximum number of proteins existing in the HPN-DREAM challenge dataset. I modeled a family of cell line specific BNs for the four breast cancer cell lines given this PKN.

My key findings suggest that this method is capable of constructing cell line specific Boolean models, which is extremely valuable given the heterogeneity of breast cancer due to many genetic modifications. An algorithm is implemented to analyze these cell line specific BNs to

study similarity among these breast cancer cell lines. I have highlighted the common and distinct behaviors.

Further, I validated the inferred BNs using the testing dataset provided by the HPN-DREAM challenge. This dataset was not included while inferring BNs. My models have a Root Mean Square Error (RMSE) of 0.31 with respect to the traces for the testing data, providing an optimal fit to the testing data. Furthermore, I also validated the cell line specific Boolean models by comparing them to the canonical mTOR pathway. The obtained results are comparable to the top performing teams of the HPN-DREAM challenge. In addition, this approach can also be used as a complementary method to identify erroneous experiments. Related to this work, I have published an article in Plos Computational Biology journal [RPS+18].

1.2.2 Extension of the *caspo-ts* system

Diverse Boolean models

Since the ASP solver uses a backtracking algorithm to exhaustively generate BNs, it can lead to a situation where successive BNs share very similar properties. This can be problematic specially in the case of a large solution space where discovering or analyzing all BNs becomes computationally hard. To resolve this issue, a diverse enumeration scheme has been introduced. This feature has been implemented in *caspo-ts* and allows it to break up the clusters of similar BNs, hence generating diverse BNs. I refer to the modified *caspo-ts* as *caspo-ts*^{D}.

I have demonstrated the results of the proposed approach on two different benchmark scenarios in systems biology: (1) an artificial dataset to model *TCR signaling* and (2) the *HPN-DREAM* challenge dataset to model breast cancer cell lines.

Results suggests substantial improvements of *caspo-ts*^{D} in solution quality by discovering more signaling behaviors than *caspo-ts*. Moreover, *caspo-ts*^{D} is able to find BNs in cases where *caspo-ts* is unable to find any. Related to this work, I have published an article in the Computational Methods in Systems Biology conference [RKR+18].

Parallel model checking

The ASP part of the *caspo-ts* system over-approximates BNs. This over-approximation removes a large set of BNs that have no reachable traces, reducing the number of invalid BNs. However, over-approximated reachability does not guarantee to reproduce all time series traces. Hence, at the final step the *caspo-ts* system uses a model checker to check exact reachability of all (binarized) traces existing in the experimental data by the given BN. This is the most time consuming part of the *caspo-ts* system. The verification of this reachability is a PSPACE-hard problem and the computation time for checking reachability is highly variable depending on the BN under verification. It can take from an hour to months. Moreover, *caspo-ts* specifies

all properties in one big specification, which can be slow to verify, especially in the case of large scale networks. I have improved this step, by splitting up the specification, to reduce the computational time of true positive BN detection. Results suggest substantial improvements in computation time.

1.3 Organization of the thesis

This thesis consists of six chapters.

Chapter 1 "Introduction" provides a broad overview and motivation behind this thesis. It precisely specifies the aims and contributions of this work.

Chapter 2 "Background and Related Work" provides background introduction on breast cancer, phosphoproteomic data, signaling networks, and computational modeling. We discuss four different computational methods to model signaling networks: ordinary differential equations, genetic algorithms, dynamic bayesian networks, and integer linear programming. We state the advantages and disadvantages of these methods as compared to the modeling approach (answer set programming) used in this work.

Chapter 3 "Answer Set Programming" introduces the modeling approach used in this thesis. We define basic notations and explain them with examples. Then we discuss related work in the context of modeling signaling networks using answer set programming. We also discuss related work in the context of learning diverse solutions using answer set programming.

Chapter 4 "Computational Discovery of Dynamic Cell Line Specific Boolean Networks from Multiplex Time-Course Data" discusses the first contribution of this thesis. We describe in detail the *caspo-ts* system and its application on a real case study (HPN-DREAM Challenge). We build signaling networks given phosphoproteomic data and prior knowledge networks. We used multiple criteria to evaluate the learned networks. We also highlight different characteristics (such as enumeration order, false positive rate) of the *caspo-ts* method, which are resolved in the following chapter.

Chapter 5 "Computing Diverse Boolean Networks from Phosphoproteomic Time Series Data" introduces the improved version of the *caspo-ts* methods. This chapter is related to the second contribution of the method. Here, we describe the new enumeration criteria to sample the solution space of the *caspo-ts* method. We also give a new algorithm to improve the computational time of verifying a solution using model checker.

Chapter 6 "Conclusions and Future Work" provides the summary and the future perspectives of this work.

2

Background and Related Work

2.1 Breast cancer

Breast Cancer is a remarkably complex and heterogeneous disease. It develops as a result of uncontrolled growth of abnormal cells due to genetic mutations. Mutation causes cells to multiply and divide chaotically. Mostly, this leads to the situation where multiple copies of abnormal cells give birth to a tumor. There exists a high diversity within the same type of breast tumors in terms of genomic alterations [RvJH+13]. Moreover, there is also substantial difference within tumor bearing patients. This high level of diversity poses a great challenge for cancer therapy, and demands diverse clinical features [Pol11].

2.1.1 Breast cancer types

Breast cancer can develop in different sites of the breast: the lobules, the ducts or the tissue in between them. The milk producing section of the breast is called lobules. Milk travels through the pathway from lobules to the nipple using ducts (see Figure 2.1). Depending on the area, breast cancer can be divided into two subtypes: (1) Non-invasive and (2) Invasive. With the non-invasive type, the cancer is completely confined to the ducts or lobules and does not spread to the surrounding connective tissues. The non-invasive breast cancer is further subdivided into two types: (1) ductal carcinoma in situ and (2) lobular carcinoma in situ. With the invasive type, the cancer breaks through the lobular and duct wall, and spreads into the surrounding connective tissues. There are further categories of the invasive breast cancer; the most common ones are: (1) invasive ductal carcinoma and (2) invasive lobular carcinoma [SDS+10].

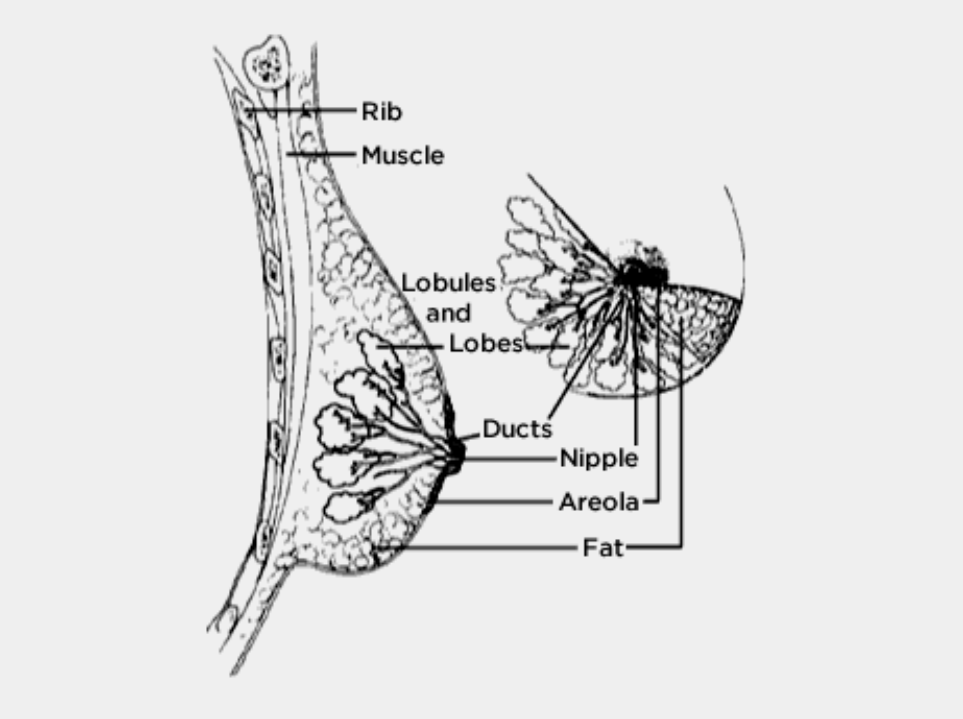


Figure 2.1 – Breast structure (Image source: National Cancer Institute www.cancer.gov).

In recent years, researchers have been studying different genetic mutations in breast cancer through gene analysis techniques (expression microarrays). This led to the development of another classification criteria for breast cancer, i.e., molecular and genetic classification [SLB+10]. According to this criteria, breast cancer can be divided into four molecular subtypes [HS11, DCBL17]:

1. Luminal A or B,
2. Triple negative (A or B) or Basal like,
3. HER2 type,
4. Claudin low.

Luminal breast cancer is hormone receptor positive (estrogen and progesterone). This cancer is slow to grow because of tight cell-cell junctions. Luminal breast cancer is further subdivided into luminal A or B, based on the existence of human epithelial receptor 2 (HER2). HER2 is absent in luminal A and present in luminal B breast cancer. Triple negative breast cancer does not contain any of the most common three receptors (HER2, estrogen and progesterone). Hence, hormone therapy cannot be applied to treat this type of breast cancer. Triple negative breast cancer is further divided into subtype A and B. Subtype A has an abundance of basal markers while subtype B is enriched with stem cell markers. HER2 positive breast cancer contains an abundance of HER2 proteins. This type of cancer is more aggressive than other types. As triple negative breast cancer, claudin low breast cancer also displays the absence of three hormone receptors. Claudin genes (3, 4 and 7), and E-cadherin proteins are absent in

claudin low breast cancer. Studies show that claudin low cancer is enriched with stem cells or cancer initiating cells, which can help in studying earlier stages of tumor cells [Per11, SFG+14, DCBL17].

2.1.2 Cell lines

Cell culture is the process in which cells are cultivated in an artificial environment. Cells can be isolated through a variety of means. They can be derived from the living tissue and disaggregated by mechanical or enzymatic means. They can also be taken from an already established cell line. Cells are maintained and proliferated under controlled conditions, known as the primary culture stage. At this stage, cells are usually subcultured to provide more room for cultivation. After subculturing, cells capable of proliferation are selected and as with the outgrowth from the primary culture stage, give rise to a cell line. Cell lines developed through a primary culture usually have a genetically determined life span. However, some cell lines are transformed into immortal cells and can grow indefinitely under optimal conditions. This can happen because of variety of reasons, e.g., chemical induction or spontaneous occurrence [Fre06].

Cell lines provide a simple and powerful model for studying and analyzing breast cancer biology. They serve as an unlimited source of homogeneous self-replicating material, and are easy to handle, and replace in case of contamination. However, they pose a number of challenges too. Cell lines are susceptible to genetic and epigenetic drifts during their culture [BSE05]. If a cell line is stored for a long time, a subpopulation can arise and cause phenotypic drift resulting in various clones within it [DCBL17].

Despite of their challenging nature, cell lines have been widely used as an experimental model for in vitro studies [NCF+06, BFA+10, PZS+10, FAC13]. They became a powerful tool to investigate how apoptosis, migration and proliferation are deregulated during breast cancer progression. Breast cancer cell lines help to generate quantifiable and reproducible results [VGR07]. They have provided valuable insights into understanding different aspects of breast cancer [LL04]. They have created a large amount of knowledge about breast cancer biology [Eth96, SLB+10, YXW+10, TCJ+10]. They have suggested novel cancer therapies over the past decades [WBB+99, WBM+98, Mas00]. They have been used in preclinical studies and predicted accurate clinical outcomes [CGL10]. In this thesis, we are using four breast cancer cell lines (BT20, BT549, MCF7, and UACC812) to study signaling behaviors.

Molecular classification of breast Cancer cell lines used in this thesis

As described above, breast cancer cell lines are categorized according to the three important receptors: ER (estrogen receptor), PR (progesterone receptor), and HER2 (human epithelial

receptor 2) [DCBL17]. In Table 2.1, we show the molecular classification of the four cell lines used in this study.

Table 2.1 – Molecular classification of breast cancer cell lines.

Cell Lines	ER	PR	HER2	Subtype
BT20	−	−	−	Triple Negative A
BT549	−	−	−	Triple Negative B
MCF7	+	+	−	Luminal A
UACC812	+	+ / −	+	Luminal B

In the following, we discuss some history and research work related to these breast cancer cell lines.

BT20

In 1958, the BT20 cancer cell line was isolated from a 74 year old woman. It is the first human breast cancer cell line and was established by Lasfargues and Ozzello [LO58]. It depicts the invasive breast cancer type. Even though it was the first breast cancer cell line, it has not been widely used to study breast cancer. Here, we cite some of the work which has been done using this cell line alongside other cell lines.

In the past, there was a lack of experimental models to study biological properties of human breast cancer. For this, they transplanted the BT20 cell line into nude athymic mice. Results showed the development of tumors at the injection site. Their observations showed that nude athymic mice are suitable for studying human breast cancer [OSM+74].

This cell line together with other breast and ovarian cancer cell lines have been used to study the HER2/neu assay sensitivity. According to the results, the BT20 cell line did not contain an over-expression of HER2 [RJC+02].

A group of 41 breast cancer cell lines (including BT20) was studied to identify BRCA1 mutants. BRCA1 germ mutations put female mutation carriers to high risk of ovarian and breast cancer. Four new cell lines were identified with BRCA1 mutations [EHN+06].

This cell line belongs to the triple negative breast cancer, which means hormone therapy cannot be applied to this kind of cancer. Triple negative breast cancer patients have a poor treatment outcome because of the lack of validated molecular targets. There is a clear need to enhance knowledge about this type of cancer to develop better therapies. The BT20 cell line has been used to study the triple negative breast cancer type [CGL10]. Results suggests that the BT20 cell line has very high expression of the EGFR protein and a genetic amplification of the EGFR gene [LG87].

MCF7

In 1970, the MCF7 cancer cell line was developed using pleural effusion from a 69 year old female metastatic breast cancer patient. It was established by Soule and colleagues at the Michigan Cancer Foundation [SVL+73]. MCF7 is the most studied breast cancer cell line. It mimics several invasive breast cancers which expresses the estrogen receptor (ER). It is widely used to study estrogen response and resistance both in vivo and in vitro, as this cell line maintains a substantial level of ER, considering ER maintenance is not a trivial task. There have been around 25,000 publications related to this cell line till now.

MCF7 played a major role in the development of antibodies for ER positive tumors, since Green *et al.* developed the first monoclonal antibody to ER [GNEJ80]. These antibodies led to the identification of cDNA clones which express ER mRNA. This aided in the cloning and sequencing of the ESR1 gene [WGG+85]. These antibodies also helped in measuring ER levels in human breast cancer, thus serving as a guide on the use of hormone therapy for ER positive tumors. This cell line also expresses progesterone, glucocorticoid, and androgen receptors [HCM75]. Hence, it is used as a valuable model to study other hormone response pathways.

An area of research focuses on estrogen based stimulation of MCF7 cells. Earlier studies were focused on how estrogen regulates the growth factor signaling [OS11]. Recent studies have revealed that estrogen represses and induces a large number of genes simultaneously. This generates a complex network of alterations, which coordinate to change growth [CPM+12].

MCF7 cells have also been used to study hormone resistance. Some researchers have developed hormone resistant variants of MCF7 cells by either chronic exposure to anti estrogens or estrogen withdrawal. Initially, this slowed down the cell growth, but finally growth resumed. It has been shown that estrogen deprived cells are highly sensitive to estrogen stimulation and express substantial levels of ER [JSB+98]. These studies have identified transcriptomic and epigenetic alterations, which eventually lead to changes in growth factor signaling. The single cell cloning of cells without classification of ER positive or negative clones, highlighted the heterogeneity existing in breast cancer [OZG+01].

MCF7 cell lines do not have an amplified level of HER2. However, Osborne *et al.* have developed MCF7 cells with amplified HER2 [BSS+92]. They showed how anti HER2 inhibitors can block growth and can be used for treatment. The studies have been validated clinically. Moreover, many laboratories have developed anti HER2 resistant cell lines. Despite the limitation imposed by cell culture, this cell line has tremendously advanced the knowledge about breast cancer and gave new directions to breast cancer research [LOD15].

BT549

In 1978, the BT549 cell line was derived from a 72 year old women. This cell line was established by Coutinho and Lasfargues. It is a breast ductal carcinoma cell line. This cell line represents the triple negative breast cancer type [LC81].

Lehmann *et al.* analyzed gene expression profiles from 21 breast cancer datasets. They identified 587 triple negative breast cancer cases from these breast cancer datasets. Cluster analysis was performed to identify 6 subtypes of triple negative breast cancer. They derived gene expression signatures from these subtypes to select the representative cell line of these subtypes. Then prominent signaling pathways were pharmacologically targeted in these cell lines to study the response to targeted therapies. Results showed that different cell lines have different sensitivities. This suggests that the heterogeneity of this disease can be studied through these triple negative cell lines and can help with designing effective preclinical treatments [LBC+11].

In [GMN+12], a group of 25 triple negative breast cancer cell lines were analyzed to identify similarities between cell line and triple negative breast cancer. These cell lines were studied on three molecular levels: genomic, transcriptomic, and epigenomic. They categorized cell lines in three groups. Two groups consisted of ER negative cell lines, while one encompassed of three ER negative and all ER positive cell lines The first two groups agreed with the existing knowledge, while one group was not consistent with existing studies. They further extended the characterization of these breast cancer cell lines. This provided valuable knowledge about suitability of a particular cell line for modeling different features of the breast cancer disease [GMN+12].

Recently, a new classification criteria was proposed to categorize breast cancer cell lines [SSK+17]. The relationship between RNA, TP53 mutation status and protein expression was also quantified. Their analysis revealed heterogeneity within cultures of established cell lines. They compared their finding with other studies, to help guide the selection of cell line models for in vivo and in vitro studies [SSK+17].

UACC812

In 1988, the UACC812 cell line was isolated from a 42 year old women. This cell line was established by Meltzer and colleagues [MLD+91]. It represents the HER2 positive breast cancer type.

Wang *et al.* investigated mechanisms of resistance to HER2 targeted drugs using a panel of HER2 positive cell lines. They used two drugs (trastuzumab and lapatinib) alone and in combination to study drug resistance. Their results revealed that the resistance to trastuzumab is due to the reactivation of the HER2 pathway. Furthermore, resistance to lapatinib or lapatinib combined with trastuzumab is associated with an alternative signaling through the ER pathway.

They suggested to completely block the HER network and inhibit ER to develop an optimal therapy [WMG+11].

Giuliano *et al.* analyzed the effects of HER2 targeted therapies on ER and Bcl2 expression in clinical tumor samples and preclinical models. Results suggested that Bcl2 and ER expression increased significantly in breast tumor xenografts treated with anti HER2 therapies. They revealed that the co-regulation of Bcl2 or ER with an anti HER2 therapy can prevent the increased Bcl2 and ER expression in HER2 positive breast cancer patients. They also reported that tumor progression slowed down with endocrine therapy in the presence of restored ER expression in xenograft tumors treated with anti-HER2 therapy [GHW+15].

Zhang *et al.* investigated the role of EPOR (erythropoietin receptor) in the inhibition and the resistance to the trastuzumab drug, in HER2 positive breast cancer. They identified EPOR mRNA and protein expression in HER2 positive breast cancer cell lines (UACC812, MDA-MB-453, and SKBR3). They suggested that EPOR expression may influence tumor proliferation and progression in HER2 positive breast cancer [ZDX+12].

Brennan *et al.* studied the connection between the JAM-A (junctional adhesion molecule A) and aggressive tumor, using breast cancer cell lines and clinical datasets. They suggested that over-expression of JAM-A may increase breast cancer progression in HER2 positive breast cancer. They also suggested that JAM-A can serve as a potential drug target and biomarker of HER2 positive breast cancer [BMH+13].

2.2 Phosphoproteomics experimental dataset

Cancer is both a genomic and a proteomic disease at a functional level. A genetic defect may ultimately lead to alterations in a protein signaling network. A defective signaling pathway can cause cancer growth, metastasis, invasion and survival. The future of cancer therapy is personalized treatment. Genomic profiling has been extensively used to personalize chemotherapy treatment. However, it became apparent that genomic profiling represent only one level of detail of the overall process because most of the pharmaceutical targets are proteins. Proteomic profiling provides direct information about protein signaling pathways. Proteomic profiling can be used to examine the signaling network in a normal and a cancerous state. Phosphoproteomics is a branch of proteomics, where researchers focus on studying proteins containing post-translational modifications. Such proteins are called phosphoproteins or phosphorylated proteins. Phosphorylation plays a ubiquitous role in regulating different processes, i.e., protein functions, cellular growth, degradation of proteins and signaling [NLSBW08].

Protein arrays are used to measure protein expression in a high-throughput manner. Protein arrays are categorized into a forward phase protein array and a reverse phase protein array (RPPA). Forward phase protein arrays (FPPA), also referred to as Enzyme Linked Immunosor-

bent Assays, involve immobilization of multiple antibodies on a surface (see Figure 2.2). FPPA provides a way to analyze the level of multiple proteins in a single sample simultaneously. In reverse phase protein arrays (RPPA), also referred to as Lysate Arrays, different samples (cell, tissue lysates) are immobilized on a surface and are analyzed for the presence of a single protein (see Figure 2.2). Hence, RPPA is ideally suited for measuring the level of a single protein for multiple samples simultaneously [NBK12, MLE10]. For example, it can be used to study the regulation of proteins in healthy and cancerous cells [HRS+09]. It is mainly used to: (1) define new and efficient targeted drug therapies for individual patients, (2) identify and validate biomarkers, and (3) verify drug effects (on or off targets, downstream signaling) [SGTP17].

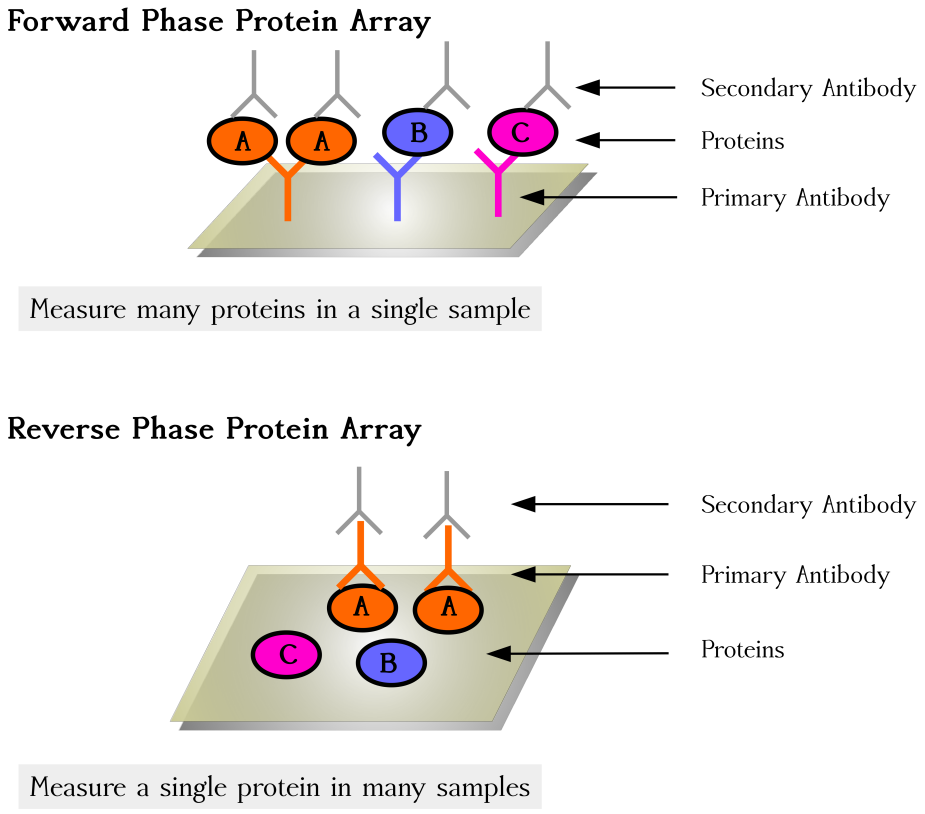


Figure 2.2 – Forward and reverse phase protein array format.

RPPA has emerged as a standard protein profiling platform over the past few years. RPPA provides an excellent way to capture the activated pathway or protein as a result of phosphorylation. MacBeath and Schreiber are the pioneer developers of protein microarrays [MS00]. They used high-throughput detection to identify protein to protein interaction. Another variation of protein microarrays named as "RPPA" was proposed by the Paweletz *et al.* in 2001 [PCB+01]. Since 2011, an annual meeting is held to provide a platform for scientists and researchers for exchanging ideas related to RPPA technology.

RPPA is a high-throughput technology that performs quantitative measurements of hun-

dreds of proteins in biological samples (clinical and preclinical). It consists of micro-blots of protein lysates from multiple samples of cell lines or tissues on a single array where each sample is described by at least one spot. Each array is incubated by one antibody to identify the corresponding protein expression across multiple samples simultaneously. Each array can contain thousands of samples. For high-throughput measurement of many proteins, multiplexing on multiple arrays of the same set of lysates is performed using different antibodies (see Figure 2.3) [CH15].

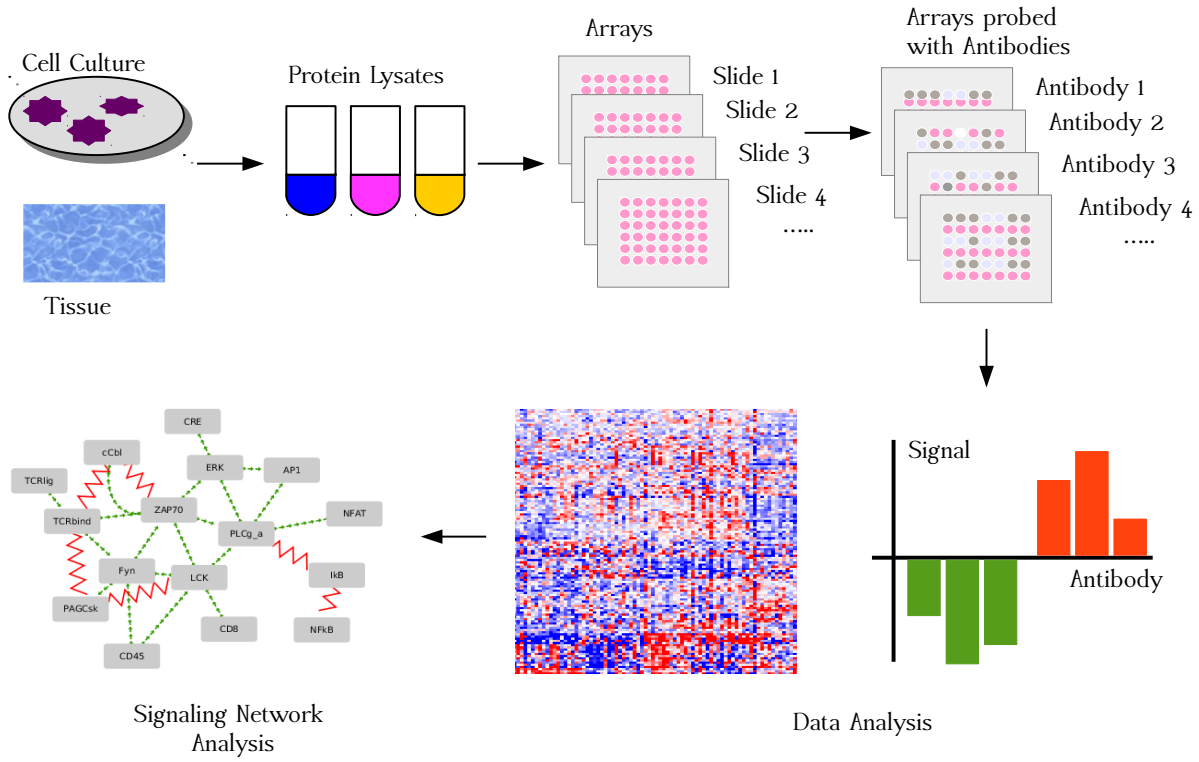


Figure 2.3 – Overview of analysis of signaling networks using RPPA technology.

RPPAs offer excellent data for studying deregulated signaling networks in cancer. It allows for investigating and comparing samples treated at different time points under different conditions with different doses. It is a useful tool to quantify multiple phosphorylated proteins even in a very small sample, which makes them suitable for individual patient therapy. Several studies have investigated heterogeneity among protein levels within a primary tumor, tumor and metastases of the same patient using data generated by the RPPA technology. Results showed significant heterogeneity in a subset of proteins within a tumor, and between metastases and the primary tumor. This suggests to analyze the samples from multiple locations instead of a single sample. Another study used RPPA to distinguish colon and ovarian cancer by finding molecular markers. Other studies found key proteins which play important role in the regulation of signaling pathways in different cancers such as breast cancer, leukemia, and glioma. A first commercial RPPA assay (the TheraLink HER Family Assay) was developed by Theranostics

Health in 2018 and has been given access to by some insurance companies. This assay can be used to guide personalized treatment by linking drug targets with available therapies [The18].

In contrast to RPPA, there exist other proteomic technologies such as mass spectrometry, western blot, and enzyme linked immunosorbent assay (ELISA) (See Table 2.2). Mass spectrometry based methods can analyze thousands of proteins at a time but they are unable to resolve all proteins within a sample. In contrast, RPPAs are highly sensitive and can detect even low abundance proteins. Mass spectrometry based approaches can be used to discover new candidate biomarkers by comparing healthy tissues with cancerous tissues. RPPA can be used to validate these markers in small samples (taken from patients), thereby deriving an individual precise cancer therapy. There are 380 validated antibodies available for RPPA. As compared to ELISA which requires two antibodies for a protein, RPPA requires only one antibody against the same protein. Western blot and ELISA require high amount of protein lysates as compared to the RPPA technology [BB15, CH15].

Table 2.2 – Characteristics of different protein analysis platforms.

	Pros	**Cons**
Western Blot	High specificity	High sample, Low-medium throughput
ELISA	Quantitative, High sensitive	High sample, Costly setup for high-throughput
Mass Spectrometry	High multiplex, Discovery of new biomarkers	Low throughput, Complex sample preparation
RPPA	High sensitive, High-throughput	Costly setup, Specific antibodies

In this thesis, we are using the HPN-DREAM dataset, which was generated by RPPA quantitative proteomics technology [HNJC$^+$17, HHC$^+$16]. These data contain phosphorylated measurements of multiple proteins under sets of perturbations. Perturbation refers to the combination of stimuli and inhibitors.

2.3 Prior knowledge networks

Prior Knowledge Networks (PKNs) are available in different databases such as Reactome, PID, and kegg among others [DW15, WDD$^+$14a, KG00, C$^+$04, KvIH$^+$12, Nis01, SBR$^+$06, SMC$^+$17, XRS$^+$00, PNK$^+$04, HMPL$^+$04, ZMPQ$^+$02, RMD08]. PKNs are graphs where molecules are represented by nodes and interactions are represented by edges. We can construct a PKN through different tools or softwares such as ReactomeFIViz [WDD$^+$14b] which is available as a Cytoscape [SMO$^+$03] plugin. A PKN alone cannot be used to build reliable dynamical models or to explain underlying biological behaviors [RCAdS15], because signaling behaviors are rewired in specific contexts. The signaling behavior may differ in cancerous and normal cells due to many genetic modifications [HLM$^+$12]. Therefore, it is extremely important to manifest how these networks are regulated in different diseases. In order to overcome

this issue, methods have been proposed which take into account both literature based knowledge (such as PKNs) and experimental data (such as phosphoproteomic datasets) to build signaling networks [OPS+15, MTH+12, GVE+13, SRAE+09, VGE+12]. In this thesis, we use a PKN (built using ReactomeFIViz) combined with the phosphoproteomic time series dataset of four breast cancer cell lines to generate cell line specific Boolean Networks.

2.4 Computational modeling of protein signaling networks

Network modeling has been widely used for studying phosphoproteomic data, yielding important insights into protein interactions, functions, and evolution. Figure 2.4 shows the work-flow of the computational modeling. Computational modeling bridges the gap between traditional biology and high-throughput datasets. A model is constructed using different formalisms like mathematical modeling, stochastic search methods, bayesian networks, integer linear programming, answer set programming, and then integrated with the experimental data. Afterwards, models can be validated through testing data. These testing data can be generated by traditional biology experiments. If a model's predictions coincide with the testing data, then it can be used to predict novel events. These novel events can be further validated through experiments before suggesting novel biological insights.

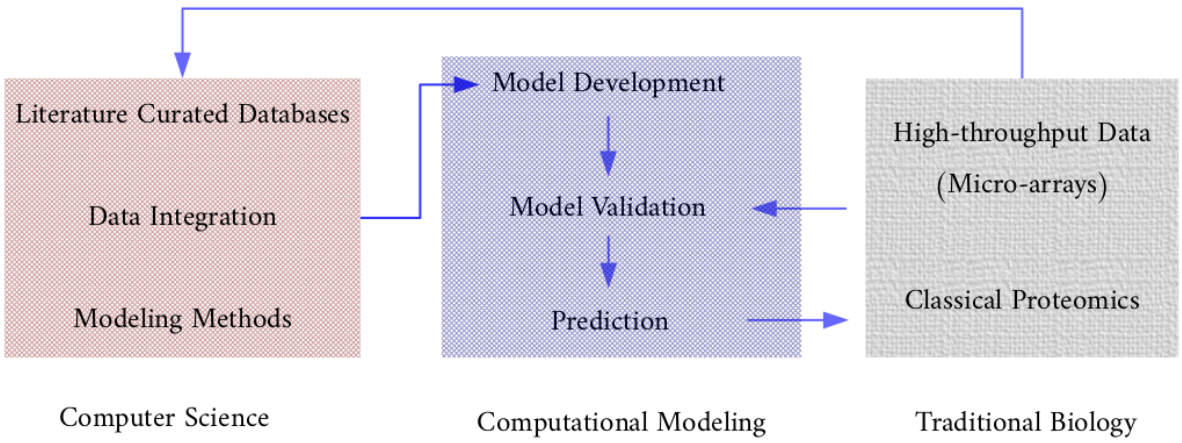


Figure 2.4 – Work-flow of the Computational Modeling (This work-flow was inspired by [KHJ+06]).

Network modeling is an important part of this thesis. In the following, we discuss the general frameworks that can be used for network discovery.

2.4.1 Ordinary differential equation models

Ordinary differential equations (ODEs) based approaches have been widely used for modeling complex and dynamic signaling networks. ODEs represent the interaction among various

molecules (such as proteins or genes). In ODEs, the various biological species (such as proteins) are represented as variables, where each variable has an equation reflecting its dynamic evolution over time. Here, we describe two types of ODEs:

1. Law of Mass Action,
2. Hill Function.

Here, we show how to formulate biochemical reactions (such as phosphorylation) using above mentioned equations.

Law of mass action

This law states that the rate of reaction is proportional to the chance of a collision of reactants, which in turn, is proportional to the concentrations of the participating molecules to the power of the molecularity (such as the number in which they enter the specific reaction) [JYL+17]. For example, we can define the molecular interaction between A, B and C by the following equation:

$$A + B \underset{k_-}{\overset{k_+}{\rightleftharpoons}} C, \tag{2.1}$$

where A and B are reactants, C is the product, and k_+ and k_- are the forward and the reverse kinetic rate constants . The following ODEs can be used to derive the change in the concentration of A, B, and C over time:

$$\frac{d[A]}{dt} = \frac{d[B]}{dt} = k_+[C] - k_-[A][B], \tag{2.2}$$

$$\frac{d[C]}{dt} = k_+[A][B] - k_-[C]. \tag{2.3}$$

Hill function

In the context of protein signaling networks, Hill functions can be used to represent the state of proteins such as activation or inhibition. In the following, Hill functions can be used to describe the change in protein expression over time. The Hill function is associated with each protein involved in the signaling network.

$$\frac{dy}{dt} = \sum_{i=1}^{M} f_+(x_i) + \sum_{j=1}^{N} f_-(x_j) - y * d_y, \tag{2.4}$$

$$f_{\overset{+}{-}}(x) = k_x y \frac{x^{\overset{+n_x}{-}}}{H_x^{\overset{+n_x}{-}} + x^{\overset{+n_x}{-}}}, \tag{2.5}$$

where y denotes the concentration of activated protein, x_i represents ith protein ($i = 1, 2, ..., M$), which activates protein y , and x_j is the jth protein ($j = 1, 2, ..., N$), which inhibits protein y. $f_{+}^{-}(x)$, represents the activating or inhibiting profile induced by protein x, respectively; $k_x y$ denotes the activation or inhibition rate; H_x represents the microscopic dissociation constant and n_x is the Hill coefficient; d_y denotes the degradation rate of protein y [JYL+17].

Iadevaia et al.

In [ILM+10], a computational approach is proposed to model the IGFR signaling network in the MDA-MB231 breast cancer cell line, using a set of ODEs with the law of mass action. The IGFR signaling network representing activation and inhibition interactions, was mapped into a set of 77 chemical reactions. For example, the following equation was used to represent the activation of protein PIK31 by IRS1,

$$PIK31 + IRS1^* \underset{k_-}{\overset{k_+}{\rightleftharpoons}} PIK31^*, \tag{2.6}$$

where k_+ and k_- are the forward and the reverse kinetic rate constants. These set of chemical reactions were mapped into a system of 127 ODEs. There were total of 313 unknown parameters, representing initial concentration of proteins and kinetic rate constants. The model reduction was performed to decrease computational complexity of the model. The reduced model consisted of 41 chemical reactions, 65 ODEs, and 161 unknown parameters. The quality of the reduced model was ensured by comparing protein profiles predicted by it, with the ones predicted by the original model. To obtain values for the 161 unknown parameters, the mass action model was trained against experimental data using the particle swarm optimization method. Training data consisted of time course measurements of six readouts proteins (p-AKT, p-TSC2, p-GSK3, p-p70S6K, p-mTOR, and p-MAPK). This dataset contained normalized protein profiles of MDA-MB231 cells after IGF1 stimulation. The RPPA technology was used to generate this data. They obtained 10 sets of parameters which explained the experimental data equally well. To determine the quality of the inferred model, testing data were used to generate responses of the IGFR network after MEK inhibition. Computational results were verified experimentally, confirming the accuracy of the modeled network. The mass action model was used to inhibit an individual molecule to predict the response of the IGFR network. Results suggest that inhibition of an individual molecule can activate another molecule, for example through a feedback loop. Three protein p-AKT, p-p70S6K, and p-MAPK levels are usually up-regulated in cancer. To decrease these protein levels, the authors identified optimal inhibition targets using random sampling of model parameters (initial protein concentration and kinetic constants). ODEs were solved with these randomly sampled parameter values. The au-

thors identified five main drug targets p-IGFR, p-MAPK, p-MEK, p-IRS-1, and p-AKT. They also performed experimental validations of their findings. Experimental findings agreed that the optimal combination (inhibition of the PI3K and MAPK pathways) of drugs decreased cell proliferation and inhibited cell signaling. However, non-optimal combination (MEK and mTOR inhibitors) did not sufficiently inhibit signaling.

Solomon et al.

Solomon *et al.* [IGT10] proposed a mathematical approach based on ODEs, to model the HER2/3 AKT protein signaling pathway in breast cancer. They used experimental data from two breast cancer cell lines (SKBr3 and BT747). This dataset contained short term (48 hours) and long term (2 days) effects of inhibition of TKIs (Gefitinib). In this study, 46 proteins (species) were modeled using ODEs. The connectivity information was derived from the literature. All kinetic reactions were represented as first or second order mass action equations. For example, following equation was used to model phosphorylation of B with rate K_b,

$$\frac{d(pB)}{dt} = K_b * B, \tag{2.7}$$

where p represents the phosphorylation of the protein.

The authors also took into account the trafficking of proteins between the cell membrane and cytoplasm, since HER3 can take hours or days to transfer to cell membrane. This slow transfer can be caused by the feedback loops involved in the pathway or DNA reprogramming (cells rewrite their DNA to survive). Two versions of the model were created to account for the transfer of protein between cell membrane and cytoplasm. DNA reprogramming was modeled by a long delay, and feedback was modeled as a reaction rate following the law of mass action. They implemented the process of protein transfer between cell membrane and cytoplasm by defining two compartments. One compartment contained HER2 and HER3, and other contained internal HER3. Transfer between these two compartments was modeled using diffusion differential equations. Parameter estimation was performed using simple search algorithm, allowing the models to fit experimental data. In this work, they described the detailed analysis of one model which has a feedback loop mechanism to show the slow transfer of HER3 between two compartments. The authors applied Gefitinib on this model to study its effects. For this, the model was augmented with a chemical reaction (which follows the kinetic law of mass action), representing that Gefitinib binds to HER2 and its dimers. Results showed pHER3 and AKT recovered their levels even after persistent inhibition of HER2, and HER3 transferred from cytoplasm to the cell membrane. After they inhibited the trafficking between compartments and applied Gefitinib. Results showed pHER3 and AKT did not recover their levels, and HER3

did not transfer to the cell membrane. Another simulation was performed where AKT was activated externally. Results showed decrease in pHER3 level and HER3 is transferred from the cell membrane to the cell. They kept the parameters where simulation results agreed with the experimental data. Finally, they proposed and simulated a treatment scheme where AKT was activated for two days, which caused drainage in HER3 from the cell membrane, and then Gefitinib was applied which inhibited the HER2/3 and reduced transferring. Their simulation results suggest that the proposed scheme works better than the traditional scheme where Gefitinib was applied alone. Moreover, low level of AKT was maintained, eventually leading to the apoptosis of cancer cells.

Shao et al.

Shao *et al.* [SPJ+13] proposed a systematic approach to study therapeutic effects on cancer cells, and side effects on liver cells. Several canonical pathways were selected by investigating literature curated databases such as KEGG and IPA [KG00, Ing]. These pathways were filtered according to the list of underlying proteins, to build a generic pathway of 26 proteins. Two pathways (cancer cell line specific and liver specific) were generated by training the generic pathway on two different experimental datasets (imaging data and cue signal response data). The cue signaling response dataset was a multiplex time series dataset, containing measurement of 12 proteins under 7 perturbations (6 stimuli and 5 inhibitors) at 4 time points. The cell imaging data consisted of measurements of 5 readouts for PC9 lung cancer cell line under kinase inhibitor (GW843682) with 12 concentration levels. A mathematical model based on ODEs with Hill functions was formulated for the generic pathway. For each protein in this pathway, Hill functions were used to represent inhibiting or activating effects, and were formulated according to Equation 2.4. The mathematical model consisted of 21 ODEs and 151 parameters. Then this mathematical model was trained against experimental data (imaging data) to build a cancer cell line specific pathway. To build the liver cell specific pathway, the generic pathway was trained against the cues response signaling dataset. This model was represented with 52 ODEs (following the law of mass action) and 83 parameters. For parameter estimation, an objective function was defined to minimize the distance between the simulation and experimental results. The genetic search algorithm was adopted as a two stage approach for optimal parameter estimation. During the first stage, the algorithm was repeated 50 times to select the best parameters with minimal error between simulation and experimental results. Then the model was simplified by removing links (containing non-best parameters). During the second stage, best parameters were refined by repeating the search algorithm for a simplified model. To perform validation, a leave-one out approach was used. Simulation predictions agreed with experimental results. They analyzed the effects of 27 kinase inhibitors and identified 6 key inhibitors. Out of these, the PF02341066 kinase inhibitor was identified as a proper inhibitor

for suppressing cancer while avoiding damage to the liver cells. They also analyzed the effect of combined inhibitions by selecting 4 out of 6 effective key kinase inhibitors. Their findings revealed the threshold for each kinase inhibitor in a combination to predict the expected combined effect.

Conclusion

Mathematical modeling offers useful and powerful tools to model small-scale networks. Models elucidated using mathematical modeling are complex and require explicit specifications of kinetic parameters of the system; parameter estimation becomes computationally intensive as networks grow larger [WMG08, SK13, MTH+12, MW07, ABLS06]. One way to obtain kinetic parameters is from the literature, then find remaining unknown parameters by fitting the model to the experimental data. However, parameter values differ by the order of magnitude. This can be due to incorrect experimental measures or inaccurate modeling. Therefore, parameter estimation is a challenging question in mathematical modeling. Several heuristic methods such as genetic algorithms, and particle swarm optimization are used to estimate parameters in ODE modeling [JYL+17, Blo09].

2.4.2 Stochastic models

Stochastic modeling frameworks are used to model signaling networks accurately, by offering a strategy to cope with the uncertainty and noise inherent in biological processes. Different stochastic methods can be used to model signaling networks such as genetic algorithms.

Genetic algorithms

A genetic algorithm is a bio-inspired search method, which follows the principal of natural selection and evolution. It starts with the belief that the offspring inherit properties of their parents. If parents have the best fitness then their offspring will be better than parents. This is an iterative process, and finally the fittest individuals will be found. Figure 2.5 shows general steps involved in solving problems using genetic algorithm. It starts with defining an initial population. Each individual in this population represents a solution to the problem. A fitness function is evaluated for each individual according to some objective function. Then a stopping criteria is evaluated to create a new population or to terminate the search process. Genetic algorithms have been widely used for solving optimization problems. They are also used for parameter estimation of mathematical models based on ODEs.

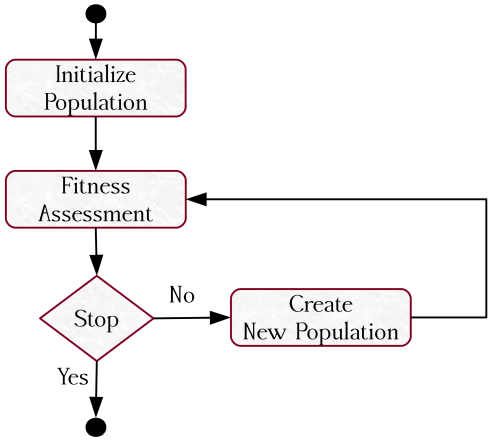


Figure 2.5 – Genetic algorithm.

Saez et al.

Saez *et al.* [SRAE+09] proposed a method (CellNOpt; Cell Net Optimizer), to train protein signaling networks against experimental data in order to build predictive BNs. A protein signaling network of downstream of seven cytokine and growth factor receptors in liver cells was built using ingenuity systems [Ing], and then augmented with literature based information. The network consisted of 82 nodes and 116 edges. The network was compressed by removing non identifiable elements using the CellNOpt software, resulting in a network structure with 31 nodes and 53 edges. The superstructure of logical models was identified from the compressed network, consisting of 131 hyper-edges (logic gates). Hyper-edge refers to the generalization of an edge with multiple inputs and outputs. The resulting superstructure of BNs is trained against the experimental data (cue signal response dataset) using a genetic algorithm. This dataset contained measurements of 16 proteins before, and 30 min after, stimulation. The genetic algorithm was run multiple times to optimize the objective function. The objective function was based on minimizing the distance between the data and the simulation while penalizing model size. The resulting family of BNs was validated using those data absent from the training dataset. The results showed a good fitness of BNs to the training dataset. The key findings suggest that the trained networks have fewer interactions as compared to the protein signaling network, but has higher false positive, false negative rate. New links or interactions were suggested which improved the fitness of BNs to the experimental data. The resulting models were more predictive than the protein signaling network due to the elimination of nonfunctional links.

Mishra et al.

A hybrid approach based on genetic algorithm and ODEs was proposed by Mishra et al. [MBC+15] to model cell fate decisions and cancer signaling pathways. The authors started with building an apoptosis network (N_1) representing the signaling activities inside a normal

cell. The network N_1 consisted of 22 proteins, and was constructed by exploring literature knowledge. Twenty two ODEs were formulated for each protein to simulate the N_1 network. These ODEs were solved using ode45 (Matlab library function), to infer time series for each protein. Some parameter values were inferred directly from the literature, while others were selected randomly. From the N_1 network, the cancer network (N_2) was inferred by fitting the simulated data to the cancer dataset using the genetic algorithm. This dataset contained six phenotypic responses of 35 proteins at five time points , for three triple negative breast cancer cell lines BT20, MCF7 and MDA-MB-453. The genetic algorithm was iterated 150 times to discover 50 cancer networks. The rewired events discovered by the genetic algorithm were different in different cancer networks. The rewired events with high frequency were kept to construct N_2 network. The N_2 network is a modified version of the N_1 network, containing two newly added edges while 3 edges were deleted. Various inferred rewired events were verified and confirmed through existing literature. From the N_2 network, the cancer network (N_3) is inferred by fitting the simulated data to the drug treatment dataset using the genetic algorithm. Th drug treatment dataset consisted of measurements of the signaling and cell fate data under six treatments. The genetic algorithm was repeated 150 times to discover the drug sensitive network. The N_3 network is a modified version of the N_2 network, containing two newly added edges while 2 edges were deleted. Most of the rewiring events inferred by this method are consistent with the literature. In this work, authors modeled only 18 out of 35 proteins existing in the experimental dataset.

Conclusion

Here, I have described two studies based on non-deterministic stochastic search method (genetic algorithms). Saez *et al.* [SRAE$^+$09] used genetic search algorithm to train Boolean networks against experimental data. However, it was not guaranteed that genetic search would yield the lowest value of the objective function, so all interactions were exhaustively checked to decrease the model size. Mishra *et al.* [MBC$^+$15] also used the genetic algorithm to infer models which fit experimental data. They discovered different rewiring events over different iterations of the algorithm. Hence a decent number of iterations were required to converge the solutions. Overall, stochastic search methods are useful for modeling signaling networks, especially with their inherent ability to deal with noise. However, quality of the solution in case of genetic algorithms is highly dependent on the initial population, well written objective function, and the number of iterations. Similar to our method (*caspo-ts*), the genetic algorithm based method generates a family of solutions. In comparison to *caspo-ts*, stochastic search methods cannot generate a complete set of solutions, hence they cannot guarantee a global optimal solution.

2.4.3 Dynamic bayesian networks

A bayesian network is a directed acyclic graph G, representing a probabilistic relationship between random variables X_i where $i = 1 \dots n$. These random variables represent nodes and the probabilistic dependencies among them is represented by edges. The edges are described by the joint probability distribution $P(X_1, \dots, X_n)$. The joint probability distribution of the graph has the following general form:

$$P(X_1, \dots, X_n) = \Pi_{i=1}^{N} P(X_i = x_i || X_j = x_j, \dots, X_{j+p} = x_{j+p})$$

where X_j is a parent of X_i in G [FLNP00].

Dynamic bayesian networks [DGH92, DGH$^+$95, MR02] are an extension of bayesian networks in which a directed graph represents how random variables X_i evolve over time. Dynamic bayesian networks can incorporate feedback loops. In the context of modeling time series data for protein signaling networks, proteins are represented by random variables X_i, time is represented by T_j where $j = 1 \dots n$. The dynamic bayesian networks described here follow the assumption that time can flow forward only, therefore the value of the variable X_i at $t + 1$ is only dependent on value of the variable X_i at t [FMR98, Hus03]. The dynamic bayesian networks can be referred to as static bayesian networks unrolled through time where each variable is represented at multiple time points [HLM$^+$12].

There are two steps to model bayesian networks: (1) structure learning and (2) parameter learning. The structure learning step involves finding a set of candidate directed acyclic graphs which best explain the data. Then these graphs are scored against some bayesian metric (Bayesian Dirichlet equivalence or Bayesian Information Criteria). The graph with highest scoring metric is selected. In the parameter learning step, the conditional probability distribution of each node is learned. The PKN is taken into account by assuming a particular distribution over a directed acyclic graph.

Hill et al. 2012

Hill *et al.* [HLM$^+$12] proposed a framework to learn cell specific protein signaling networks from phosphoproteomic data given a PKN. They started by building a structure of the network which best explains the data. The PKN is taken into account using an informative prior distribution on network structure. Then the posterior distribution over the network is calculated using the marginal likelihood and prior distribution. The marginal likelihood employed here penalizes models with complex parameters, hence reducing the model complexity. The prior distribution contained the expected number of edges (interactions existing in the PKN) and penalized the unusual edges (not part of expected edges). Finally, the posterior probabilities of edges are calculated by model averaging. Model averaging is used to highlight edges which

are common in many network structures. This network building approach was applied to learn signaling network for the MDA-MB-468 breast cancer cell line. The learning data contained measurements of 20 proteins at 8 time points under 4 growth conditions. The PKN consisted of 20 protein and 74 edges. The robustness of the cell line specific network to the PKN was investigated by adding and deleting an edge in the PKN. The 25 perturbed PKNs were generated with changes in one third (25) of the total edges (74). The results showed the robustness to modification in the PKN. The robustness of the results to the data perturbation was also investigated, by deleting part of the data and replacing it with the average of adjacent time points. The analysis showed that results are robust to the perturbation of the data. They generated a set of testable hypothesis, discovered various known and unknown signaling behaviors. The known inferred edges were validated through existing literature knowledge. Some of the novel edges were validated through experimental validation.

Hill et al. 2017

Hill *et al.* [HNJC$^+$17] learned cell line specific networks given PKN and phosphoproteomic data for four breast cancer cell lines (BT20, BT549, MCF7, UACC812) using a variant of dynamic bayesian networks. The learning data contained measurements of 35 proteins at 7 time points in all cell lines under a combination of perturbations (8 stimuli and 5 inhibitors). The PKN contains information about the connection of these 35 proteins with 65 edges curated from literature. A dynamic bayesian network approach was used to learn 32 context specific networks (4 cell line $*$ 8 stimuli) using learning dataset given PKN. Each network contained 49 edges on average, and 40$\%$ of these edges were not part of prior network. The learned network consist of probabilities associated to each edge in all these 32 networks. For validation, the changes in these context specific networks under inhibition were compared with the true descendants existing in the canonical pathway. Based on this, a true and false positive rate was calculated to draw the area under the operating curve (AUROC). It was observed that on average 8 proteins showed changes in one context which were not observed in other context. Experimental validation was performed for 78 $\%$ of these observations (104 out of 134 observations). To verify robustness of the approach, some part of the data (between 1 to 6 time points) for all proteins was deleted and replaced with adjacent time points. Initial and final time points were kept outside from random removal of the data. The robustness of the learning approach to the prior network was verified by adding and deleting an edge in the PKN. They generated a set of testable hypotheses by discovering some novel interactions. They discovered 235 novel interactions and some (six) of these interactions were validated using western blot analysis. Their key findings suggest that the signaling networks varies according to biological background. Heterogeneity of inferred networks strongly supports the fact that existing computational methods should be improved to take into account the context specificity.

Conclusion

The afore-mentioned approaches provide a useful way to model signaling networks in specific contexts. However, the dynamic bayesian network approach (like any other data driven approach) suffers from missing variables (intermediate nodes) due to limitation of measurements available in the experimental dataset. This hinders the mechanistic interpretation of the learned networks. Mostly, approaches based on dynamic bayesian network modeling assume the homogeneity of network structure and parameters through time. The softening of this assumption can be computationally expensive due to rapid increase in graph space and number of parameters [HLM$^+$12]. Moreover, predictions discovered through these networks are relatively less accurate. Also, these networks do not provide the activation and inhibition information [SHM$^+$15]. Generally, bayesian approaches are computationally expensive. They also cannot perform well with small datasets produced by protein microarrays[NSS$^+$08].

2.4.4 Integer linear programming

Integer linear programming (ILP) [NW88] is used to solve optimization problems. An ILP solver takes a description of such a problem consisting of a set of constraints and an objective function as input, calculating a solution that satisfies all constraints and minimizes the objective function. The ILP approach uses a set of variables, and a set of equalities and inequalities to represent the constraints. The variables are required to have integer values. A typical ILP program is expressed as follows:

$$minimize \qquad c^T x \tag{2.8}$$

$$subject\ to \qquad Ax \leq b \tag{2.9}$$

where c and b are vectors, A is a matrix, $x \geq 0$, and $x \in \mathbb{Z}^n$ where $\mathbb{Z}$ is the set of integers. Here, we give an example of the coloring problem for a graph $G = \{V, E\}$ with nodes $V = \{1, \ldots, n\}$ and edges $E \subseteq V \times V$. This problem states that adjacent nodes of graph G cannot have the same color. We assume that there are n colors in total. The binary variable y_k where $k \in \{1, \ldots, n\}$ is used to denote whether the color k is used for coloring. Another variable x_{ik}

is used to denote the color k for node i. This problem can be expressed in ILP as follows:

$$minimize \quad \sum_{k=1}^{n} y_k \tag{2.10}$$

$$subject\ to \quad \sum_{k=1}^{n} x_{ik} = 1 \qquad i \in V \tag{2.11}$$

$$x_{ik} - y_k \leq 0 \qquad i, k \in V \tag{2.12}$$

$$x_{ik} + x_{jk} \leq 1 \qquad (i, j) \in E, k \in V \tag{2.13}$$

$$0 \leq x_{ik}, y_k \leq 1 \qquad i, k \in V \tag{2.14}$$

$$x_{ik}, y_k \in \mathbb{Z} \qquad i, k \in V \tag{2.15}$$

The first three constraints ensure that each node is colored and the last two constraints are used to restrict x_{ik} and y_k to binary values [1].

Mitsos et al.

The authors in [MMS+09], presented a framework based on integer linear programming (ILP) to learn cell type specific pathways and to study the effects of different drugs on these pathways. Two types of datasets were constructed; one to learn the cell specific signaling network, and the other to learn signaling alterations when four drugs were induced. The first experimental dataset (D_1) consisted of 13 phosphoproteins measured under 55 perturbations (stimuli and inhibitors). The second experimental dataset (D_2) contained measurements of 13 phosphoproteins under 55 conditions (stimuli and drugs). The experimental data were normalized to convert values to a range of [0,1]. The prior network consisted of 74 proteins and 105 reactions. They built a cell type specific network (Boolean model) given the prior network and the experimental dataset (D_1). They described their problem in the form of equations and stated two objective functions. The first objective function is used to infer the network which best fits the experimental data. The second objective function is used to minimize the size of the network. Then the ILP solver is used to infer networks satisfying the constraints and optimizing the objective functions. The inferred cell line specific networks consisted of 49 proteins and 44 reactions. This network contained 5 stimuli and 13 phosphorylated proteins. They evaluated the effects of four drugs on this cell type specific network: Raf kinase inhibitor (Sorafenib), two potent EGFR kinase inhibitors (Erlotinib), the dual EGFR/ErbB-2 inhibitor (Lapatinib), and Gefitinib. They used similar learning process described above to learn 4 drug induced networks; using the cell specific network as the prior network and experimental dataset (D_2). Their results revealed the main known alterations in the pathway and also several unknown drug

1. The same example is encoded in ASP in the next chapter to draw the comparison between ILP and ASP formulations.

effects. Their key finding suggests that lapatinib, erlotinib, and gefitinib have similar kind of effects, very few alterations differentiating them, while sorafenib drug induce totally different signaling alterations.

Melas et al.

Melas *et al.* [MMM+11] extended the above mentioned ILP approach to build an extended pathway to link signaling behavior to a cellular response such as cytokine release or cell growth. They constructed two kinds of datasets for normal and cancer hepatocytes. One dataset (D_1) consisted of measurements of 16 phosphorylated proteins under approximately 50 perturbations (7 stimuli and 5 inhibitors). The other dataset (D_2) contained measurements of 33 cytokines under same set of perturbations. The learning process started with the construction of the canonical pathway via literature search, which covered the key phosphoproteins and stimuli. Then the ILP framework [MMS+09] was modified to add edges (to link proteins with cytokines) not existing in the canonical pathways. This extended network consisting of canonical and non-canonical edges was optimized to fit both experimental datasets (D_1, D_2). The optimization of the objective function was done in three main steps: 1) by removing the canonical edges which contradict signaling dataset (D_1), 2) removing non-canonical edges which contradict (D_2), and 3) removing the edges which have no effect on the network to decrease the overall network size. The ILP solver was used to optimize the objective functions to generate 100 different solutions for normal and cancer hepatocytes (Huh7). These solutions differed by 10% in the value of their objective function. The extended pathway for normal hepatocytes consisted of a total of 47 edges (19 canonical and 28 non-canonical edges). The extended pathway for cancer hepatocyte (Huh7) consisted of 43 edges (26 canonical and 17 non-canonical edges). These pathways are significantly different from each other, supporting the fact that Huh7 cells are not as responsive to stimuli (Toll Like Receptor) as normal hepatocytes are. The predictions performed through these models were also experimentally validated. The validation of the learning approach was performed by assessing the model's sensitivity to the canonical pathway, experimental design, and measured data. To verify sensitivity to the generic topology, 10 $\%$ of edges were replaced with random edges. The results showed that optimized pathways are sensitive to the canonical pathway. For verifying sensitivity to the experimental design, 5 to 50 $\%$ of experiments were left out. Results suggest that fitness error increases with the removal of number of experiments. The sensitivity to the measured data was checked by replacing part of the data-points with random numbers between 0 and 1. The fitness error also increased with the removal of number of data-points.

Conclusion

Mitsos *et al.* [MMS+09] provided an ILP based framework to study the effects of drugs on cell specific signaling networks. They successfully identified drug induced alterations in a network. Similar to our method (*caspo-ts*), the ILP based approach [MMS+09] is topology dependent and cannot determine the alterations outside the constructed networks. Melas *et al.* [MMM+11] proposed an extended ILP framework to study the cellular responses (growth, phenotypic, cytokines) under different signaling events. Their framework is based on the assumption that signaling and cellular dataset must be measured under same experimental conditions. The proposed approach is quite sensitive to changes in topology and datasets as compared to the *caspo-ts*. Similar to *caspo-ts*, ILP solvers enumerate a family of solutions and can find global optimal solutions. However, open source ILP solvers are not as efficient as commercial ILP solvers are [MMS+09]. All in all, ILP provides a useful paradigm to model signaling networks. However, ILP lacks a simple programing or modeling language which makes ILP coding a difficult task to write and maintain.

2.5 Conclusion

ODEs are suitable for detailed modeling of small systems where a complete description is available to model the system. Stochastic methods (more precisely genetic algorithms) cannot guarantee global optimal solutions as ILP and ASP offers (shown by [VGE+12]). However, the lack of an accessible programming language makes ILP hard to model the problem (depicted in graph coloring problem). Moreover, open source ILP solvers are not adequate to model problem of inferring BNs from time series data [MMS+09].

This thesis is focused on inferring BNs using an ASP based modeling approach, by combining traditional signaling networks with complex phosphoproteomic time-series data generated by RPPA in [HNJC+17, HHC+16]. This approach allows to build large scale networks without having to deal with large scale parameters as in case of ODEs, stochastic and bayesian approaches. Since ASP is an important part of this thesis. Therefore in the next chapter, we are introducing the modeling approach (ASP) used in this thesis, to formulate the problem of finding BNs from PKN and phosphoproteomic time series data.

3

Answer Set Programming

3.1 Introduction

Answer Set Programming (ASP) is an emerging simple, yet powerful, framework for reasoning and knowledge representation. The simple modeling language and flexible reasoning modes of ASP have led to the use of ASP in numerous domains such as systems biology [BCT+04, GSTV11, VGE+12, OPS+15], a decision support system for NASA shuttle controllers [NBG+01, BGN06], and product configuration [SN99]. ASP is a declarative problem solving approach, in which a problem is modeled rather than telling the computer how to solve it [GKKS12]. Figure 3.1 shows the work-flow of the ASP paradigm. Modeling aims at describing the problem in the form of a logic program using first order variables. Afterwards, solving generates stable models using searching algorithms to find a solution for the stated problem. These resulting solutions are called stable models or answer sets.

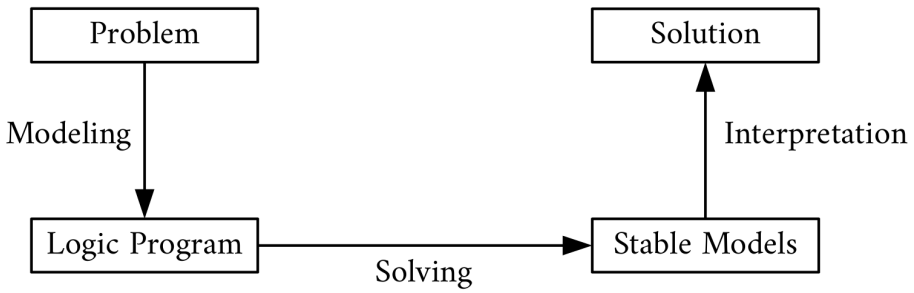


Figure 3.1 – ASP workflow

In this thesis, we use ASP to solve the combinatorial optimization problem of learning BNs by confronting phosphoproteomic data with PKNs. Available ASP solvers use high performance solving techniques, allowing us to enumerate all optimal solutions for a given problem. Indeed, there exist other solving techniques like satisfiability testing (SAT) [BHvM09], Integer Linear Programming (ILP) [Sch98] and Constraint Programming (CP) [DC03]. ASP solving techniques build upon SAT solving techniques. However, SAT-based systems lack ASP's rich modeling language, which forces the user to write complex programs for generating encodings, making it difficult to maintain and modify them. Another modeling difference is that the problem instances and the problem encodings are inseparable in SAT. Further, SAT encodings can be easily translated into the ASP programs in a modular way, but not vice versa [Nie99]. In general, the translation of an ASP program to a SAT encoding can lead to an exponential space explosion unless the language is extended [LR06]. Another difference between SAT and ASP solving is that the ASP semantics is non-monotonic while SAT is monotonic. This makes it easy to model concepts such as reachability and inertia in ASP. Constraint Programming and Integer Linear Programming also rely on monotonic foundations. Moreover, these formalisms focus on solving problems with constraints over real-valued variables or integers. However, there are extensions to ASP (for example clingcon), which allow for using similar constraints in ASP encodings. This allows for modeling a large class of problems with a natural and expressive modeling language using state-of-the-art solving techniques [SW18].

3.2 Basic ASP syntax

ASP borrows basic terminologies from classical logic. Variables are denoted by strings starting with capital letters, such as U, V, $Color$, etc. Symbols for predicates are represented using strings starting with lowercase letters, like for example, p, q, $node$, etc. Function symbols are denoted using strings starting with a lowercase letter, too, for example, f, g, h, etc.[1] Each predicate and function symbol is associated with an arity written as p/n and f/n where $n \geq 0$ gives the arity. A term is a variable, integer, or function $f(t_1, \ldots, t_n)$ such that f is a function symbol of arity n and each t_i is a term. An atom has form $p(t_1, \ldots, t_n)$ such that p is a predicate of arity n and each t_i is a term. Function symbols with arity zero are called constants. Furthermore, the parenthesis after a function or predicate are omitted if the arity is zero. Atoms or their negations are called literals. Terms and atoms without variables are called ground terms [GKKS12]. Table 3.1, describes the basic connectives used to form logic programs.

1. In general the alphabets of predicate and function symbols can overlap because this information can be derived from context within a logic program. Both alphabets are kept disjoint in the logic programs presented in this thesis.

Table 3.1 – Basic connectives to form logic programs

	Logic Program
if	$\leftarrow$
and	$\wedge$
default negation	$\neg$

A *logic program* consists of *rules* of form

$$h \leftarrow b_1 \wedge \cdots \wedge b_m \wedge \neg b_{m+1} \wedge \cdots \wedge \neg b_n$$

where h is an atom, $0 \leq m \leq n$, and each b_i is an atom. If a rule contains variables, we obtain a set of ground rules by substituting all possible ground terms for the variables. Given a logic program, the corresponding ground logic program is the set of all possible ground rules obtained from its rules. Such a (ground) logic program induces a set of stable models determined by the stable model semantics [GL88]. Each stable model is a subset of the atoms occurring in the logic program. Atoms appearing in this set are said to be true, and false otherwise. A rule is satisfied if its body (the part after the $\leftarrow$) is not satisfied, or its head atom h is true. A rule body is satisfied if all the atoms b_1 to b_m are true and all the atoms b_{m+1} to b_n are false. A rule with an empty body is called a *fact*. A *model* satisfies all (ground) rules of a logic program and a stable model also satisfies a minimality criterion. This criterion requires that each atom in a stable model is proved by some rule. For this, a true atom has to appear in at least one rule head with a satisfied body. Or more formally, a model X is a *stable model* if there is no smaller model $Y \subset X$ satisfying the reduct of the program; the reduct comprises all rules whose bodies are satisfied by X. In the following, we simply refer to the stable models of a logic program as its solutions. Next, we give a simple example of a logic program [2].

Example 3.2.1. Here, we model the light switch scenario.

$$switchOn \leftarrow \neg switchOff \tag{3.1}$$

$$switchOff \leftarrow \neg switchOn \tag{3.2}$$

$$light \leftarrow switchOn \tag{3.3}$$

This program consist of three atoms $switchOn$, $switchOFF$ and $light$. The first rule states that if the switch is not off then the switch is on. The second rule states that if the switch is not on then the switch is off. The last rule states that if the switch is on then there is a light. This problem consists of two stable models as solutions. The first solution consists of one atom

2. This example was inspired by [BL13].

$\{switchOff\}$ and the second solution consists of two atoms $\{switchOn, light\}$. Let's inspect the first solution. Clearly, all rules are satisfied by this model but only the body of the second rule is satisfied. Hence, the reduct consists of the second rule only. Since the empty set does not satisfy the reduct, $\{switchOff\}$ is a stable model. The second solution can be verified similarly but this time the reduct consists of the first and the third rule, which means that both $switchOn$ and $light$ have to be included in the model of the reduct.

3.2.1 ASP extensions

Next, further extensions [Sim99] to logic programs are described, which are frequently used in practice and ease modeling problems with ASP.

Choice rules

A *choice rule* has form

$$\{h_1; \ldots; h_o\} \leftarrow b_1 \wedge \cdots \wedge b_m \wedge \neg b_{m+1} \wedge \cdots \wedge \neg b_n$$

where $0 \leq o$ and each h_i is an atom. Unlike with the normal rule above, a choice rule can be used to prove any subset of the atoms h_1 to h_o whenever its body is satisfied.

Example 3.2.2. One nice aspect of choice rules is that they allow for writing more compact programs. Remember that in the previous example we needed two atoms to model that a switch is on or off. Choice rules allow for a more compact and easier to read representation. The program

$$\{switchOn\} \leftarrow$$
$$light \leftarrow switchOn$$

has the same solutions as the program in Example 3.2.1 without using the extra atom $switchOff$ to capture that the switch is off. Since the first choice rule can be used to prove any subset of $\{switchOn\}$ there are two solutions: $\emptyset$ and $\{switchOn, light\}$.

Integrity constraints

An integrity *constraint* has form

$$\leftarrow b_1 \wedge \cdots \wedge b_m \wedge \neg b_{m+1} \wedge \cdots \wedge \neg b_n$$

It cannot be used to prove atoms, but instead to remove candidate solutions satisfying its body.

Example 3.2.3. Coming back to our light switch domain, we model our toy problem just using choice rules and integrity constraints.

$$\{switchOn; light\} \leftarrow$$
$$\leftarrow light \wedge \neg switchOn$$
$$\leftarrow \neg light \wedge switchOn$$

The first rule can prove arbitrary subsets of $\{switchOn, light\}$. Then the second integrity constraint discards all solutions where the light is on but the switch is not. Similarly, the last integrity constraint prunes all solutions where the switch is on but the light is not. Hence, we obtain the same answer sets as in Example 3.2.2.

Cardinality aggregates

A cardinality aggregate can be used to express that a set of atoms has to have a certain cardinality. A *cardinality aggregate* has form

$$l\ \{e_1; \ldots; e_n\}\ u$$

where $n \geq 0$, the lower bound l and upper bound u are integers, and each element e_i has form $l_0 : l_1, \ldots, l_m$ for $m \geq 0$ and each l_j is a literal. An aggregate element e_i is satisfied whenever the literals l_0 to l_m are satisfied. The head literals of satisfied elements are collected in a set whose cardinality has to be between l and u (inclusive) for the cardinality aggregate to be satisfied. One of the bounds can be omitted; if the lower bound is omitted, then the cardinality has to be smaller than u and greater than l, otherwise. Cardinality aggregates can occur in both rule heads or rule bodies. In a rule head they can be used to prove the literals l_0 of true aggregate elements. This leads to another way to model the light domain example.

Example 3.2.4. The program

$$1\ \{switchOn; switchOff\}\ 1 \leftarrow$$
$$light \leftarrow switchOn$$

has the same solutions as Example 3.2.1. Here, the first rules states that exactly one of the atoms $switchOn$ and $switchOff$ has to be true.

Sum aggregates

A *sum aggregate* has form

$$l\ sum\{e_1; \ldots; e_n\}\ u$$

where $n \geq 0$, the lower bound l and upper bound u are integers, and each element e_i has form $t_0, \ldots, t_o : l_1, \ldots, l_m$ for $o \geq 1$, $m \geq 0$, each t_k is a term and each l_j is a literal. Unlike cardinality aggregates, sum aggregates allow for expressing that the sum of the integers given by a set of tuples has to be between a lower and upper bound. An aggregate element is satisfied if the literals l_0 to l_m are satisfied. In this case, the term tuple $t_0, \ldots, t_o$ contributes the integer t_0 to sum up [3]. Unlike cardinality aggregates, this form of aggregate can only be used in a rule body. Like with cardinality aggregates, at least one bound has to be specified.

Example 3.2.5. The following program extends the light switch scenario:

$$\{on(s_1); on(s_2); on(s_3)\} \leftarrow$$
$$light \leftarrow 2\ sum\{1, s_1 : on(s_1); 1, s_2 : on(s_2); 1, s_3 : on(s_3)\}$$

Each switch has an associated weight and the sum of weights of active switches has to be larger than 2 for the light to turn on. Let's consider the solution $\{on(s_1), on(s_3), light\}$. This solution satisfies the first and third aggregate element, hence, we obtain the set of tuples $\{(1, s_1), (1, s_3)\}$. Both tuples in this set have weight 1 and thus their sum is 2, which satisfies the lower bound. This allows us to derive atom $light$ and we indeed have a solution.

So far we have not considered variables in aggregates. The above example can actually be written more compactly by introducing a variable in the aggregate.

$$\{on(s_1); on(s_2); on(s_3)\} \leftarrow$$
$$light \leftarrow 2\ sum\{1, S : on(S)\}$$

Variable S in the aggregate is a local variable because it occurs only between the braces $\{\ldots\}$. In this case, this does not lead to multiple ground rules but in the three aggregate elements as in the preceding program by substituting s_1, s_2, and s_3 for S.

3.2.2 Optimization statements

An objective function to maximize or minimize a lexicographically ordered tuple of sums can be described in the ASP program using optimization statements. For example, a minimize

3. If t_0 is not an integer it is simply ignored.

statement can be expressed as:

$$minimize\{e_1; \ldots; e_n\}$$

where $n \geq 0$ and e_i has form $t_0@p_0, t_1, \ldots, t_o : l_0, \ldots, l_m$ for $o \geq 1$, $m \geq 0$, p_0 is a term, each t_k is a term and each l_i is a literal. The accumulation of integers works almost exactly as with sum aggregates. The only difference is the optional priority p_0 (which defaults to 0) in the elements e_i. Sums are calculated for each priority individually and then ordered by priority and compared lexicographically when determining optimal solutions.

$$\begin{aligned} \{max(a, 20); max(b, 30)\}\ 1 \leftarrow \\ 1\ \{min(c, 30); min(d, 10)\} \leftarrow \\ maximize\{W@1, X : max(X, W)\} \\ minimize\{W@2, X : min(X, W)\} \end{aligned}$$

In the above program, priority levels indicate that minimization (with priority level 2) is more important than maximization (with priority level 1). Solution $\{max(b), min(d)\}$ induces the tuple of sums $(10@2, 30@1)$, which is indeed an optimal solution for the above program because at most one atom over $max/1$ and at least one atom over $min/1$ can be true.

Graph coloring problem

Here, we give a logic program for the graph coloring problem (defined for ILP in Section 2.4.4). We first describe the input graph in the form of facts. Predicate $node/1$ determines the available nodes and predicate $edge/2$ the edges between the nodes. The example graph in

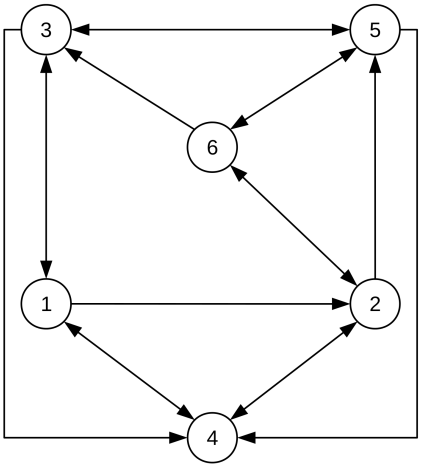


Figure 3.2 – Directed graph

Figure 3.2 is defined by the following facts:

$$
\begin{array}{lll}
node(1..6) \leftarrow & & \\
edge(1,2) \leftarrow & edge(1,3) \leftarrow & edge(1,4) \leftarrow \\
edge(2,4) \leftarrow & edge(2,5) \leftarrow & edge(2,6) \leftarrow \\
edge(3,1) \leftarrow & edge(3,4) \leftarrow & edge(3,5) \leftarrow \\
edge(4,1) \leftarrow & edge(4,2) \leftarrow & \\
edge(5,3) \leftarrow & edge(5,4) \leftarrow & edge(5,6) \leftarrow \\
edge(6,2) \leftarrow & edge(6,3) \leftarrow & edge(6,5) \leftarrow
\end{array}
$$

We use rule 3.4 to define that each node can have a different color. So predicate $color/1$ corresponds to variable y in the ILP formulation of the graph coloring problem. Another rule (3.5) is added to assign exactly one color to each node. Here, predicate $assign/2$ corresponds to variable x in the ILP formulation and the rule captures Equations 2.11 and 2.12. Next, rule 3.6 is added to ensure that two adjacent nodes must not have the same color. This rule corresponds to Equation 2.13 in the ILP formulation. Rule 3.7 is used to eliminate isomorphic solutions w.r.t. the selection of colors [4]. It ensures that colors are selected successively beginning with the first color ($color(1)$). Having a rule like this helps when enumerating all optimal solutions by drastically reducing the resulting set of solutions. Note that there are still more isomorphic solutions that could be pruned with additional rules. Finally, the optimization statement 3.8 is used to describe the objective function. This objective function minimizes the number of colors used and corresponds to Equation 2.10 in the ILP formulation.

$$\{color(C)\} \leftarrow node(C) \tag{3.4}$$

$$1\ \{assign(U,C) : color(C)\}\ 1 \leftarrow node(U) \tag{3.5}$$

$$\leftarrow edge(U,V) \wedge assign(U,C) \wedge assign(V,C) \tag{3.6}$$

$$\leftarrow color(C) \wedge \neg color(C-1) \wedge node(C-1) \tag{3.7}$$

$$minimize\{1, C : color(C)\} \tag{3.8}$$

By specifying the above constraints, an ASP solver computes 6 optimal stable models. One of the computed stable models is shown below and depicted in Figure 3.3:

$$\{assign(1,1), assign(2,3), assign(3,3), assign(4,2), assign(5,1), assign(6,2)\}$$

4. Here an important feature of ASP is used: terms can contain arithmetic operations. Such operations are evaluated during the grounding phase, e.g., $C-1$ evaluates to 2 if C is three.

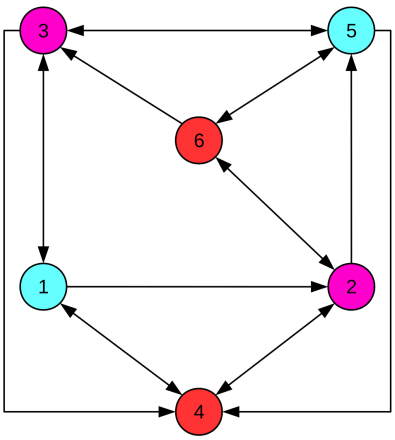


Figure 3.3 – Directed graph after coloring

3.3 Related work

Here, we discuss the related work of finding optimal Boolean Networks (BNs) from PKN and phosphoproteomic data using ASP. We also discuss the related work of finding diverse solutions of a logic program having a large solution space.

3.3.1 Learning Boolean networks

Videla et al.

Method description. Videla *et al.* [VGE+12] proposed an approach based on ASP to infer BNs from PKNs and experimental data. It takes three inputs a PKN, perturbations, and experimental observations. A PKN specifies the canonical information about connections among proteins existing in the experimental data. Their method can only handle PKNs without any feed-back loops. A perturbation consists of a combination of stimuli and inhibitors. Experimental observations consist of measurements of readouts proteins. They model the combinatorial optimization problem of finding BNs using facts, rules, and the objective function. The objective function takes two parameters into account: the fitness of the model to the data, and the model size. They derive the structure of the model (BN) from the PKN. Then, the values of readouts are calculated for each model by fixing the value of stimuli and inhibitors. The optimization is applied on these models to minimize the fitness score and the size of the model.

Case study. Videla *et al.* illustrated their approach on mid and large scale datasets, compared results with the genetic algorithm optimization framework [SRAE+09] (Section 2.4.2). Furthermore, they started with building two PKNs (P_1 and P_2 for mid and large scale case studies) representing growth and inflammatory signaling from literature. These PKNs were compressed to remove non-observable and non-controllable nodes, creating four compressed

networks. Then, hyper-graphs compatible to these four compressed networks were created. A hyper-graph refers to the extension of the graph with AND and OR gates. For each compressed network, five hyper-graphs (of three different sizes) were selected randomly to generate 240 in silico datasets. The size of the hyper-graph is calculated by summing up the number of source nodes for all child nodes.

Results. Videla *et al.* inferred 8 global optimal models for the mid-scale case study and 2 global optimal models for the large-scale case study. They also executed the same problem with the CellNOpt system ([SRAE+09]) and the comparison is drawn in Table 3.2. The biggest advantage of ASP over genetic algorithms is the calculation of an optimal score and the enumeration of all optimal solutions having this score. They also studied the impact of size on the solution space (number of models). They concluded that if they relax the size limit of the model then the size of the solution space increases exponentially. They also discovered that both approaches are able to find all global optimal solutions in 240 datasets. However, results suggest that the models of minimal sizes are best identified with an ASP solver. Computation time was also compared with CellNOpt and results showed that ASP solvers are faster than CellNOpt by 5 orders of magnitude.

Table 3.2 – Comparison of ASP solver and CellNOpt as presented by Videla *et al.*

	Mid-scale case study		Large-scale case study	
	ASP solver	CellNOpt	ASP solver	CellNOpt
Execution Time	0.03 s	9.2 h	0.07 s	27.8 h
Best Solution	0.06 s (optimal)	7.2 h	0.04 s (optimal)	24.5 h
Number of models	8 (optimal)	66	2 (optimal)	206
Size of model	16	16 to 24	26	27 to 36

Ostrowski et al.

Method description. Ostrowski *et al.* [OPS+15] proposed a new method (caspo time series: *caspo-ts*)[5] to learn Boolean models from phosphoproteomic time series data given a PKN. A family of candidate BNs compatible with the PKN is exhaustively enumerated. Afterwards an over-approximation constraint is used to filter out BNs which cannot reproduce the phosphoproteomic dataset. An objective function is used to minimize the distance between the actual time series and the time series produced by the BNs. Because of the over-approximation criteria, some of the returned BNs may not reproduce the time-series traces. Such false positives

5. This approach is based on ASP and model checking as described in the Figure 1.2 (formal description is given in Chapter 3).

can be ruled out using model checking. The remaining BNs are true positive. True positive BNs are guaranteed to reproduce all traces of the phosphoproteomic time-series data.

Case study. The authors illustrated the working of this approach on three case studies. For each case study they used three different versions of PKNs. Hyper-graphs compatible with these PKNs were used to generate in-silico datasets with 10 perturbations. The first case study (C_1) represents EGF-TNFα signaling ([MTH$^+$12]). Case study C_1 consists of 13 nodes and 16 edges. The second case study (C_2) represents TCR signaling ([KSRL$^+$06]). From this case study, three different case studies ($C_{2.1}, C_{2.2}, C_{2.3}$) were created by modifying the experimental design. Case study $C_{2.1}$ consists of 14 nodes and 22 edges, $C_{2.2}$ consists of 16 nodes and 25 edges, and $C_{2.3}$ consists of 40 nodes and 58 edges. Third case study C_3 represents the ERBB receptor regulated G1/S transition model ([SFL$^+$09]. Case study C_3 consists of 19 nodes and 50 edges. Finally, they utilized two optimization schemes (cardinality and subset) of the ASP solver to derive minimal models for these case studies.

Results. Their findings suggest that *caspo-ts* is able to find the first optimal solution within seconds while the full enumeration can take longer. They also observed that the full enumeration of optimal solutions in the case of C_3 and $C_{2.3}$ can take hours because the search space is huge. Next, they used model checking to determine the true and false positive rate of inferred models. The true positive rate was high ($\geq 78\%$) in all case studies except $C_{2.3}$ and C_3. They also verified the sensitivity the *caspo-ts* to the modified and incomplete PKN for case study C_1. For this, they generated different versions of PKNs by adding, removing and replacing edges randomly. Results depicted sufficient robustness to the changes in the PKN. They also compared the results with the previously proposed approach ([VGE$^+$12, GVE$^+$13]) using case study C_1. They used the approach of Videla *et al.* to learn BNs with one time point and compared the fitness error with *caspo-ts*. Results show that *caspo-ts* outperforms Videla *et al.* approach to find an optimal model with the data in most of the experiments.

Conclusion

Regarding inferring BNs, the methods [SRAE$^+$09, VGE$^+$12] (of Videla and Saez) restrict themselves to learn BNs from two time points (start, end), assuming the system has reached an early steady-state when measurements are performed. This assumption prevents one from capturing interesting characteristics like loops as shown in [MTH$^+$12]. To overcome this issue, Ostrowski *et al.* proposed the *caspo-ts* method. The first part of this thesis is in the context of applying and ameliorating the *caspo-ts* method on a real case study of four breast cancer cell lines.

3.3.2 Learning diverse solutions

Another important part of my work is focused on the improvement of the *caspo-ts* search method in case of a large case study. The *caspo-ts* system uses a backtracking algorithm (described in Chapter 5, Section 5.2.3) to exhaustively generate solutions. It can lead to a situation where successive solutions share very similar properties. This can be problematic, especially in the case of a large solution space where discovering or analyzing all solutions becomes computationally hard. To resolve this issue, a diverse enumeration scheme has been introduced. In the following, we focus on discussing the work on finding diverse or similar solutions using ASP; similar work for constraint programming and SAT solving can be found here [HHOW05a, Nad11a].

Eiter et al.

Method description. Eiter *et al.* [EEEF09] proposed an approach to study similar or diverse solutions using ASP. They have introduced online/offline methods to enumerate diverse solutions.

The offline methods are based on computing all solutions in advance, while the online methods follow an incremental solving approach. For finding diverse or similar solutions in the case of the offline methods, they start with a complete set of solutions. These solutions form the nodes in a graph and distance-labeled edges between nodes represent the distances between solutions. The idea is to find a clique in this graph using edges with distances greater than k to find diverse solutions and edges with distances less than k to find similar solutions. This clique is computed using ASP.

They introduced three variations of online methods. The first variation encodes everything in one program consisting of three parts: (1) compute n solutions in parallel, (2) compute distances between solutions, and (3) remove answers where the distance between any two solutions is not equal to k. The second online method does not calculate solutions in advance but rather updates the solution set with a new solution in each run with a solution that has exactly distance k from all solutions found so far. The third variation of the online methods is similar to the second method but modifies the ASP solver *clasp* to incorporate a distance function in it.

Results. Eiter *et al.* illustrated this approach on phylogeny reconstruction explaining the evolutionary development of biological species or other entities (taxonomic units). They built diverse or similar phylogenies (solutions) for Indo-European languages. The methods are used to find diverse or similar solutions among eight phylogenies.

The first offline and online methods guarantee to find an optimal solution since they consider the whole solution space of a given problem, while online methods two and three cannot

guarantee an optimal solution given the fact that the solving process is based on the first solution found. In their experiments, the offline method three performed better than all other methods.

Zhu et al.

Method description. Zhu *et al.* [ZT13] considered Eiter's approach in a more general setting where optimal solutions are calculated based on user preferences. Their approach first identifies optimal solutions and then computes a set of diverse or similar solutions. They studied three methods to solve this problem.

The first method is an iterative algorithm using an additional program to test optimality. This program decides if a computed answer set is optimal according to the preference rules. The current answer set is updated if the next solution is better than the current one. At the end, an optimal answer set is returned. Next, to find an alternative optimal answer set, their algorithm finds another answer set which is also optimal and different from the solutions found so far. Similar or diverse solutions are found by adding distance constraints. This process continues iteratively to find further solutions.

The second method is based on introducing a modification to ASP solver *clasp* so that it checks the optimality of the answer sets using a tester program during the search. Otherwise, it is equivalent to the previous method.

Both of the previous methods are based on multiple calls to the ASP solver while the third method models all requirements in one disjunctive logic program. In general, such programs are more complex to solve than normal logic programs.

Results. Zhu *et al.* illustrated the working of their approach on four computational problems: finding one optimal solution, an alternative optimal solution, three similar optimal solutions, and eight diverse optimal solutions. They compared their results on the basis of ranked (where one preference is prioritized over the other) and unranked preferences.

The results show that the iterative method performs best with unranked preferences when finding optimal solutions while the third method (based on modeling) performs well for finding similar or diverse solutions. However, for ranked preferences the third method is slower than the iterative method.

Romero et al.

Method description. Romero *et al.* [RSW16] introduced a framework (asprin 2) to compute diverse or similar optimal solutions in case of computational problems with preferences. They introduced three methods to find diverse optimal solutions: (1) enumeration, (2) replication, and (3) approximation.

The enumeration method first finds all solutions for a given problem and then computes diverse solutions.

The replication method converts a logic program into an equivalent logic program with a maxmin preference; diverse optimal solution are then computed for the rewritten program.

Finally, the approximation method iteratively identifies diverse optimal solutions w.r.t. previously found solutions. They implemented three different variations of this method. The first variation is based on maximizing the minimum distance to previously found solutions. The second variation calculates a partial interpretation by maximizing the minimum distance to previously found solutions and then calculates a model close to this partial interpretation. And the third variation uses a heuristic to derive diverse optimal solutions.

Results. They illustrate the working of their approach on datasets from six different domains: biological network repair, metabolic network expansion, crossing minimization, circuit diagnosis, design space exploration, and timetabling.

The enumeration and replication methods are not as effective as the approximation methods since enumeration and replication methods traverse through an exponential number of optimal solutions to find an exact solution. The approximation techniques do not promise to provide an exact result but performs well in case of the large scale problems. Results suggest that the third variation of the approximation method performs better than the others in terms of computation time. They also compare the quality of the diversification where the third method based on heuristics performed worse than the other methods. A clear trade-off emerged between quality and performance.

Conclusion

The best performing approach of Eiter *et al.*, [EEEF09] modifying the *clasp* solver, cannot guarantee to produce optimal solutions as the solving process is biased toward the first enumerated solution. Moreover, they did not consider optimization problems. Hence, their methods are not directly applicable to *caspo-ts*, since it enumerates optimal (subset minimal) solutions in order to produce simpler and more relevant solutions. Zhu *et al.* [ZT13] can handle programs with preferences and also proposes modifications to the *clasp* solver but their approach becomes impractical as the search space increases. Therefore, we decided to extend the approach of [RSW16] for computing optimal diverse solutions in ASP. The novelty of this extension is that our approach uses heuristics for both the computation of optimal (subset minimal) solutions and the diversification. In the thesis, we refer to the modified *caspo-ts* as *caspo-ts*^{D}.

4

Computational Discovery of Dynamic Cell Line Specific Boolean Networks from Multiplex Time-Course Data

4.1 Introduction

In this chapter, we improve and apply the caspo time series (*caspo-ts*) method conceived to identify optimal BNs from time-series phosphoproteomic measurements and a prior knowledge network. The article related to this work is published in the Plos Computational Biology journal [RPS+18].

We use a large-scale real-case study from the HPN-DREAM challenge. This dataset contains phosphoproteomic time series data of four Breast Cancer cell lines. Here, we show that the discrete approach, based on Answer Set Programming and model-checking techniques, provides accurate results. We identify optimal networks explaining the dynamic behavior of the experimental data for a network of 64 components and 178 interactions. Our models have a Root Mean Square Error (RMSE) of 0.31 with respect to the testing data. It is well known that the methods addressing dynamic properties on large networks do not scale well because of the considerable size of the respective state transition graph. Using *caspo-ts*, we identify the BNs given the inherent combinatorial explosion that appears when considering real case data. Our key findings suggest that this method is capable of constructing cell line specific Boolean models, which is extremely valuable given the heterogeneity of breast cancer due to many genetic

modifications. Furthermore, we also validate the cell line specific regulatory mechanisms using the inferred Boolean models.

4.2 Caspo-ts framework

We chose the *caspo-ts* method [OPS$^+$15, OPS$^+$16] for the inference of BNs. This method was tailored to handle phosphoproteomic time series data. The input of the method consists of a prior knowledge network (PKN) and normalized phosphoproteomic time series data under different perturbations. It generates a family of BNs whose structure is compatible with the PKN and that can also reproduce the patterns observed in the experimental data. The *caspo-ts* workflow is shown in Figure 4.1. It consists of two main steps, ASP solving and model checking. Here, we formally describe the inputs, outputs, solving and model checking processes of the *caspo-ts* framework.

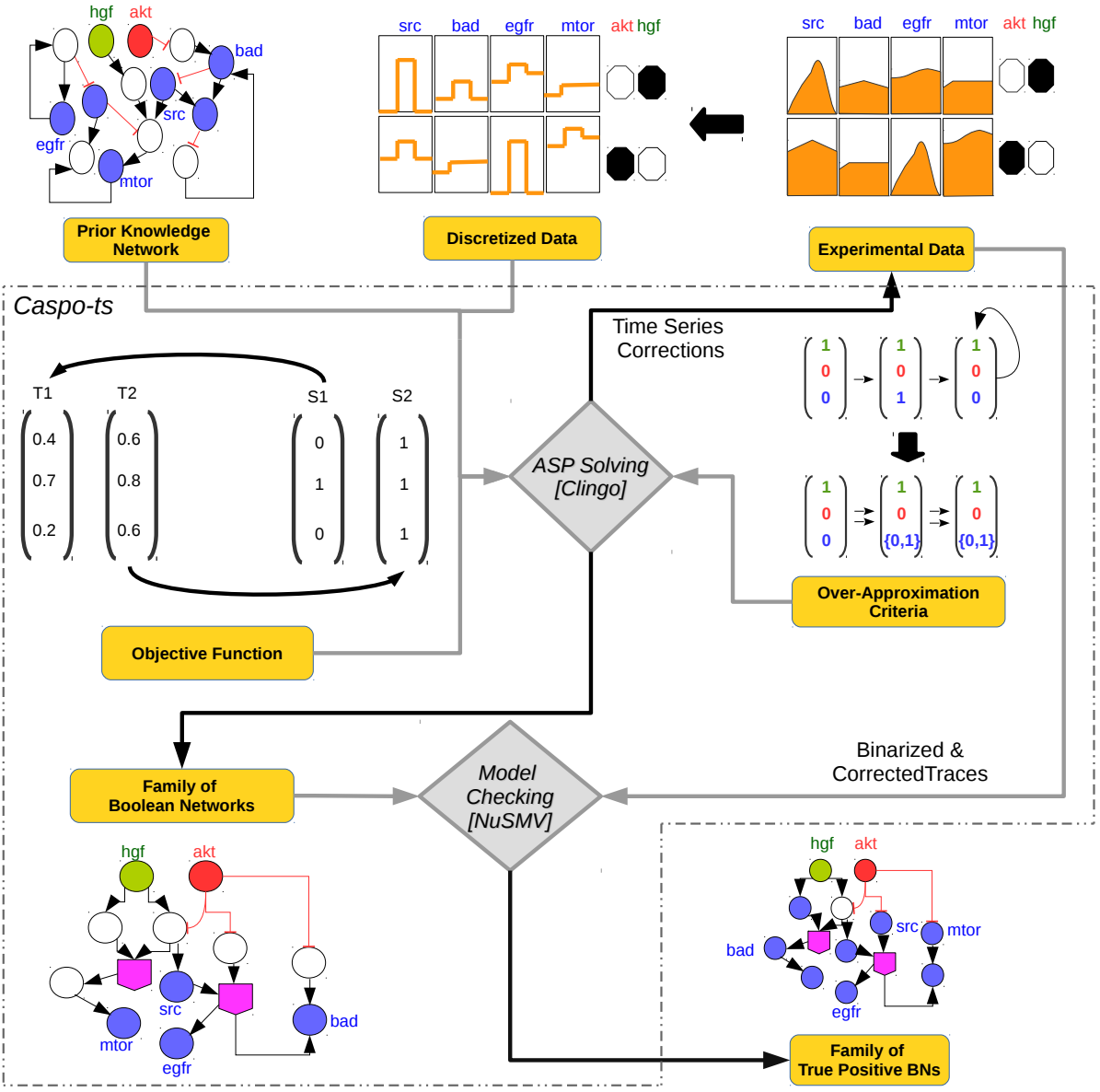
4.2.1 Prior knowledge network

A prior knowledge network (PKN) is modeled as a labeled (or colored) directed graph (V, E, σ) with $V = \{v_1, v_2, \ldots, v_n\}$ the set of nodes, $E \subseteq V \times V$ the set of directed edges and $\sigma \subseteq E \times \{+1, -1\}$ the signs of edges. The set of nodes is denoted by $V = S \cup I \cup R \cup U$ where S are stimuli, I are inhibitors, R are readouts, and U are unobserved nodes. Stimuli, inhibitors, readouts, and unobserved nodes are encoded by different colors in the graphs. Stimuli are shown in green, inhibitors in red, readouts in blue, and unobserved nodes in white (Figure 4.1). Moreover, the subsets S, I, R, U are all pairwise disjoint except for I and R. Stimuli are used to bound the system and also serve as interaction points of the system, these nodes can be experimentally stimulated, *e.g.* cellular receptors. Inhibitors are those nodes which remain inactive or blocked over all time points of the experiment by small molecule inhibitors. Stimuli and inhibitor nodes take Boolean values $\{0, 1\}$, representing the fact that the node was stimulated (1) or inhibited (0). Readouts are experimentally measured given a combination of stimuli and inhibitors. They usually take continuous values in [0;1] after normalization. Unobserved nodes are neither measured nor experimentally manipulated. In this study, we use the term *perturbation* to refer to the combination of stimuli and inhibitors, similarly to other studies such as [Hei16, HHC$^+$16, HNJC$^+$17]. Thus, the set of all possible perturbations is the set of sets $\mathcal{P} = (S \cup I)$. The PKN is one of the main inputs of the *caspo-ts* framework.

4.2.2 Phosphoproteomic time series data

Phosphoproteomic time series data are obtained by measuring the temporal changes in phosphorylated proteins under a perturbation (Figure 4.1). Without loss of generality, we as-

Figure 4.1 – *Caspo-ts* workflow. *Caspo-ts* receives as input data a prior knowledge network (PKN) and a discretized phosphoproteomic dataset. In this example the phosphoproteomic data consists of two perturbations involving *akt* (inhibitor) and *hgf* (stimulus): 1) $akt = 0$, $hgf = 1$, and 2) $akt = 1$, $hgf = 0$. A black colored perturbation means the inhibitor or stimulus was perturbed (1) while white represents the opposite (0). Readouts are specified in blue and describe the time series under given perturbations. Using this input data, *caspo-ts*, performs two steps: ASP solving and model checking. In the ASP solving step: (i) a set of BNs compatible with the PKN is generated, (ii) afterwards an *over-approximation constraint* is imposed upon each candidate BN to filter out invalid BNs that do not result in an over-approximation of the reachability between the Boolean states given by the phosphoproteomic dataset, and finally (iii) BNs are optimized using an objective function minimizing the distance to the experimental measures. The ASP step also introduces repairs in some data points of the time series that added penalties to the objective function. These corrected traces will be given to the model checker. In the model checking step, the exact reachability of all the (binarized and corrected) time series traces in the family of BNs is verified.



sume that the time series data are related to the observation of $m \leq n$ nodes for the nodes $\{v_1, \ldots, v_m\}$ (so the nodes $\{v_{m+1}, \ldots, v_n\}$ are not observed). The observations consist of normalized continuous values: a time series of k data points is denoted by $T_P = (t_P^1, \ldots, t_P^k)$, where $P \subseteq S \cup I$ is a perturbation and $t^j \in [0;1]^m$ for $1 \leq j \leq k$. These data will be discretized in order to link it with further BNs discovery (see the ASP solving and model checking steps). Figure 4.2 depicts an example of phosphoproteomic time series data. The phosphoproteomic dataset is the second input of the *caspo-ts* system.

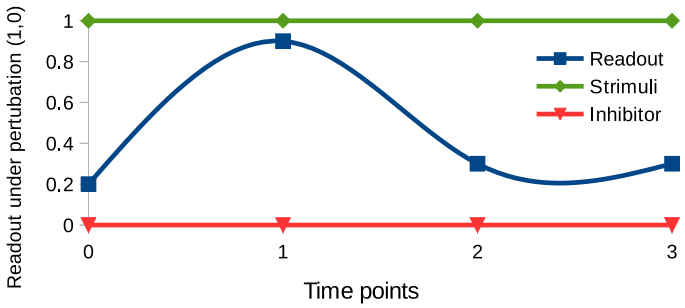


Figure 4.2 – Phosphoproteomic time series data. Here the values between zero and one of three proteins are shown in different colors. In this figure, we see the time series of one readout protein (blue) under a perturbation of one stimulus (green) and one inhibitor (red). Stimuli have value 1 and inhibitors have value 0 across all time points of an experimental perturbation. Readouts take continuous values in [0;1] after normalization. In some phosphoproteomic datasets an inhibitor can also act as a readout protein, which means that there are perturbations where it will be measured.

4.2.3 Boolean network

A Boolean Network (BN) [Kau69, Ino11] is defined as a pair $B = (N, F)$, where

— $N = \{v_1, \ldots, v_n\}$ is a finite set of nodes (or variables/proteins/genes),

— $F = \{f_1, \ldots, f_n\}$ is a set of Boolean functions (regulatory functions) $f_i : \mathbb{B}^k \to \mathbb{B}$, with $\mathbb{B} = \{0, 1\}$, describing the evolution of variable v_i.

A vector (or *state*) $x = (x_1, \ldots, x_n)$ captures the values of all nodes N at a time step, where x_i represents the value of the node v_i, and is either 1 or 0. There are up to 2^n possible distinct states for each time step. Next, we define the transition $x \to x'$ between two states of a BN. If there is no update for node v_i then $x'_i = x_i$. If there is an update for node v_i then the state of a node v_i at the next time step is determined by $x'_i = f_i(x_1, \ldots, x_n)$. Note that usually only a subset of the nodes influence the evolution of node v_i. These nodes are called the *regulatory nodes* of v_i. The state of each node can be updated in a synchronous (parallel) or asynchronous fashion. In the synchronous update schedule, the states of all nodes are updated, while in asynchronous update schedule, the state of one node can be updated at a time. The work presented in this article is independent of the update schedule routine, hence any number of nodes can be updated at a

time. We use $x \rightarrow^* y$ to say that state y can be reached from state x with an arbitrary number of steps. Notice that the BN is the main output of the *caspo-ts* method.

4.2.4 ASP solving

PKN expansion

Given a PKN and a phosphoproteomic dataset, a family of candidate BNs, compatible with this PKN, is exhaustively enumerated. The BNs are represented by Boolean formulas in Disjunctive Normal From (DNF), i.e., as a disjunction of conjunctive clauses. A clause can be seen as a reaction, where the proteins represented positively are available, and the proteins represented negatively are absent. A Boolean formula in DNF encompasses all possible reactions to update the value of a protein. We refer to these BNs as subset minimal BNs.

Here, we use two properties to describe the BNs compatible with the PKN [VGE+12]. Intuitively, the first property states that if there is an edge in a BN then this edge must exist in the PKN as well. The second property states that the BNs use the smallest DNF formulas possible, in the sense that no conjunctive clause can be removed from a DNF formula without changing the Boolean function it represents. It means a Boolean formula should have as few nodes as possible.

1. Evidence property:

   Given a PKN (V, E, σ) and $v \in V$, a logical formula ϕ in DNF has an evidence in (V, E, σ) with respect to v if and only if for every propositional variable w that occurs positively (resp. negatively) in ϕ, there exists an edge $(w, v) \in E$ and $((w, v), 1) \in \sigma$ (resp. $((w, v), -1) \in \sigma$).

2. Redundancy property:

   Given a logical formula ϕ in DNF, with $\phi = \bigvee_{j \geq 1} c_j$ where each c_j is a conjunctive clause, ϕ is a redundant formula if and only if for some $k, l \geq 1$ with $k \neq l$ and some logical conjunction r it holds that $c_k = c_l \wedge r$.

Over-approximation

After generating a set of BNs (compatible with a PKN), a constraint is imposed to check the reachability of the states from another within state graph of a BN. Since this reachability is a computationally hard problem (PSPACE-complete) [CEP95], we use an over-approximation criteria. This constraint is imposed upon each candidate BN to filter out invalid BNs, that do not result in an over-approximation of the reachability between the Boolean states given by the phosphoproteomic dataset.

A meta-state $u = (u_1, u_2, \ldots, u_n)$ is a vector of dimension n over non-empty subsets of $\mathbb{B}$, noted $\mathbb{M} = \{\{0\}, \{1\}, \{0, 1\}\}$; the set of meta-states is $\mathbb{M}^n$. Meta-states characterize a set of Boolean states: a state $x \in \mathbb{B}^n$ belongs to a meta-state u, written $x \in u$, iff each Boolean component x_i belongs to the set u_i. Given a state x, we use $\bar{x}$ for the corresponding meta-state $(\{x_1\}, \ldots, \{x_n\})$. We define the transition relation $u \rightrightarrows v$ between the meta-states u and v as follows: $u \neq v$ and $v = (u_1, \ldots, u_i \cup \{f_i(x) \mid x \in u\}, \ldots, u_n)$ for some $1 \leq i \leq n$.

In [OPS+16], it has been shown that if y is reachable from x ($x \rightarrow^* y$) then there exists a meta-state u such that $y \in u$ and $\bar{x} \rightrightarrows^* u$. This definition is further refined to describe the necessary condition for reachability called support consistency. A state x is support consistent with state y denoted by $x \rightsquigarrow^* y$, if and only if there exists a meta-state u with $\bar{x} \rightrightarrows^* u$ such that $y \in u$ and for all $1 \leq i \leq n$ either

— $y_i \neq x_i$, or

— $y_i = x_i$ and $u_i \neq \{0, 1\}$, or

— $y_i = x_i$, $u_i = \{0, 1\}$, and there exists $z \in u$ such that $f_i(z) = y_i$.

If state y is reachable from state x ($x \rightarrow^* y$) then $x \rightsquigarrow^* y$. Since we are using the over-approximation criteria, it is possible that some of BNs may fail to reproduce some trajectories of the time series data. These BNs are called false positive (FP). To filter out the false positive BNs, exact model checking is applied.

Optimization

The goal is to generate a BN that can reproduce the experimental data as well as possible. For this purpose, BNs are optimized by defining an objective function minimizing the distance between the actual time series, T_P, and the predicted time series, Y_P:

$$RMSE = \sqrt{\frac{1}{m * k * |\mathcal{P}|} \sum_{i=1}^{m} \sum_{j=1}^{k} \sum_{P \in \mathcal{P}} ((t_P^j)_i - (y_P^j)_i)^2}, \tag{4.1}$$

where m is the number of observed nodes, k is the number of time points, and $\mathcal{P} \in S \cup I$ is the set of perturbations. If a BN can verify the time series traces of experimental data then it will have minimal $RMSE$. In addition, the optimization step highlights the data points in the time series which added penalties to the RMSE. Such data points are automatically corrected before the model checking step.

All the analyses described in this step are performed using ASP, using the `clingo` 4.5.4 solver [GKKS14]. This solver guarantees finding optimal solutions, and all BNs outputted by the ASP solver step will be identically optimal.

4.2.5 Model checking and true positive BNs

From the previous step, a set of optimal BNs that over-approximate the phosphoproteomic time series data is produced. This set of BNs is verified with exact *model checking* to detect true positive (TP) BNs. *TP BNs* are guaranteed to reproduce all the (binarized) trajectories under all perturbations by verifying exact reachability in the BN state graph. For this, we have used computational tree logic (CTL) implemented in the `NuSMV` 2.6.0 [CCG+02], which is a symbolic model checker. Figure 4.3 describes the model checking process of the *caspo-ts* system. It takes the Boolean Model and properties as input and generates two types of output: (1) true, which means the property is satisfied, or (2) false, with a counter example which means the property is not satisfied.

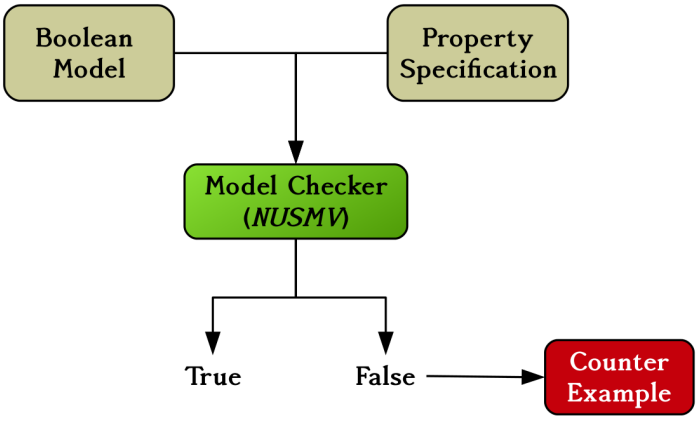


Figure 4.3 – Model checking process.

Properties are written using the existential (E) and finally (F) operators. The existential quantifier asserts that the property holds for at least one path. The finally operator asserts that the property will hold eventually. They are specified in the following manner:

$$Exp_e^{t_0} \rightarrow EF(Exp_e^{t_1} \wedge EF(Exp_e^{t_2} \wedge EF(\ldots EF(Exp_e^{t_n})\ldots))), \tag{4.2}$$

where $Exp_e^{t_i}$ refers to the state under perturbation $e \in \mathcal{P}$ at time point t_i. The above property (4.2) states that there exists a way to reach time point t_n from time point t_0 in the perturbation e. We specify properties for all perturbations ($\mathcal{P}$). If a BN verifies all these properties then it is a true positive (TP) BN. If any of the properties is dissatisfied then the BN is called a false positive (FP).

4.2.6 Computation of root mean square error

The *caspo-ts* method can also generate the RMSE for the data absent (testing data) from the training data. It generates over-approximated traces of the BN and compares them with

the traces existing in the testing data (see Algorithm 1). For this computation it uses the same equation as given in Equation 4.1. There are two types of RMSEs returned by *caspo-ts* - discrete and model. The *discrete RMSE* is imposed by the discretization of the method. Since we use a discrete learning approach, our recovered traces will be in {0,1} and this introduces an error with respect to continuous measurements in [0;1]. The *model RMSE* refers to the learned BN error with respect to the normalized time series data; that is, the model RMSE is at least as large as the discrete RMSE. When the difference between these two is zero then the inferred BNs are able to recover the discrete trajectories without any error. If the model RMSE is greater than the discrete RMSE then the inferred BNs have some errors in the recoverability of the discrete time series data. The delta denotes the difference between discrete and model RMSE. The lower value of delta means that the BN can successfully reproduce the traces of the experimental data with minimum error possible.

Data: BNs and Time Series Traces
Result: RMSE scores
1 **for** *each model* i *in B* **do**
2 Generate over-approximated traces for i ;
3 Calculate the distance between the over-approximated traces of i and the actual traces using Equation 4.1;
4 Minimize the distance between the over-approximated and the actual traces;
5 **end**

Algorithm 1: Computing Root Mean Square Error.

Note that, this Algorithm 1 is given to illustrate a general idea about RMSE calculation. The ASP program does more than generating the over-approximated values of proteins of BNs. It also optimizes the minimum distance between over-approximated Boolean traces and actual time series traces. It guarantees to capture the minimum distance that is possible of Boolean traces of BN to the actual time series traces.

4.3 Caspo-ts logic program

Here, we provide the pseudo logic program to describe the input data given in the *caspo-ts* modeling framework Section. The logic program in written in the ASP language. ASP is a powerful declarative logic programming language for knowledge representation and reasoning [Lif08]. The basic idea is to encode the problem using a non-monotonic logic program and then feed it into the ASP solver, which computes the solution of the problem in the form of models (also known as answer sets). Note that we only provide pseudo encodings here, please refer to [OPS+16] for details.

4.3.1 Modeling PKN and data

Facts, rules and constraints are the building blocks of ASP programs. Here we use facts to describe the inputs. The PKN (V, E, σ) is described by the following facts:

$$node(v) \leftarrow \text{for } v \in V$$
$$edge(u, v, s) \leftarrow \text{for } (u, v) \in E \text{ and } ((u, v), s) \in \sigma$$

Facts over the predicate $node$ are used to model the nodes existing in the PKN. A fact $edge(u, v, s)$ is used to describe an edge between node u and v with sign s.

For each perturbation $P \in \mathcal{P}$ and phosphoproteomic time series T^P, we have the following facts:

$$clamped(P, v, 0) \leftarrow \text{for } v \in P \cap I$$
$$clamped(P, v, 1) \leftarrow \text{for } v \in P \cap S$$
$$obs(P, j, v_i, s) \leftarrow \text{for } s = (t_j^P)_i, 1 \leq j \leq k \text{ and } 1 \leq i \leq m$$

Facts over predicate $clamped$ set the value of node v in perturbation P to either 0 or 1 depending on whether the node belongs to inhibitor set (I) or stimuli set (S) respectively. Facts over predicate obs set the value of the readout node v_i in perturbation P at time point j to value s.

4.3.2 Over-approximation

To filter the compatible BNs with the over-approximation criteria, we start with generating Boolean values for each node of the BN with the following rule:

$$1\{state(P, T, U, (0; 1))\}1 \leftarrow P \in \mathcal{P}, node(U), T = 1..k.$$

Here, $state$ predicate is implemented as a choice rule. It has four arguments. The first argument represents the perturbation P, the second represents the time-point T, the third is the node U, and the fourth represents the guessed value for the the node U. States are subject to further constraints which are omitted here for brevity.

Further, we calculate the meta-states from the given states of the BNs with the following rules:

$$meta(P, T, U, V) \leftarrow state(P, T, U, V)$$
$$update(P, T, U, V) \leftarrow \text{“node U takes value V if its formula can evaluate to V”}$$
$$meta(P, T, U, V) \leftarrow update(P, T, U, V)$$

The first rule initializes the meta-states from the *state* predicate. The second rule selects possible updates by evaluating the Boolean functions associated with the nodes. The third rule updates the current meta-state with the *update* predicate. We omit the exact encodings here and refer to the literature [OPS+16] for details.

The following integrity constraints encode the support consistency condition described in the over-approximation Section:

$$\leftarrow state(P, T+1, U, V), \neg meta(P, T, S, V)$$
$$\leftarrow state(P, T+1, U, V), \neg update(P, T, U, V), update(P, T, U, W), V \neq W$$

The first rule ensures that the state at time-point $T+1$ is in the meta-state reachable from time-point T. The second rule implements the condition in the third item in the definition of support consistency described in the over-approximation (see Section 4.2.4). It states that if a state at time-point $T+1$ has value V and in the update predicate it has value W then there must exist an update in-between to recover the final value V.

4.3.3 Objective function

Here, we describe the objective function to minimize the difference between the phosphoproteomic time series data and the guessed time series for the BN. The following rules declare the objective function by using $minimize$ directive:

$$minimize\{M - 49, P, T, U : obs(P, T, U, M), state(P, T, U, 0), M \geq 50\}$$
$$minimize\{50 - M, P, T, U : obs(P, T, U, M), state(P, T, U, 1), M < 50\}$$

If the conditions after the " : " do not evaluate to true, then the guessed time series does not match with the corresponding measurements (phosphoproteomic data) and a penalty is accumulated by the minimize statement. Note that since ASP does not support continuous values, we converted them to integers in the range from 0 to 100 by using the formula: $\lfloor x*100 \rfloor$.

4.4 Graph similarity measure

Here, we present a graph similarity measure in order to check the variability among the family of BNs generated by *caspo-ts*. This measure serves to compare the reactions existing in a gold standard network (A) with a family of BNs ($\mathcal{B}$) and is based on the Jaccard similarity coefficient which measures the similarity of these models (see Algorithm 2).

Data: Gold Standard Network (A) and Family of Boolean Networks ($\mathcal{B}$)

Result: Similarity index for each model in the family of Boolean networks

1 Initialize two structures A and $\mathcal{B}$;
2 **for** *each model* M *in* $\mathcal{B}$ **do**
3 | Calculate the intersection of A with M;
4 | Apply Jaccard similarity coefficient ($J(A, M)$) ;
5 **end**

Algorithm 2: Graph Similarity Algorithm

The Jaccard index between A and M is defined as length of the intersection divided by the union:

$$J(A, M) = \frac{\mid A \cap M \mid}{\mid A \cup M \mid} = \frac{\mid A \cap M \mid}{\mid A \mid + \mid M \mid - \mid A \cap M \mid} \tag{4.3}$$

4.5 HPN-DREAM challenge case study

4.5.1 Data acquisition

The DREAM portal provides unrestricted access to complex, pre-tested data to encourage the development of computational methods. In this study, we are focused on the HPN-DREAM challenge, which was motivated by the fact that the same perturbation may lead to different signaling behaviors in different backgrounds, making it necessary to build a model which can perform unseen predictions (absent from the learning data). The main goal of the HPN-DREAM challenge is to learn signaling networks efficiently and effectively to predict the dynamics of breast cancer [HHC$^+$16, HNJC$^+$17].

Learning data

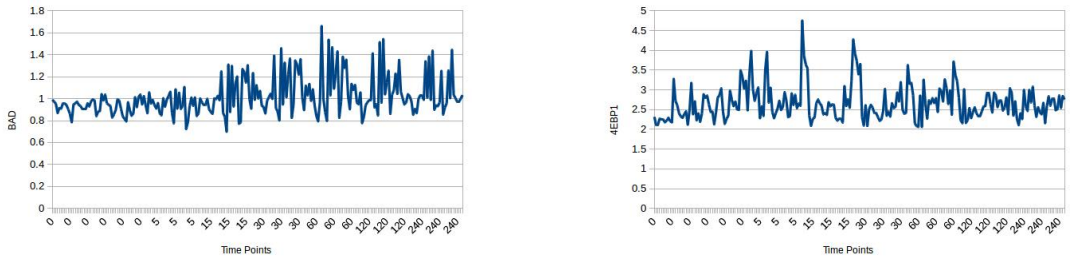
Reverse Phase Protein Array (RPPA) quantitative proteomics technology was used for generating the dataset of this challenge. The measurements focus on short term changes on up to 45 proteins and their phosphorylation over 0 to 4 hours. The HPN-DREAM dataset includes temporal changes in phosphorylated proteins at seven different time points ($t_1 = 0$min, $t_2 = 5$min, $t_3 = 15$min, $t_4 = 30$min, $t_5 = 60$min, $t_6 = 120$min, $t_7 = 240$ min). The learning data consists of four cancer cell lines (BT20, BT549, MCF7 and UACC812) under different perturbations (8 stimuli and 2 or 3 inhibitors). The number of perturbations varies from 24 to 32 depending on the cell line. In each cancer cell line up to 45 phosphorylated proteins are measured against different sets of perturbations over multiple time scales.

Testing data

Test data is available for assessing the performance of networks learned from the learning data. The HPN-DREAM portal provides testing data for four cancer cell lines (BT20, BT549, MCF7 and UACC812) under different perturbations (8 stimuli and 1 inhibitor). They contain gold standard datasets of time series predictions of up to 45 proteins having the same time scale as learning data [Hei16, HHC+16, HNJC+17]. The number of perturbations varies from 7 to 8 depending on the cell line. These data will be used to test the quality of the BNs inferred by *caspo-ts*.

Normalization

The protein measurements of the HPN-DREAM challenge have variable ranges as depicted in Figure 4.4. The values of 4EBP1 are oscillating between 2 and 5 while the values of BAD are oscillating between 0.6 and 1.8.



(a) BAD (b) 4EBP1.

Figure 4.4 – Protein values over time. Here, the x-axis represents the time points and the y-axis represents the protein measurements.

Maximum value based normalization was used to set the measurements between a common scale, *i.e.,* 0 and 1, in order to assign activation or inactivation values to variables or species of the BN. Equation (4.4) describes the formula used for the normalization. Given time series T^P, we obtain time series T'^P:

$$(t'^P_j)_i = \frac{(t^P_j)_i}{\max\{(t^Q_l)_i \mid Q \in \mathcal{P}, 1 \leq l \leq k\}}, \tag{4.4}$$

where $i \in \{1, \ldots, m\}$ are the measurements, $j \in \{1, \ldots, k\}$ are time-points, and $P \in \mathcal{P}$ are the perturbations. Here $(t^P_j)_i$ represents the value of protein i under perturbation P at time-point j and the denominator denotes the highest value of protein i under all perturbations and time-points.

4.5.2 Data preprocessing

The learning and testing datasets used in this study were extracted from the HPN-DREAM challenge and correspond to time series protein measurements upon different perturbations of four breast cancer cell lines (UACC812, BT20, BT549, and MCF7) [HNJC$^+$17, HHC$^+$16]. Since readout signals are measured on variable ranges depending on the protein, a normalization step was necessary. The learning dataset had a few noisy, inconsistent and incomplete time series data points. The *caspo-ts* system identified these inconsistencies existing in the time series data. A recurrent experimental inconsistency observed was an oscillation in the protein signal upon experimental inhibition of the same protein.

To resolve the above mentioned issues, we performed the following data preprocessing steps on the learning dataset:

1. Set the protein values between a common scale, *i.e.,* 0 and 1, using a maximum-value-based normalization scheme (4.4).
2. For time point 0 the expression of some readout proteins under some perturbations was not available. Thus, control experimental readings were used as the time point 0 for such proteins.
3. In some cases readout measurements were duplicated for the same time point. To solve this noise issue we chose one time point arbitrarily.
4. We removed inconsistent perturbations where the protein AKT was inhibited and was having a dynamic behavior as a readout protein.
5. We considered only perturbations with complete time series data, since guessing the missing time points automatically with *caspo-ts* for this case study will be computationally expensive.

The experimental errors pointed in steps 2-5 were raised as warning or exceptions by *caspo-ts*. Steps 1 to 5 were applied to the learning dataset. Only step 1 was applied to the testing dataset. After removing perturbations with inconsistent behaviors or incomplete time series from the learning dataset, we had 15, 13, 13 and 18 perturbations for MCF7, BT20, BT549 and UACC812 cell lines respectively measuring 23 readouts. Figure 4.5 shows the preprocessed dataset for cell line BT20.

4.5.3 Prior knowledge network

Our first result was to build the consensus protein signaling network associated with the HPN-DREAM challenge dataset. The structure of the prior knowledge network (PKN) was generated by mapping the experimentally measured phosphorylated proteins (HPN-DREAM dataset) to their equivalents from literature-curated databases and connecting them together

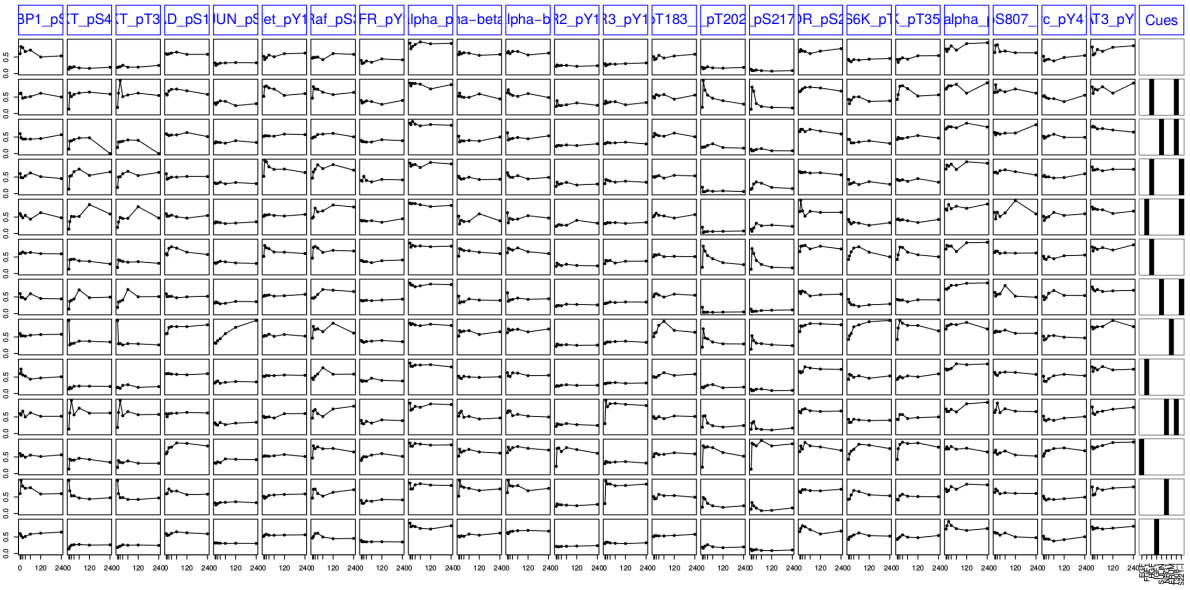


Figure 4.5 – BT20 cell line dataset. Cues are the perturbations representing combination of stimuli and inhibitors. The x-axis shows the time point while the y-axis shows the normalized protein measurements.

within one network. The PKN (Figure 4.6) was built using the ReactomeFIViz app (also called the ReactomeFIPlugIn or Reactome FI Cytoscape app) [WDD+14b], which accesses the interactions existing in the Reactome and other databases [WDD+14b, WDD+14a]. The PKN shown in Figure 4.6 consists of 64 nodes (7 stimuli, 3 inhibitors, and 23 readouts) and 178 edges.

4.5.4 Cell line specific Boolean networks

In this section, we show the generated BNs for each cell line. For this, we used *caspo-ts* to learn the BNs from the PKN (Figure 4.6) and the phosphoproteomic data of four breast cancer cell lines - BT20, BT549, MCF7, and UACC812. We inferred a family of cell line specific BNs for each cancer cell line. As explained in the *caspo-ts modeling framework* (section 4.2), the *caspo-ts* method produces BNs fulfilling two criteria, (i) satisfaction of the over-approximation criteria and (ii) optimality with respect to the RMSE objective function. ASP-optimal solutions were fast to collect, their computation time ranged from 36 seconds to 3 minutes depending on the cell line, as shown in the Table 4.1.

Afterwards, these ASP-optimal BNs were given to the model-checker for further verification. This second step is more complex and we put a restriction on the computation time of 7 days for each cell line. The number of verified BNs varies from one cell line to another, depending on a number of factors such as the number of perturbations, the order of answer sets in the solutions space, and the perturbation order. The total number of verified ASP-optimal BNs within the 7 day time-frame were 188, 231, 52, and 150 for the BT20, BT549, MCF7 and

Figure 4.6 – Breast cancer signaling pathway. This figure shows the reconstructed signaling network from a combination of databases. An arrow shows the positive regulatory relationship between two proteins, while a T shaped arrow indicates inhibition. Green nodes are stimuli, blue nodes are readouts, white nodes are unmeasured or unobserved, and blue nodes with a red border represent inhibitors and readouts at the same time. Note that in the node labels, we have added the phosphorylation sites to the protein names in order to connect them to the experimental measurements.

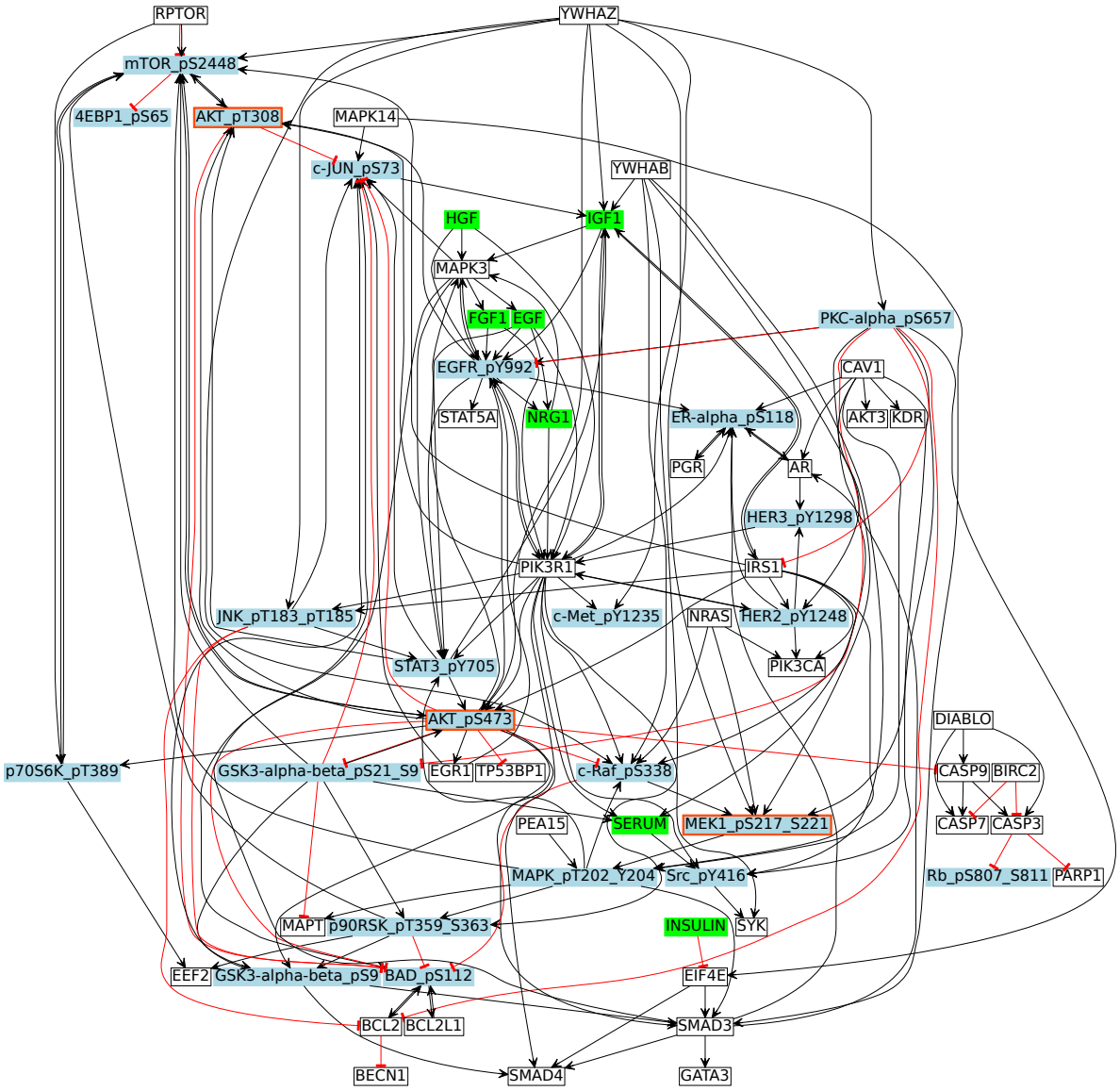


Table 4.1 – Computation summary. Here, we show the number of verified solutions, true positive and false positive BNs, and their computation (ASP solving and Model Checking steps) time for each cell line. The ASP solving was performed on a standard laptop machine. The model checking task was performed on a cluster with 560 cores and 1.9 Tb of RAM.

Cell Line	**Number of Solutions**	**True Positives**	**False Positives**	**Time**	
				ASP solving	**Model Checker**
$BT20$	188	72	116	210 seconds	$<$ 7 days
$BT549$	231	191	40	93 seconds	$<$ 7 days
$MCF7$	52	21	21	36 seconds	$<$ 7 days
$UACC812$	150	0	150	197 seconds	7 days

UACC812 cell lines respectively. We obtained 72, 191, and 21 true positive BNs for BT20, BT549, and MCF7 cell lines respectively with an optimal fit to the data. For the UACC812 cell line, we were unable to obtain true positive BNs within the 7 day time limit for verification. Hence, we kept the first 20 BNs from the 150 ASP-optimal BNs for the UACC812 cell line. Union of these BNs are shown in Figures (4.7, 4.8, 4.9 and 4.10). It is worth noting that we generated true positive BNs for UACC812 cell line by allowing the model checker to run without bounding it to the 7 day time limit. This result pointed us to the fact that the similar BNs are clustered together in the solution space generated by *caspo-ts*. We analyze this aspect and present the detailed results of the UACC812 cell line in the chapter 5.

An aggregated network was built (Figure 4.11) by combining the BN families (with 191, 21, 72, and 20 BNs for BT549, MCF7, BT20, and UACC812 cell lines respectively) obtained for the four cell lines by keeping the hyper-edges (Boolean functions) having a frequency higher than 0.3 within each BN family. The frequency is calculated by counting the number of common Boolean functions and dividing it by the total number of Boolean functions within the BN family of each cell line. This aggregated network contains 34 nodes and 74 Boolean functions involving 36 AND gates. As compared to the PKN (Figure 4.6), the inferred networks are highly specific to each cell line. In Figure 4.11, all cell lines share only 4% of Boolean functions which are shown in thick black colored edges. This shows that the inferred BNs of these four breast cancer cell lines are very diverse and different from each other.

To measure cell line similarity, we calculated the similarity score by applying the Graph Similarity Measure on the family of BNs (with 191, 21, 72, and 20 BNs for BT549, MCF7, BT20, and UACC812 cell lines, respectively). This algorithm receives two parameters as input: (1) one gold standard BN and (2) a family of BNs. It outputs a score in $[0; 1]$, measuring the average of the similarity scores between each BN in the family and the gold standard BN. In our case, the gold standard BN is the aggregation of one family of BNs. The similarity scores between all pairs of breast cancer cell lines are shown in Table 4.2. Figure 4.11 agrees with the results presented in Table 4.2 as we can see the clear discrepancies among the four cell lines. It

Figure 4.7 – Union of BNs of BT20. Here, we show the union of BNs for the cell line BT20. This network is generated by combining 72 true positive BNs. It contains 31 nodes and 41 boolean functions with 12 AND gates. There are 2 stimuli, 2 inhibitors and 21 readouts.

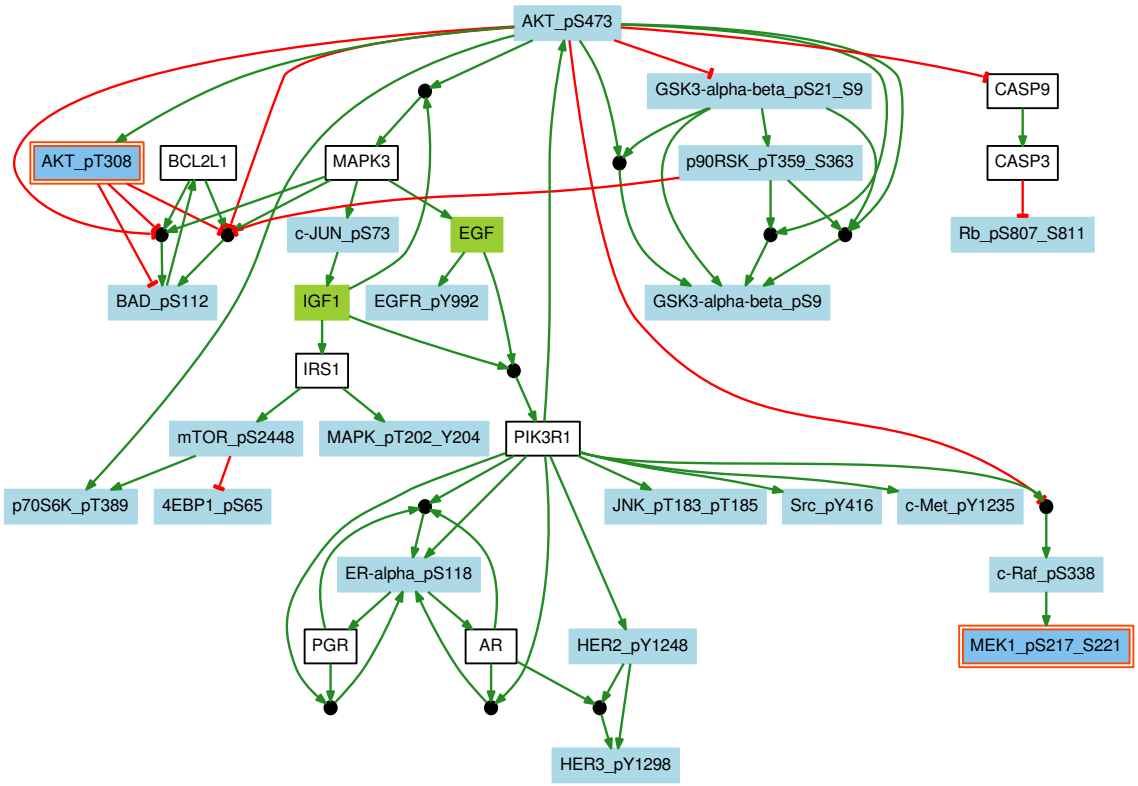


Figure 4.8 – Union of BNs of BT549. Here, we show the union of BNs for the cell line BT549. This networks is generated by combining 191 true positive BNs. It contains 28 nodes and 53 boolean functions with 35 AND gates. There are 5 stimuli, 2 inhibitors and 17 readouts.

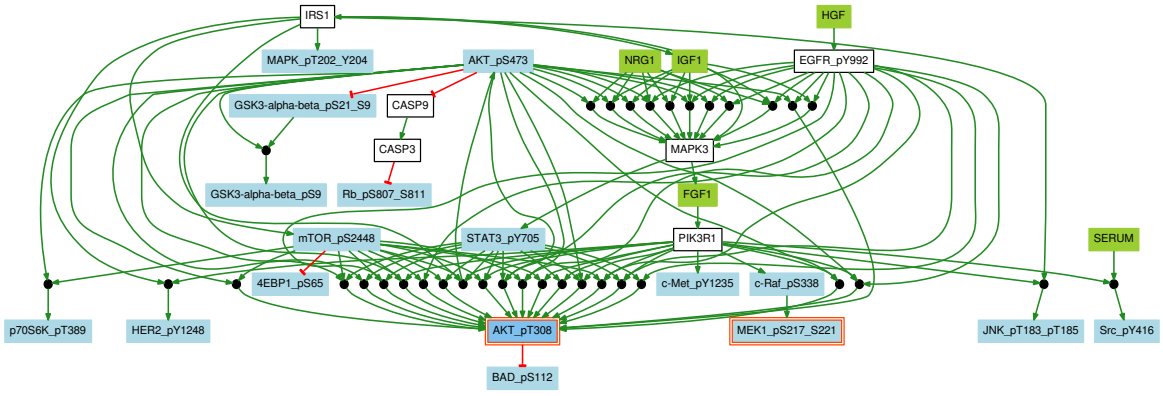
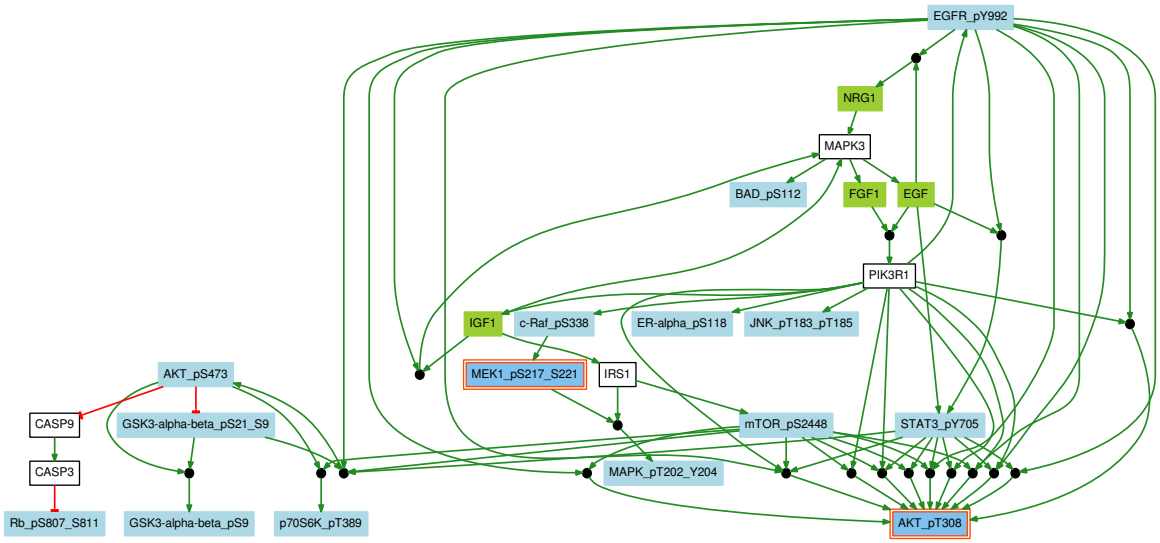


Figure 4.9 – Union of BNs of MCF7. Here, we show the union of BNs for the cell line MCF7. This network is generated by combining 21 true positive BNs. It contains 24 nodes and 37 boolean functions with 19 AND gates. There are 4 stimuli, 2 inhibitors and 15 readouts.



can be seen that 23% of the Boolean functions are shared among BT549 and MCF7, and also between BT20 and UACC812. BT20 shares the least number of Boolean functions (15%) with BT549. This table revealed pronounced differences among different cell lines of breast cancer. We also analyzed the diversity of Boolean functions among the family of BNs within the same cell line. The similarity among Boolean functions from BT20 (0.73) and MCF7 (0.63) is higher than the ones from BT549 (0.43) and UACC812 (0.46) cell lines.

Table 4.2 – Similarity scores among breast cancer cell lines.

Cell Lines	**Size of BNs' family**	**Similarity Score**			
		BT20	**BT549**	**MCF7**	**UACC812**
$BT20$	72	0.73	0.15	0.17	0.23
$BT549$	191	**	0.43	0.23	0.20
$MCF7$	21	**	**	0.63	0.21
$UACC812$	20	**	**	**	0.46

4.5.5 Heterogeneity among cell lines

There are a total of 69 distinct Boolean functions shown in Figure 4.12 along with their respective frequencies. It is interesting to note that the B549 and UACC812 cell lines have

Figure 4.10 – Union of BNs of UACC812. Here, we show the union of BNs for the cell line UACC812. This network is generated by combining 20 BNs. It contains 33 nodes and 54 boolean functions with 29 AND gates. There are 6 stimuli, 2 inhibitors and 18 readouts.

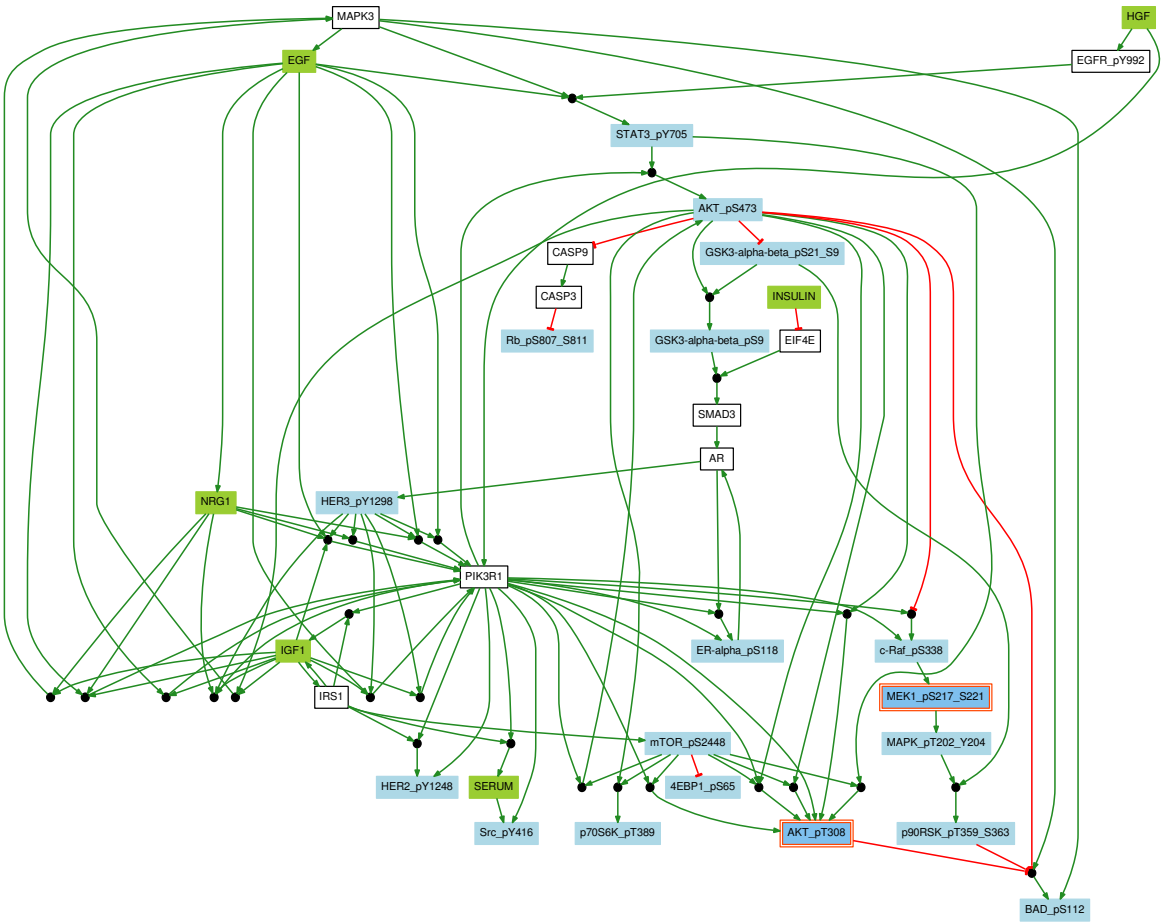
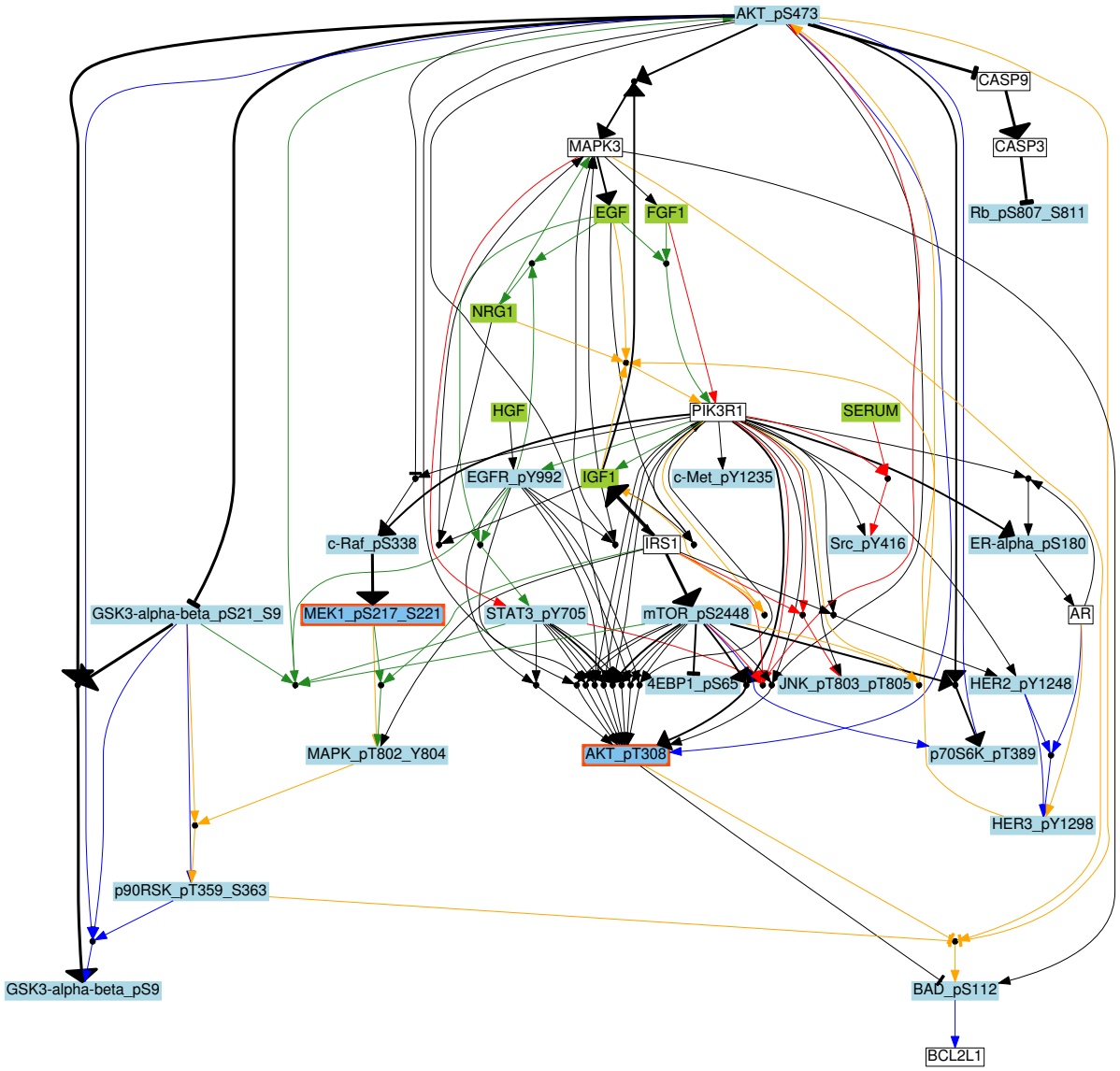


Figure 4.11 – Boolean network of breast cancer cell lines. The aggregated graph for all cell lines. Blue, red, green and brown colors have been used for each cell line BT20, BT549, MCF7 and UACC812, respectively. The nodes are connected by logic gates (AND or OR) to their direct predecessors. Edges are used to show influences ($\rightarrow$ for positive and $\dashv$ for negative). An AND gate is depicted by a small black circle where the incoming edges correspond to the inputs of the gate. An OR gate is depicted by multiple incoming edges to the node. A different color scheme is used to represent different types of nodes. The green color is for stimuli, the red for inhibitors, the blue for readouts, and the white for unobserved nodes. Black edges denote common hyper-edges across cell lines; the thickness of the black hyper-edge denotes the number of cell lines sharing this hyper-edge.



more distinct models among their family of BNs with a variable frequency range. This shows that these cell lines have different mechanisms agreeing with the results obtained through graph similarity measure given in Table 4.11.

Figure 4.13 shows the common Boolean functions along with their frequency in all BNs. Interestingly, only 4% of the Boolean functions are shared in all cell lines and 99% of these shared functions have the same frequency. In this figure, there is only one Boolean function which is frequent in 3 cell lines and has a lower frequency in BT20.

4.5.6 Biological literature related to the Boolean functions discovered by *caspo-ts*

The *caspo-ts* method revealed that cell line specific reactions are clustered around the *AKT*, *MAPK3*, and *PIK3R1* proteins. *PI3K* is an important factor for cancer development in HER2 amplified cancers (UACC812) as compared to non-HER2 amplified (BT20, BT549 and MCF7) cancer cell lines. The HER2 amplified tumor is more aggressive than other types of tumor (see Chapter 2 Section 2.1.1). We can see from Figs 4.7, 4.8, 4.9 and 4.10 that *PIK3R1* exists in all cell lines but is rather more connected in the UACC812 cell line with 10 incoming edges, while in others with only 1 incoming edge. The *PIK3R1* node in UACC812 (Figure 4.10) has a centrality measure of 0.37 while in the other three cell lines the centrality measure is less than 0.11. The centrality measure is used to quantify the most important node within a network i.e., the number of times a node has been used as a bridge (along the shortest path) to connect to other nodes in the network [AGW14].

It has been established that *P1K3R1* (the regulatory unit of PI3K) plays an important role in suppressing tumors (breast, lung, ovarian, bladder, and prostate) [SWF$^+$05, TWK$^+$10, SHGAL$^+$08]. Recently, it has been found that *PIK3R1* is mutated in 3% of breast cancer cell lines [N$^+$12]. Therefore, it is worth studying the impact of the *PIK3R1* regulatory unit in breast cancer.

4.5.7 Evaluation

The performance of the *caspo-ts* method is evaluated using three criteria: 1) RMSE calculation using a typical learning and testing data approach, 2) random data comparison, 3) AUROC (Area Under the Operating Curve) score. The BNs are learned using the learning dataset only. The prediction accuracy is evaluated by comparing the RMSE of trajectories in the testing dataset with those recovered by the learned networks (see Equation 4.1 in Section 4.2.4). To check how our method performs in case of random time series, we calculated the *RMSE* score for random data and compared it with the one obtained using the learning and testing data. Next, the validity of these networks was verified by comparing them with the canonical MTOR

Figure 4.12 – Heterogeneous Boolean functions. The Boolean functions are represented on the y-axis and the frequency of each Boolean function is shown on the x-axis. A Boolean function, or hyper-edge, is of the form $node \leftarrow expr$, where $node$ is the receiver of the Boolean clause $expr$ in the BN. In the Boolean clause, the *not* operator is represented by a "!" symbol and the AND operator by a "+" symbol. The disjunction of clauses is represented by multiple reactions upon the same receiver node.

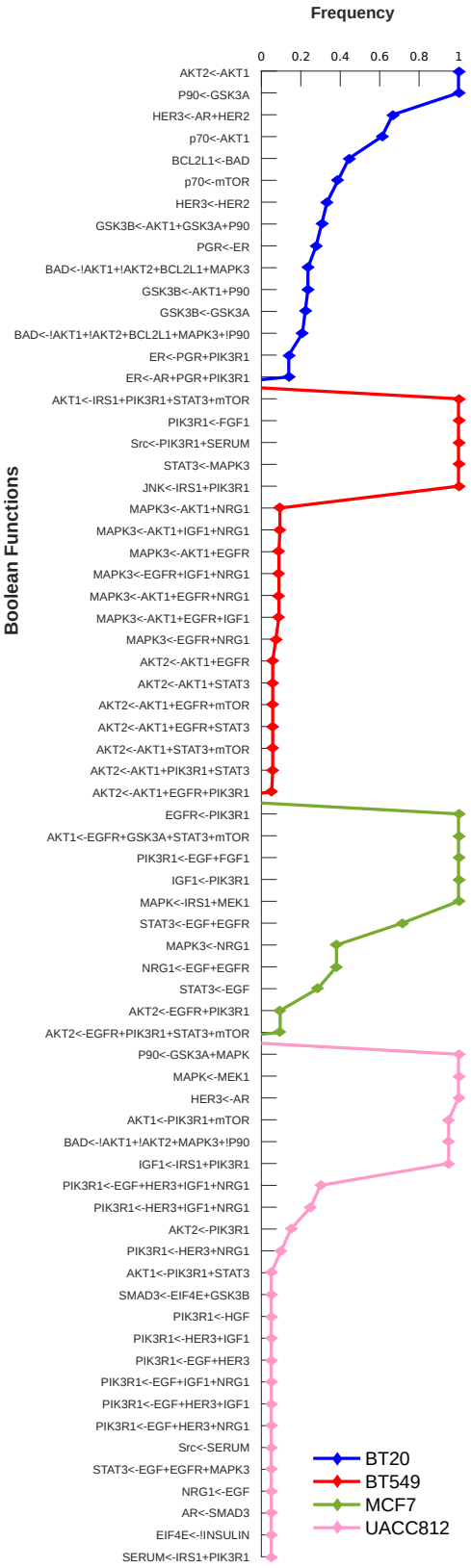
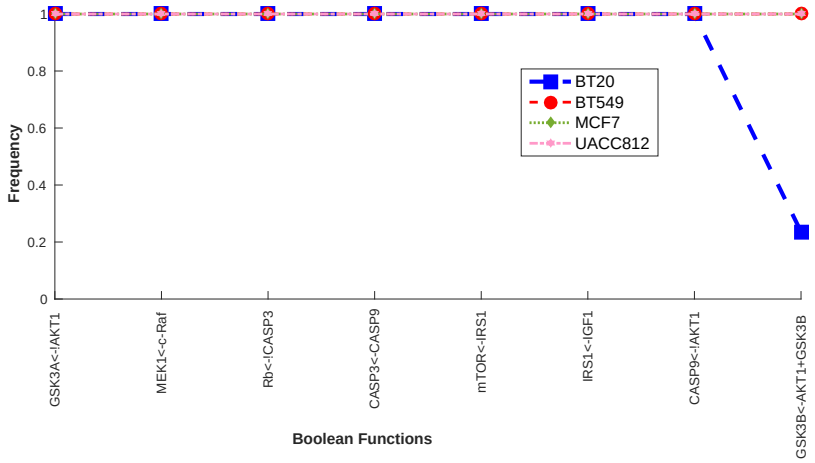


Figure 4.13 – Common Boolean functions across all four cell lines. The Boolean functions are represented on the x-axis and the frequency of each Boolean function is shown on the y-axis.



signaling pathway using two parameters, *i.e.,* true positive rate (TPR) and false positive rate (FPR).

Validation using root mean square error criteria

The goal is to verify that the BNs learned using *caspo-ts* applied to the HPN-DREAM dataset are able to recover trajectories that do not exist in the learning data. For this purpose, we used experimental testing data to check the specificity of the trajectories of the proposed networks. These testing data were provided by the HPN-DREAM challenge organizers. Table 4.3 shows the corresponding RMSE in case of learning and testing data. It can be seen that the inferred BNs are able to produce the discrete trajectories without any error in the learning dataset for all cell lines. These BNs can reproduce trajectories of the learning data with an error of 0.3464, 0.3498, 0.3207, and 0.3464 in BT20, BT549, MCF7, and UACC812 cell line respectively. It is encouraging to see that the inferred BNs are able to recover the discrete testing trajectories without any error in MCF7, and with a minimal error of 0.0009, 0.0106, and 0.0094 in BT20, BT549, and UACC812, respectively. These BNs can reproduce trajectories of the testing data with an error of 0.3302, 0.3113, 0.2772, and 0.3178 in BT20, BT549, MCF7, and UACC812 cell line, respectively.

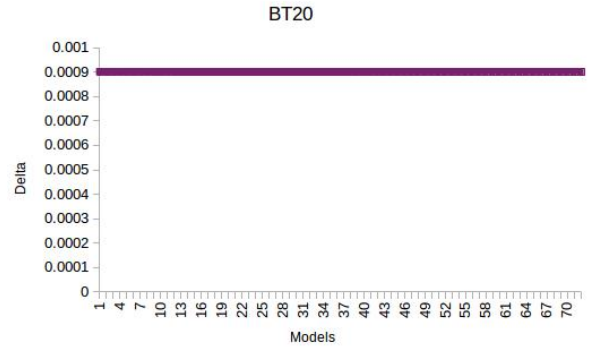
We also compared the RMSE score with the top two best performers of the HPN-DREAM challenge. We got the top position with an RMSE score of 0.31 as compared to their RMSE scores of 0.47 and 0.50. Notice that in comparison to other HPN-DREAM challenge methods based on Bayesian inference, Regression, and Granger Causality among others, *caspo-ts* does not make new predictions but it checks the recoverability of the testing trajectories with the inferred BNs (see Section 4.2.6).

Table 4.3 – **Root mean square error.** This table summarizes the RMSE results for each cell line. We have calculated the discrete RMSE (error related to the discretization of the data) and the model RMSE (*caspo-ts* error). The Delta column shows the difference between model and discrete RMSE.

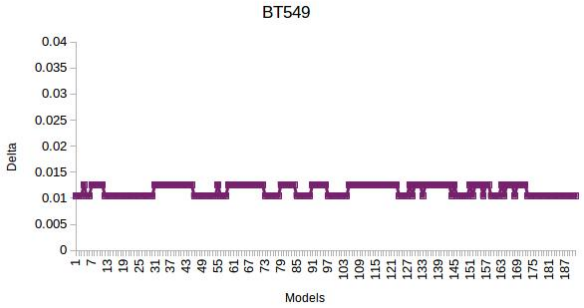
Cell Line	Learning			Testing		
	Discrete	**Model**	**Delta**	**Discrete**	**Model**	**Delta**
$BT20$	0.3464	0.3464	0	0.3293	0.3302	0.0009
$BT549$	0.3498	0.3498	0	0.3007	0.3113	0.0106
$MCF7$	0.3207	0.3207	0	0.2772	0.2772	0
$UACC812$	0.3464	0.3464	0	0.3084	0.3178	0.0094

Analyzing family of BNs using root mean square error

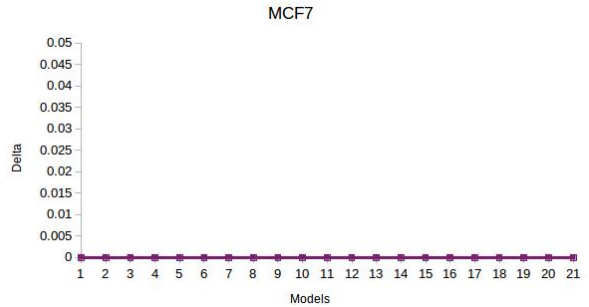
We calculated the RMSE score for each Boolean model in a family of BNs of each cell line. We show these results in Figure 4.14. We observe that the all models of MCF7 and BT20 cell lines have the same RMSE w.r.t testing data. The family of BT549 cell line has an average RMSE score of 0.0116 and UACC812 cell line has an average RMSE score of 0.0119.



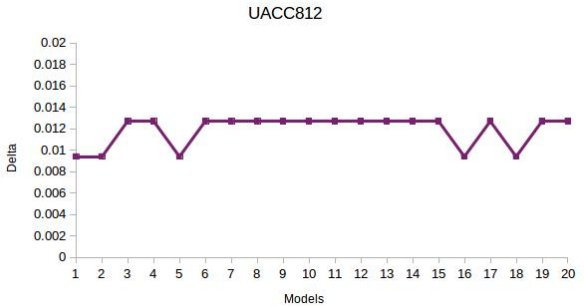
(a) BT20.



(b) BT549.



(c) MCF7.



(d) UACC812.

Figure 4.14 – RMSE for a family of BNs w.r.t testing data, where x-axis shows the true positive Boolean models and y-axis shows the delta. The delta captures the difference between discrete and model RMSE (see Section 4.2.6). Recall that we obtained different number of true positive BNs in each cell line (see Section 4.5.4).

Validation using random data samples

The objective of the following analysis is to show that the BNs obtained with *caspo-ts* using the HPN-DREAM datasets for the four cell lines have a worse RMSE score with respect to random trajectories, and therefore are very specific to the HPN-DREAM datasets. For this purpose, we generated 100 random data samples per cell line. In each sample, we generated a random value in $[0; 1]$ for each readout protein in each time point without changing the perturbations. Then, we calculated the RMSE of these samples with respect to the inferred BNs of each cell line, and finally compared it with the learning and testing RMSE of these BNs. Figure 4.15 plots the RMSE ratio (see Equation (4.5)) of the inferred BNs with respect to the learning, testing and random data.

$$RMSE\ ratio = \frac{Discrete\ RMSE}{Model\ RMSE}. \tag{4.5}$$

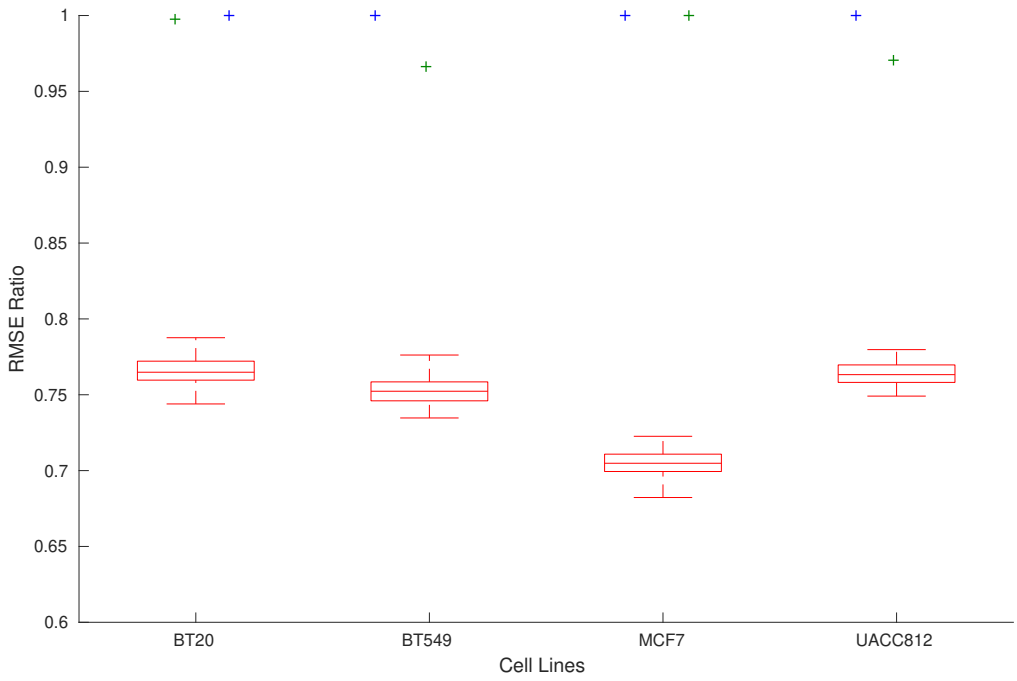
In Figure 4.15, the RMSE ratio for random datasets is displayed by red boxplots for each cell line, and the RMSE ratio for testing and learning datasets is shown as clear outliers in green and blue colors respectively. It is worth noting that the *caspo-ts* method has failed to recover random data time series, hence proving the specificity of the learned networks with respect to the HPN-DREAM challenge dataset.

Validation by introducing noise in the learning data

Additionally, we computed the RMSE of the testing data by using a *leave one out* approach. The analysis is performed for the cell line BT549. Recall that checking the exact reachability of the BNs with respect to a binarized time series trace is a computationally expensive task. The time to model check BNs differs from one network to another. For this case study, the model checking time varied from minutes to days. Because of this reason, we have done this analysis for only one cell line.

We generated 18 modified learning datasets by randomly introducing 5% of noisy data in each dataset; that is, the 5% of the total time point values were generated randomly. We used these datasets to learn 231 ASP-optimal BNs. Then we passed these BNs to the model checker to detect true positive BNs. The model checker verified the exact reachability of 4158 BNs contained in 18 modified datasets. It detected 582 true positive BNs. To verify if these TP BNs can reproduce the time series traces existing in the testing dataset of HPN-DREAM challenge, we used the Equation 4.1 to calculate the RMSE score (see Section 4.2.4). Our results (Table 4.4) show that we were able to obtain an RMSE of 0.3113 for BNs learned from 5% modified learning datasets. It also shows, that most of the ASP-optimal BNs were false positive BNs. Interestingly, the Delta RMSE, which measures how well *caspo-ts* can recover the discrete time-series data, points to regions where the randomization of the learning data

Figure 4.15 – Performance assessment with learning, testing and random datasets. The x-axis shows the cell line and the y-axis shows the RMSE ratio (see Equation (4.5)) of the inferred BNs from the HPN-DREAM data for each cell line with respect to the three datasets. The three datasets are encoded by different color codes. The RMSE ratio with respect to the HPN-DREAM learning and testing datasets is shown in blue and green colors, respectively. The random dataset RMSE ratio distribution is shown as red boxplots.



was different and therefore the ASP-optimal BNs did not recover the full discretized time series data. Overall these results demonstrate that our learning process is robust and is able to handle noisy time series data.

Table 4.4 – Computation summary. This table summarizes the analysis performed on the BT549 cell line. The # of solutions column shows the number of ASP optimal BNs. The true and false positive columns show the number of true positive and false positive BNs returned by the model checker. The Delta RMSE column shows the difference between the discrete and model RMSE. The ASP solving column represents the time spent on learning ASP optimal BNs for each sample. The model checking column shows the time spent on verifying the BNs for each sample. The last column RMSE represents the corresponding RMSE of each sample with respect to the testing data. The ASP solving was performed on a standard laptop machine. The model checking task was performed on a server with 1.5 Tb of RAM.

BT549	**# of Solutions**	**Learning Data**			**Time**		**Testing Data**
Sample #		True Positives	False Positives	Delta RMSE	ASP solving	Model Checking	RMSE
1	231	231	0	0	124 seconds	18 days	0.3113
2	231	231	0	0.0009	125 seconds	15 days	0.3113
3	231	0	231	0.0002	122 seconds	3 days	0.3113
4	231	0	231	0.0004	112 seconds	10 days	0.3113
5	231	0	231	0.0007	101 seconds	7 days	0.3113
6	231	0	231	0	110 seconds	9 days	0.3113
7	231	0	231	0	133 seconds	5 days	0.3113
8	231	0	231	0	133 seconds	6 days	0.3113
9	231	0	231	0.0003	134 seconds	6 days	0.3113
10	231	0	231	0	136 seconds	18 days	0.3113
11	231	0	231	0.0001	90 seconds	22 days	0.3113
12	231	0	231	0.0029	148 seconds	30 days	0.3113
13	231	0	231	0	104 seconds	30 days	0.3113
14	231	0	231	0.0003	91 seconds	23 days	0.3115
15	231	0	231	0.0001	329 seconds	22 days	0.3113
16	231	120	111	0.0018	100 seconds	25 days	0.3113
17	231	0	231	0	150 seconds	30 days	0.3113
18	231	0	231	0.0004	131 seconds	30 days	0.3115

Validation using MTOR canonical pathway

To perform the validation of the structure of the BNs, we calculated a set of *standard nodes* from our PKN which are downstream nodes of MTOR and belong to the canonical MTOR pathway. We then evaluated how many of these standard nodes are also downstream nodes of MTOR in the learned BNs. In the following, the set of downstream nodes of MTOR in the learned BNs is referred to as *inferred set*. The *inferred set* is specific to each cell line. True positive rate (TPR) and false positive rate (FPR) are defined by Equation (4.6) and Equation (4.7) respectively:

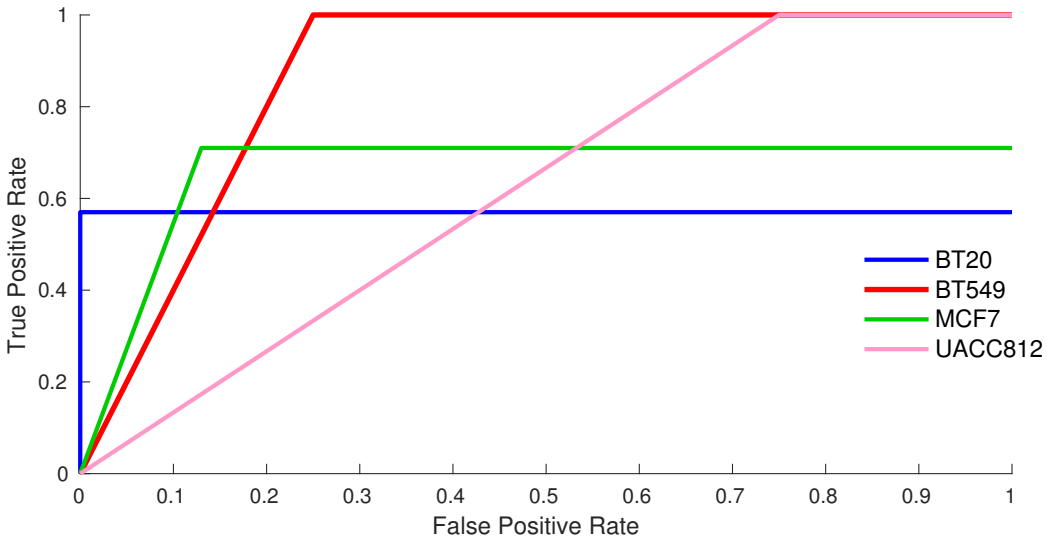
$$TPR = \frac{TP}{TP + FN}, \tag{4.6}$$

$$FPR = \frac{FP}{FP + TN}. \tag{4.7}$$

Here, TP is the number of nodes in the intersection between standard and inferred sets, FP is the number of nodes in the inferred set but not in the standard set, FN is the number of nodes in the standard set but not in the inferred set and TN is the number of nodes which are not in the standard set nor the inferred set.

Figure 4.16 shows the Receiver Operating Characteristic (ROC) curve of each cell line. For BNs of each cell line, TPR and FPR was calculated using Equation (4.6) and (4.7). BT549 cell line models are the most accurate, followed by MCF7 and BT20. We can observe the clear distinction between true positive and false positive BNs. The BNs inferred by *caspo-ts* have an average AUROC score of 0.77 which is comparable to the AUROC score of 0.78 of the top performing method of HPN-DREAM challenge. A number of assumptions made during the modeling phase may have influenced our ranking. First, since our method can pinpoint the noisy, incomplete and erroneous experiment, it allows us to use only the reliable experimental settings. Second, our method constrains the solution space to the proteins existing in the PKN, and so anything outside the prior knowledge cannot be found. From Figure 4.16, we can see that the *caspo-ts* method shows promising results for the inferred true positive BNs.

Figure 4.16 – ROC curve across all cell lines. The x-axis shows the false positive rate and the y-axis denotes the true positive rate. The average AUROC score is 0.77.



4.6 Discussion

In this chapter, we built cell line specific signaling networks for the DREAM time series dataset of 4 breast cancer cell lines (BT20, BT549, MCF7, and UACC812) using *caspo-ts*.

This method combines Answer Set Programming and Model Checking techniques to infer true positive BNs verifying the experimental data. *Caspo-ts* allowed us to handle a midscale PKN (64 nodes and 178 edges) and a real dataset subject to experimental error. *Caspo-ts* enabled us to learn key dynamic mechanisms within the BNs explaining the time series data. Our results suggest that the behavior of cell line specific signaling networks is highly variable even under the same perturbations, agreeing with the heterogeneity of breast cancer and specifically with previous analysis on this data [HNJC$^+$17]. The inferred Boolean models of each cell line were analyzed to identify commonalities as well as discrepancies. Moreover, these inferred models can be executed computationally to identify potential drug targets or to see the effect of unseen perturbations. The predictive power of these models can be increased with improvements in protein interaction databases and comprehensive experimental data.

We have discovered 38% of the cell line dependent behaviors as compared to the 33% of the HPN-DREAM challenge winner [Car14]. We have implemented an algorithm to analyze the variability among cell lines and observed pairwise similarities among these cell lines. The similarity index varies from 15% (BT20 $\&$ BT549) to 23% (MCF7 $\&$ BT549, BT20 $\&$ MCF7). We have analyzed the similarity among the family of BNs of the same cell line as well, which varies from 43% to 73%. We have evaluated the accuracy of our method with RMSE and AUROC scores. The average RMSE of the inferred BNs was 0.31 placing *caspo-ts* in first place in comparison with the top scores reported in the HPN-DREAM challenge. Various choices made during this study may have an impact on the final score. The *caspo-ts* method allowed us to remove noisy and faulty experiments, leaving us with the reliable experimental settings only. Here, we made the choice to use only the reliable experiments of the learning dataset instead of using all experimental settings. Also, we did not observe all 45 proteins as we could not find connections in our PKN for all the studied proteins, leaving us with approximately 23 proteins for each cell line.

Nonetheless, the obtained results are quite promising, making *caspo-ts* a good candidate computational method for learning models given time series datasets and a prior knowledge network. In addition, *caspo-ts* can be used to pinpoint the errors in the experimental data. In particular, we discovered four experiments where the protein *AKT* was inhibited and had a dynamic behavior as a readout protein. Our work therefore provides a novel approach to show erroneous experiments, which is crucial and complementary to current approaches. Finally, the HPN-DREAM dataset contained some noisy readings of experiments. Noisy experimental data reduces the efficiency of computational methods by increasing the variability among constructed Boolean models. To overcome this, we suggest to build automated methods to filter out the noisy experiments. This approach provides a step forward in building cell line dependent networks in the case of phosphoproteomic data.

4.7 Conclusion

We have applied the caspo time series (*caspo-ts*) approach, which is a combination of logic programming and model checking, over the time series phosphoproteomic dataset of the HPN-DREAM challenge to learn cell specific BNs. The learned BNs can be used to identify the cell specific topology. Our analysis suggests that *caspo-ts* scales to real datasets, outputting networks that are not random with a lower fitness error than the models used by the 178 methods which participated in the HPN-DREAM challenge. On the biological side, we identified the cell specific and common mechanisms (logical gates) of the cell lines.

Since *caspo-ts* uses an ASP solver to enumerate BNs, in the resulting sequence of solutions similar BNs are typically clustered together. This can be problematic for large scale problems where we cannot explore the whole solution space in reasonable time. In the next chapter, we discuss how we resolved this issue by sampling to randomly select BNs from the solution space, which allows for shuffling the order in which solutions are enumerated [RSW16]. We implemented this by dynamically modifying the heuristic of the ASP solver at execution time. We show an application of *caspo-ts* with diversification on the UACC812 breast cancer cell line and TCR signaling network. We obtained TP BNs for UACC812 cell line using extended *caspo-ts*.

5

Computing Diverse Boolean Networks from Phosphoproteomic Time Series Data

5.1 Introduction

In this chapter, we introduce the new enumeration scheme for the caspo time series (*caspo-ts*) method. The article related to this work is published in the International Conference on Computational Methods in Systems Biology [RKR+18].

The *caspo-ts* system uses an over-approximation criteria to learn candidate BNs, which can lead to some false positive (FP) BNs. These BNs are not guaranteed to reproduce all traces of the time series data. To resolve this issue, it uses model checking to filter out FP BNs, by checking the exact reachability of these BNs w.r.t. time series data. The *caspo-ts* method uses the *clingo* ASP solver [GKKS14], which is able to exhaustively enumerate all solutions. The *clingo* solver by default uses an enumeration scheme in which, once a solution is found, it backtracks to the first point from where the next solution can be found. This typically leads to the situation where successive solutions (BNs) only change in a small part. As a result, *caspo-ts* may enter a solution space where FP BNs are clustered together. Given the size of the PKN and the small number of perturbations in the experimental data, the solution space can be very large containing billions of BNs making it difficult to enumerate true positive (TP) BNs in reasonable time if it gets stuck in a cluster of FP BNs. We extend *caspo-ts* with a new enumeration scheme for breaking up clusters of similar solutions. In the following, we refer to the modified *caspo-ts* as *caspo-ts*^{D}, where "D" is used to represent diversification. We apply

both systems to two datasets: (1) an artificial dataset for modeling TCR signaling networks, and (2) the HPN-DREAM challenge dataset. Our results show substantial improvements of *caspo-ts*^{D} in solution quality by discovering more signaling behaviors than *caspo-ts*. Moreover, *caspo-ts*^{D} is able to find solutions in cases where *caspo-ts* is unable to find any in a reasonable time period. We also present an algorithm to improve the computational time of the model checking process. The results suggest huge computational time improvements in the model checking process.

5.2 Materials and methods

In this section, we describe the datasets, the *caspo-ts* system, the new algorithm to enumerate diverse BNs implemented in *caspo-ts*^{D}, and the new algorithm to improve the model checking process. Note that we are using the phosphoproteomics time series dataset which has been described in details in Chapter 4, Section 4.2.2. We are using two types of datasets: an artificial (TCR Signaling) and a real dataset (HPN-DREAM).

5.2.1 Artificial dataset

The PKN was derived from the TCR signaling model of [KSRL$^+$06] and consists of 16 nodes and 25 edges. The artificial dataset for TCR signaling was generated in [OPS$^+$16] by simulating the PKN using logic based ODEs. This dataset consists of 4 readouts, 3 stimuli and 2 inhibitors. The readout proteins were measured at 16 time points under 10 perturbations.

5.2.2 HPN-DREAM

The HPN-DREAM dataset is described in the Chapter 4, Section 4.5. The PKN (associated to HPN-DREAM) consists of 64 nodes (7 stimuli, 3 inhibitors, and 23 readouts) and 178 edges (see Figure 4.6 described in detail in Chapter 4, Section 4.5.3).

5.2.3 The *caspo-ts* system

The *caspo-ts* system is based on a combination of ASP and model checking. The ASP part of the *caspo-ts* system is used to solve the combinatorial optimization problem of finding BNs compatible with a PKN and time series data. All learned BNs are optimized using an objective function, minimizing the distance between the original and the time series data determined by the BN learned with *caspo-ts*. The ASP solver guarantees finding all optimal solutions w.r.t. an objective function. The model checking part of the system detects TP BNs by checking the reachability of time series traces given compatible BNs generated by the ASP part of the

system. TP BNs are guaranteed to reproduce all the (binarized) traces under all perturbations by verifying exact reachability in the BN state graph. Since checking this reachability is a PSPACE-hard problem, the second step can be very time consuming for large BNs. To resolve this issue, the ASP part over-approximates solutions [OPS+16]. This over-approximation removes a large set of BNs that have no reachable traces, reducing the number of calls to the model checker.

Optimization

The *caspo-ts* method optimizes the distance of over-approximated traces of BNs w.r.t. the time series traces of the phosphoproteomic dataset. For this purpose, it uses Equation 4.1 described in detail in the Chapter 4, Section 4.2.4. From this optimization step, *caspo-ts* finds a bound, which is an optimal distance that can be achieved from traces of BNs to the traces of experimental data. Then, a constraint is added that only admits solutions that satisfy this bound. We can refer to this solution space as *optimal solution space*.

Then, different parameters of the ASP solver are configured to generate subset-optimal solutions. This means that enumerating subset-optimal solutions is a projection operation over the optimal solution space. We refer to this solution space as the *subset-optimal solution space*.

Notice that the optimal solutions space is generated by writing a logic program, while the subset-optimal solution space is the projection of optimal solution space by setting different parameters of the ASP solver.

Subset minimal BNs. The BNs learned by *caspo-ts* are represented by Boolean formulas in Disjunctive Normal Form (DNF), i.e., as a disjunction of conjunctive clauses [1]. The BNs inferred by *caspo-ts* use the smallest DNF formulas possible, in the sense that no conjunctive clause can be removed from a DNF formula without changing the Boolean function it represents. We refer to these BNs as subset minimal BNs.

Sampling of solution space of BNs. Sampling techniques are used to enumerate diverse solutions from the solution space [EEEF09]. Existing sampling techniques [ZAUH16, RSW16] can be applied to the optimal solution space. In our case, we have to sample the subset-optimal solution space of the *caspo-ts* system. We discovered that existing sampling techniques can only be used with standard ASP solving, which means for the optimal solution space but not for the subset-optimal solution space. This is the main technical difficulty, which hindered our ability to enumerate diverse subset minimal solution for the *caspo-ts* system, as sampling

1. A clause can be seen as a reaction, where the proteins represented positively are active, and the proteins represented negatively are absent. A Boolean formula in DNF encompasses all possible reactions to update the value of a protein.

cannot be implemented on top of subset minimization in the current implementation of the ASP solver.

To resolve this issue, we enumerate subset minimal solutions using heuristics instead of configuring the ASP solver's parameters. Finally, a heuristic based approach is used to enumerate diverse optimal subset minimal solutions. This process of enumeration is explained in Algorithm 4.

In the following, we give a brief description of ASP solving algorithms used by *caspo-ts*. But first let us have a look at an example Boolean formula.

Example 5.2.1. To use Boolean formulas we discretize the phosphoproteomic data: values greater or equal to 0.5 are set to 1, and others are set to 0. Let protein A be associated to Boolean variable A have the formula $(B) \vee (\neg B \wedge C)$ containing the two conjunctive clauses (B) and $(\neg B \wedge C)$, where B and C are Boolean variables representing proteins B and C. This formula can be used to update the value of A. If the update is applied, A is set to 1 if either the value of B is 1, or the value of B is 0 and the value of C is 1. Otherwise, the value of A is set to 0.

ASP solving

Backtracking algorithms are used to solve computational problems by finding (all or some) candidate solutions to a problem. Davis Putnam Logemann Loveland (DPLL) [DP60, DLL62] is a backtracking algorithm which is used to solve the Boolean satisfiability problem (SAT) [BHvM09]. Given a Boolean formula, the goal is to find an interpretation assigning truth values to literals satisfying the formula. DPLL starts with arbitrarily choosing a literal and then assigning a truth value to it. If this assignment does not lead to a conflict then chose another literal and assign a truth value to it. In case of conflict, it backtracks to the last choice. ASP solvers use the conflict driven clause learning algorithm (CDCL), which is a variant of the DPLL backtracking algorithm (Figure 5.1). The main difference difference between CDCL and DPLL lies in the backtracking step. In the CDCL algorithm, when a conflict arises then instead of going just one step back it tries to analyze it and go back to the origin of the conflict (as shown in Figure 5.1).

Next, we explain in detail how the ASP solver *clingo* used by *caspo-ts* discovers solutions (BNs) using the CDCL algorithm [GKKS14], shown in Algorithm 3.

Example 5.2.2. Before continuing any further, we build up our running example. Our running

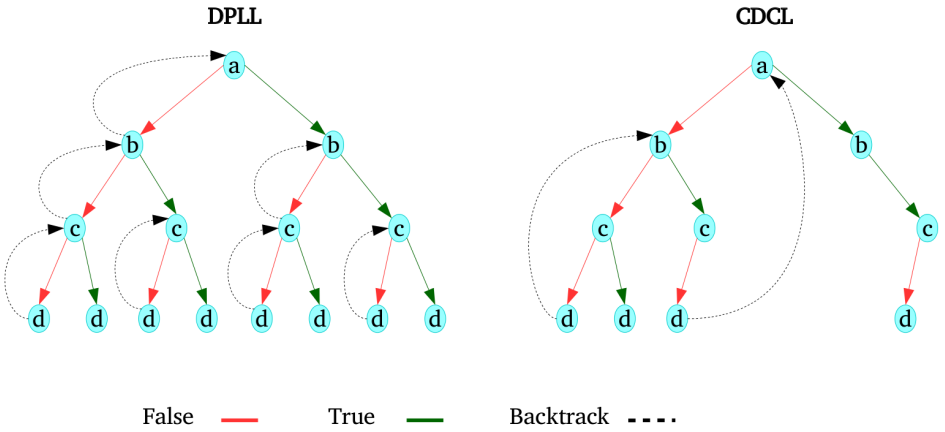


Figure 5.1 – Difference between CDCL and DPLL algorithms. The dashed arrows represent back-jumps. The red and green arrows denote false and true assignment.

example is composed of four constraints:

$$\{a, b, c\} \leftarrow \tag{5.1}$$

$$\leftarrow b \wedge \neg a \wedge \neg c \tag{5.2}$$

$$\leftarrow \neg b \wedge c \tag{5.3}$$

$$\leftarrow \neg b \wedge \neg c \tag{5.4}$$

The first constraint (choice rule) states that solutions consist of all the subsets of the set $\{a, b, c\}$. The second constraint discards all solutions where b is true, a is false, and c is false. The third discards those where b is false and c is true, and the fourth discards those where both b and c are false.

```
   Input: program P
1  Initialize assignment;
2  while assignment is partial do
3      Decide;
4      Propagate;
5      if propagation let to a conflict then
6          Analyze;
7          if conflict can be resolved then
8              Backjump;
9          else
10             return unsatisfiable;
11 return solution given by assignment;
```

Algorithm 3: Conflict-driven clause learning.

The algorithm works by extending a Boolean assignment over the atoms occurring in the

given logic program P until a solution is found. The assignment is initialized in line 1. Then it is extended by the decision heuristic and propagation in the loop in lines 2–10. The call to `Decide()` in line 3 at the beginning of the loop uses a heuristic to select an atom, makes it either true or false, and adds it to the assignment. The consequences of this decision are then propagated in the following line extending the assignment accordingly. Then it is checked if propagation leads to a conflict. If this is the case, the conflict is analyzed in line 6 and the assignment adjusted in line 8 accordingly. Note that a call to `Backjump()` takes back one or more decisions together with their consequences, and adds an additional consequence to the assignment. This property ensures that the algorithm always terminates. It can also happen that a conflict cannot be recovered from. In this case, the problem is found unsatisfiable and the algorithm returns in line 10. Once the assignment is complete, the corresponding solution (set of true atoms) is returned in line 11.

Example 5.2.3. We can now apply Algorithm (3) to our running example (c.f. Example 5.2.2). Starting with an empty assignment, we set a to false as the first decision. There are no immediate consequences and, hence, no conflict can arise. Then we decide to make b false. The consequences of this decision are that c is false via rule (5.3), and c is true via rule (5.4). Hence, we get a conflict, which is resolved and followed by a backjump. Since the conflict was caused by deciding a truth value for b, but is independent of the decision for a, the algorithm takes back all decisions and adds b as a consequence (we now know it must be true in all solutions). We can then decide to make a false again, which sets c to true via rule (5.2). This decision does not cause a conflict and the assignment is no longer partial. Hence, the algorithm terminates with solution $\{b, c\}$.

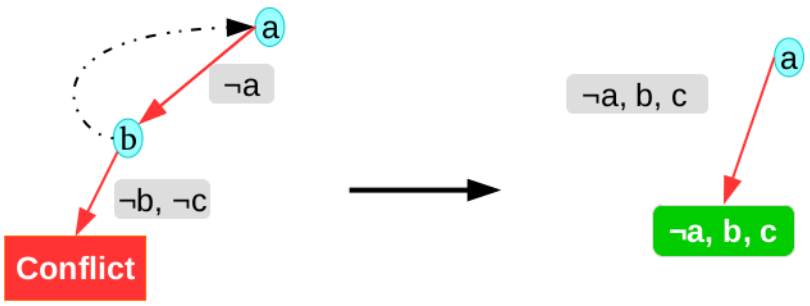


Figure 5.2 – Conflict driven clause learning for example 5.2.3. The red arrows denote false assignment. The dashed arrow represents the back-jump. The gray box shows the assignment of truth values to literals including assignments from decisions and consequences. The green box represents the solution obtained.

5.2.4 Improvements in *caspo-ts*

Modifications in the ASP solver: *caspo-ts*^{D}

Here, we describe the algorithm used for enumerating diverse subset minimal solutions. In *caspo-ts*, the algorithm is implemented in the Python programming language using *clingo*'s multi-shot solving API [KSW17]. The API allows us to customize the solving process, in particular, it allows us to customize the decision heuristic of the solving component, which is the key feature to find subset minimal answer sets. Note that we implemented the algorithm using the multi-shot solving API of *clingo* version 5. For that, we have upgraded the solver *clingo* of *caspo-ts* from 4.5.4 to 5.

Input: program P and atoms T to subset minimize
1 `Prepare(`P`);`
2 **foreach** $x \in T$ **do**
3 `SetSign(`$x, \mathit{false}, 1$`)`
4 **while** *satisfiable* **do**
5 $S \leftarrow$ `Solve();`
6 `AddConstraint(`$\leftarrow a_0, \ldots, a_n$ *for* $\{a_0, \ldots, a_n\} = T \cap S$`);`
7 **foreach** $x \in T \cap S$ **do**
8 `SetSign(`$x, \mathit{false}, 2$`);`
9 **foreach** $x \in T \setminus S$ **do**
10 `SetSign(`$x, \mathit{false}, 1$`);`
11 **if** S *is a true positive* **then**
12 `Output(`S`);`

Algorithm 4: Diverse subset minimal solution enumeration.

Algorithm 4 is used to enumerate diverse optimal subset minimal answer sets. The idea is to configure the decision heuristic in lines 8 and 10, so that it first makes all atoms subject to subset minimization false before deciding truth values for other atoms. This decision heuristic is used in function `Decide()` in Algorithm 3 line 3. This modification ensures that the solution obtained from Algorithm 3 by calling `Solve()` from Algorithm 4 is a subset minimal solution (see [BDRS15] for more details). Such a solution is output in line 12 of the algorithm. Furthermore, the algorithm calls `Solve()` in line 5 multiple times to find *all* subset minimal solutions. To not enumerate solutions twice, a constraint preventing to find the same solution again is added to the logic program P in the following line. This constraint is violated whenever a superset of the atoms in the previously found solution is true. This process is repeated in the loop in lines 4–12 until the program is no longer satisfiable and, hence, all solutions have been enumerated.

So far we only discussed how to enumerate subset minimal solutions. Now we explain how to extend the method in order to compute diverse solutions. The key idea is to make the next

solution different from the previous one. For this purpose, we assign atoms appearing in the last solution to false before assigning any other atoms. To modify the heuristic, we use function `SetSign(`a, t, l`)`, which instructs the decision heuristic to assign atom a to truth value t on level l. The decision heuristic assigns free atoms with the highest level to the designated truth values before assigning atoms on lower levels. By default all atoms have level 0 and the decision heuristic is free to make them either $true$ or $false$. The loop in lines 7–8 instructs the decision heuristic to assign atoms that appeared in the last solution to false on level 2. Any other atoms subject to subset minimization are assigned to false on level 1 in lines 9–10. We see in the experiments in the next section that this strategy breaks up clusters of similar solutions in the solution sequence.

Example 5.2.4. We continue with Example 5.2.2. Let us assume that a, b, and c are the atoms subject to subset minimization. Note that in Example 5.2.3 all decisions assigned atoms to false, so the first solution $\{b, c\}$ obtained is in fact a subset minimal solution. Let us further assume that Algorithm 4 produced this solution in the first iteration (line 5). First, the constraint $\leftarrow b \wedge c$ preventing any superset of $\{b, c\}$ as solution is added in line 6. Then, the decision heuristic is configured to set atoms b and c to false on level 2 (line 8) and atom a to false on level 1 (line 10). In the next iteration, the only possible decision is to set c to false because b is already irrevocably assigned. The consequence of this decision is to set a to true via rule (5.2). Hence, we obtain solution $\{a, b\}$, which is a subset minimal solution. This is followed by adding the constraint $\leftarrow a \wedge b$ in line 6, which in turn makes the program unsatisfiable and causes the loop in lines 4–12 to terminate. We correctly obtain the subset minimal solutions $\{b, c\}$ and $\{a, b\}$. Solution $\{a, b, c\}$ is not subset minimal and not enumerated.

Proofs for Subset Minimal Solution Enumeration

Lemma 1. In line 4, Algorithm 4 computes a subset minimal w.r.t. a set of atoms S and the currently active program.

Proof. We only change the order and truth value with which the decision heuristic of the solving algorithm from [GKS12] assigns atoms. With this modification, the algorithm is still guaranteed to find a solution if there is any. All that remains to show is that the solution found is subset minimal.

When clingo finds a stable model of a program P, it has successively assigned truth values from $\{$*true*,*false*$\}$ to all atoms. This sequence of assignments has the following property: **(P)** Each assignment is either a decision or a result of propagation. Let a be any atom that is the result of propagation. Then the assignment of a is caused by the set of decisions D assigned before a. Every stable model of P that satisfies the literals in D also assigns a to the same truth value.

Now consider that we want to subset minimize the set of atoms S and we use the heuristic method as in Algorithm 4 to obtain a stable model M. As long as there are unassigned atoms in S, the decision heuristic assigns them to false, so every atom $a \in S$ that is true in M is a result of propagation, and all decisions before a are atoms from S assigned to false. Hence, using property **(P)**, we have that every stable model that satisfies the atoms from S assigned to false in M also satisfies the atoms from S assigned to true. Then, there can be no stable model that is a subset of M w.r.t. S. □

Lemma 2. Let P be a logic program and M be a subset minimal stable model of P w.r.t. a set of atoms S. Let the intersection of M and S be the set $\{a_0, ..., a_n\}$, and let C be the constraint $\leftarrow a_0 \wedge \cdots \wedge a_n$. Then, the subset minimal stable models of P are exactly M and the subset minimal stable models of $P \cup \{C\}$.

Proof. The constraint C eliminates all stable models M' with $\{a_0, \ldots, a_n\} \subseteq M'$, so C eliminates M' if $(M \cap S) \subseteq (M' \cap S)$. This means, the constraint eliminates all stable models that are a superset of M w.r.t. S, and the result follows. □

Theorem 1. Algorithm 4 calculates all subset minimal stable models w.r.t. a set of atoms S.

Proof. By Lemma 1, the first stable model calculated by the algorithm is a subset minimal model. By Lemma 2, also the follow up stable models are subset minimal. To see that all subset minimal stable models are computed, note that by Lemma 2 in each iteration exactly one subset solution is eliminated and that the solving algorithm guarantees to find a solution if there is any. □

Modifications in the model checking

Here, we describe the modifications to reduce the runtime of the model checking process. Recall that a BN is given to a model checker to verify a set of traces under a set of perturbations ($\mathcal{P}$) existing in the experimental data. These traces are specified using a combination of existential (E) and eventually (F) operators (as explained in Chapter 4, Section 4.2.5). The model checking component of the *caspo-ts* system takes a BN and verifies these traces. Model checking itself is a PSPACE-hard problem. Giving traces as one big specification to the model checker can result in large overhead in computational time. Because in practice, a model checker performs better when solving independent problem separately. In order to resolve this issue, we propose to split the big specification in smaller parts and verify the traces in a parallel way where as soon as the traces under one perturbation are not reachable, we stop the model checking process and reject the BN as a false positive. The following Algorithm 5 explains the proposed model checking process.

Input: A BN and a set of traces under n perturbations
Start n processes verifying reachability for each perturbation in the background;
while $n > 0$ **do**
    wait for one process to finish;
    if *the process failed to verify reachability* **then**
        stop all remaining processes;
        return FP;
    $n \leftarrow n - 1$;
return TP;

Algorithm 5: Parallelized model checking algorithm.

5.3 Results

We discuss the results of applying Algorithm 4 on two different datasets. We start with the artificial benchmark, where the solution space is small enough (68338) to compute all solutions. This allows us to study this benchmark in more detail. We can analyze how well a limited number of solutions enumerated with *caspo-ts* and *caspo-ts*^{D} represents the solution space.

Then we move to the real dataset, where we cannot enumerate all solutions and can only consider a limited number of solutions because the solution space is too large. Nevertheless, we can show improved results with *caspo-ts*^{D} over *caspo-ts* by being able to enumerate more TP BNs and also more diverse BNs.

We also discuss the results of applying Algorithm 5 to the BT549 cell line. For this, we compare the computation time of the *caspo-ts* system based on previous and new implementation of the model checking process. We can show that we spend less time in verifying reachability, especially in case of FP BNs, as we might be able to stop the verifying process as soon as one of the perturbation fails to verify.

5.3.1 Artificial dataset

Here, we use the TCR signaling dataset [KSRL$^+$06] to demonstrate Algorithm 4 by describing two factors: (1) frequency of the clauses, and (2) true positive rate of BNs. The purpose of studying the first factor is to observe how many clauses we discover while learning a limited number of BNs. In this case, we expect to discover more clauses with *caspo-ts*^{D} than with *caspo-ts*. The second factor (the true positive rate) is used to study how TP solutions are distributed in the solution space. With *caspo-ts*^{D}, we expect TP solutions to be distributed much more evenly.

Fig. 5.3 depicts the frequency of clauses in the solutions of the artificial benchmark. Clauses that occurred in at least one solution are depicted on the x axis. Each tick stands for one clause and the label has format $n \leftarrow c$ where c is a clause and n is a node name. Furthermore, clauses

associated with the same node are grouped together by shading the background alternatingly in light gray and white. The frequencies of the clauses are depicted on the y axis. The red line depicts the frequency considering all 68338 solutions, while the green and blue lines depict the frequencies of the first 100 solutions computed by *caspo-ts* and *caspo-ts*^{D}, respectively. In total, there are 49 clauses appearing in the family of BNs. While *caspo-ts*^{D} (blue line) discovered 48 clauses, *caspo-ts* (green line) learned only 29 clauses by enumerating the same number of BNs (100). We also observe that the blue line is often much closer to the red line (with an average distance of 0.06) than the green line is (with an average distance of 0.20). This shows that the diverse enumeration scheme is able to produce solutions that are less similar to each other and better represent the solution space. The underlying ASP solver of *caspo-ts* by default uses an enumeration scheme that backtracks to the first point from which the next solution can be found. This approach typically leads to the situation where successively enumerated solutions only change in a small part. We can observe this behavior in Figure 5.3, where some clauses are overrepresented.

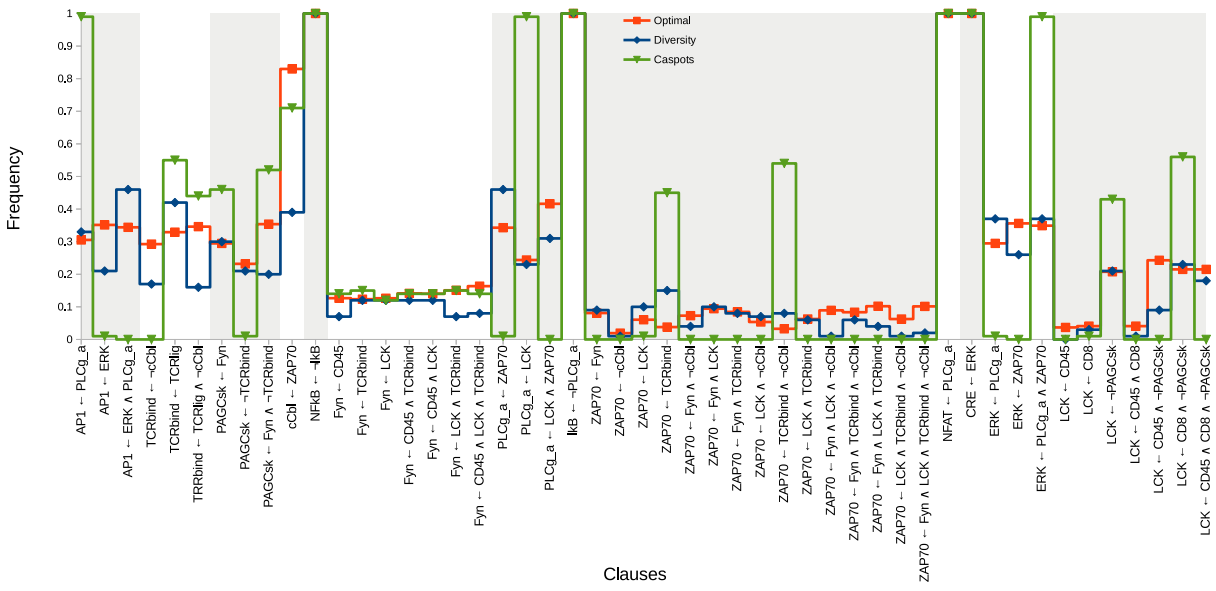


Figure 5.3 – Frequency of clauses per node in all 68338 BNs (red line), and in the first 100 BNs enumerated by *caspo-ts* (green line) and *caspo-ts*^{D} (blue line).

Figure 5.4 depicts the true positive rate of blocks of successive solutions. Each tick on the x axis stands for a block of 1000 solutions. The y axis depicts the percentage of true positives in a block of solutions. The red line depicts the overall true positive rate (78%), while the green and blue lines depict the true positive rates of *caspo-ts* and *caspo-ts*^{D}, respectively [2]. We observe that for the *caspo-ts* system there are a lot of blocks with either a lot of true positives or very few. This suggests that true positives are clustered in the sequence of enumerated solutions. We observe that the diverse enumeration scheme does not show this behavior. This is especially

2. For example, when x has value 3000, the y value in blue gives the true positive rate among the solutions 2001 to 3000 computed by *caspo-ts*^{D}.

important for enumerating true positive solutions of real world instances where only a limited number of solutions can be checked because of time constraints. With the original *caspo-ts* system, it can happen that the first cluster does not contain any true positives, making it impossible to find any true positive solution within a given time budget. The graph also shows that the diverse enumeration scheme does not sample over the full solution space. We see that before around 23000 solutions, the true positive rate is below the ideal 78% and then jumps up afterward. Thus, we conjecture that our enumeration scheme mainly breaks up local clusters of solutions with the artificial benchmark. Still, it is able to discover almost all clauses compared to the whole solution set (the frequency is 0 only once), while *caspo-ts* does not discover 20 clauses at all.

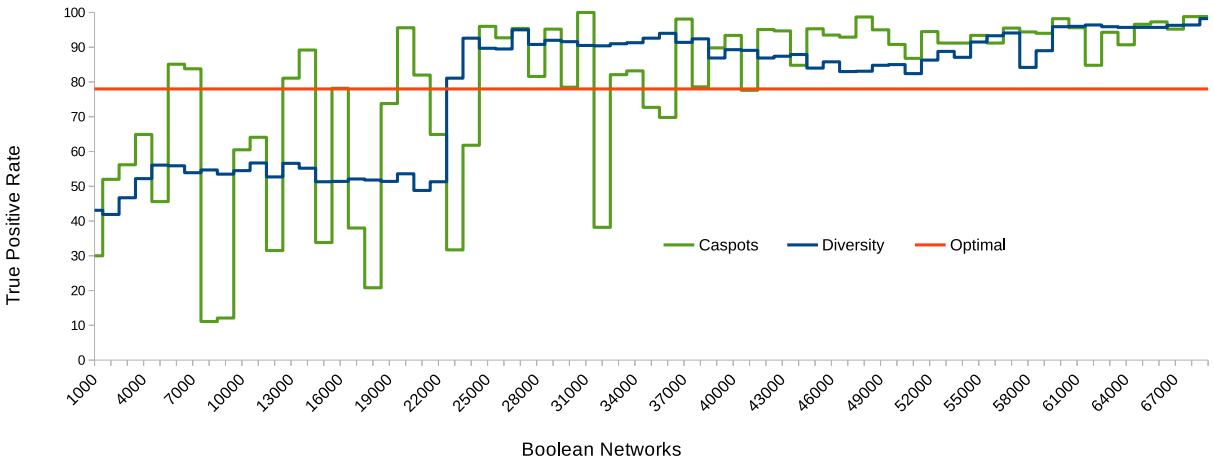


Figure 5.4 – True positive rate of BNs grouped in blocks of 1000 networks.

5.3.2 HPN-DREAM challenge dataset

Next, we show the results of applying diverse solution enumeration to the HPN-DREAM challenge dataset [HHC+16]. We discuss the results according to three aspects: (1) time to compute the first true positive BN, (2) similarity among the family of solutions, and (3) Boolean functions computed by the original *caspo-ts* and the extended *caspo-ts*^{D} system. We start the analysis with four cell lines, and then we provide a detailed analysis of the Boolean functions of one cell line discovered by *caspo-ts* and *caspo-ts*^{D}.

Given that model checking is a computationally hard problem, we stop an experiment after a system verifies (using the model checker) 46 BNs per cell line [3]. The model checking task was performed on a server with 1.5 Tb of RAM. Table 5.1 shows the number of TP BNs obtained for each cell line. We see that the number of TP BNs differs comparing *caspo-ts* and *caspo-ts*^{D}. For MCF7 we obtain 0 TP BNs with *caspo-ts* and 4 TP BNs with *caspo-ts*^{D}, while for BT549

3. Note that the model checker could only verify 32 out of 46 solutions within one month for cell line BT20 in case of *caspo-ts*^{D}. There may exist more TPs for this cell line.

we obtain 2 and 14 TP BNs, respectively. For the other two cell lines BT20 and UACC812, we obtain a comparable number of BNs. We observe that we can get more TP solutions by checking the same number of BNs with *caspo-ts*^{D}. This is an important improvement given the fact that model checking BNs is a computationally hard problem. Next, we consider the time column showing the time to compute the first TP BN for each cell line. We see that we are unable to get TP BN with *caspo-ts* in case of the MCF7 cell line, which shows that the *caspo-ts* system is stuck in a part of the search space where there are only FPs. Otherwise, for the other cell lines the time to get the first TP BN is comparable. We conclude that the difficulty to model check a BN depends on the cell line and not on the order in which solutions are found. Finally, the similarity column shows the similarity score among the set of TP BNs for each cell line. This score is calculated by comparing the clauses of one cell line with each other. We observe that the similarity among the solutions is much higher for *caspo-ts* than for *caspo-ts*^{D}. From this we conclude that, studying the same number of networks, *caspo-ts*^{D} can discover more clauses representing diverse signaling behaviors.

Table 5.1 – Number of TP BNs out of 46 BNs, time to get the first TP solution, and similarity among TP solutions per cell line.

	caspo-ts			***caspo-ts*^{D}**		
Cell Line	**TPs**	**Time**	**Similarity**	**TPs**	**Time**	**Similarity**
MCF7	0	—	—	4	6.7h	0.51
BT549	2	8.4m	0.92	14	7.9m	0.44
UACC812	20	26s	0.81	15	27s	0.45
BT20	13	20h	0.86	7+	20h	0.32

Now, we analyze in more detail the UACC812 cell line using *caspo-ts*. This cell line was the most difficult one as explained in Chapter 4, Section 4.5.4. Figure 5.5 shows the union of 10 TP BNs obtained by *caspo-ts*. There are four different kinds of nodes in the graph: (1) stimuli shown in green, (2) inhibitors shown in red, (3) readouts shown in blue, and (4) unobserved nodes shown in white. Note that blue nodes with red borders are readouts, which are also inhibitors. There are two different kinds of edges shown in red and green color. Green edges are used to show a positive influence ($\leftarrow$), and red edges are used to show a negative influence ($\vdash$). We discovered 25 clauses with *caspo-ts*, and observed that the learned BNs only contain Boolean functions with clauses of size one. We also noticed that the learned BNs are very similar to each other, as we see in Table 5.1 with the similarity score of 0.81. This relates to the fact that the ASP solver used by *caspo-ts* uses a backtracking algorithm to enumerate solutions and, hence, the solutions only change in small parts.

Next, we analyze the UACC812 cell line using *caspo-ts*^{D}. Figure 5.6 shows the union of 10 TP BNs obtained by *caspo-ts*^{D}. Nodes, edges and colors have the same meaning as in Figure 5.5. Unlike with *caspo-ts*, here we identified clauses with more than one element. They

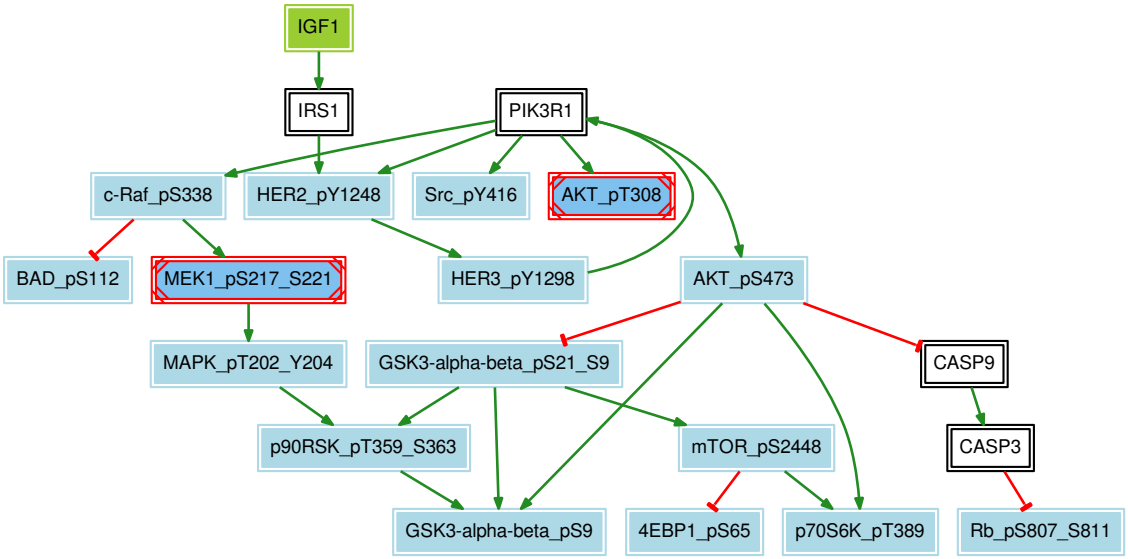


Figure 5.5 – *caspo-ts*: 10 optimal TPs BNs concatenated for cell line UACC812. All BNs are identically optimal.

are represented by black rectangles where the nodes of incoming edges are their elements. Additionally, we use dashed edges to represent clauses that were also discovered by *caspo-ts*. In total, *caspo-ts*^{D} discovered 66 clauses, 41 more than *caspo-ts*. It identified 23 out of the 25 clauses discovered by the *caspo-ts* system, and 43 additional clauses, studying the same number of BNs. It is important to note that even though for the UACC812 cell line *caspo-ts*^{D} learned 5 TP BNs less than *caspo-ts*, the number of clauses learned by *caspo-ts*^{D} is 3 times higher. It is interesting to see that 4 new stimuli appear in the solution set using *caspo-ts*^{D} while there is only one stimulus in the previous solution set discovered by *caspo-ts*. We also discover one additional readout *"JNK_pT183_pT185"* using *caspo-ts*^{D} as compare to *caspo-ts*. Since we find much more clauses with *caspo-ts*^{D}, we can get an impression of the whole solution space by just inspecting a limited number of solutions. This analysis shows the efficacy of the extended *caspo-ts*^{D} system in a real case scenario, where it is difficult to study the complete solution space because of time constraints.

5.3.3 Computation of root mean square error

Recall that the UACC812 cell line was the most difficult one to obtain true positive BNs (see Section 4.5.4). We discussed previously that it was because of the enumeration scheme used by the ASP solver. In this chapter, we have resolved this issue and obtained true positive BNs for this cell line. In Chapter 4 and Section 4.5.7, we presented the RMSE score of BNs of UACC812 cell line. Here we compare that result with the true positive BNs of this cell line in Figure 5.7. For this purpose, we used the 20 false positive and 20 true positive BNs

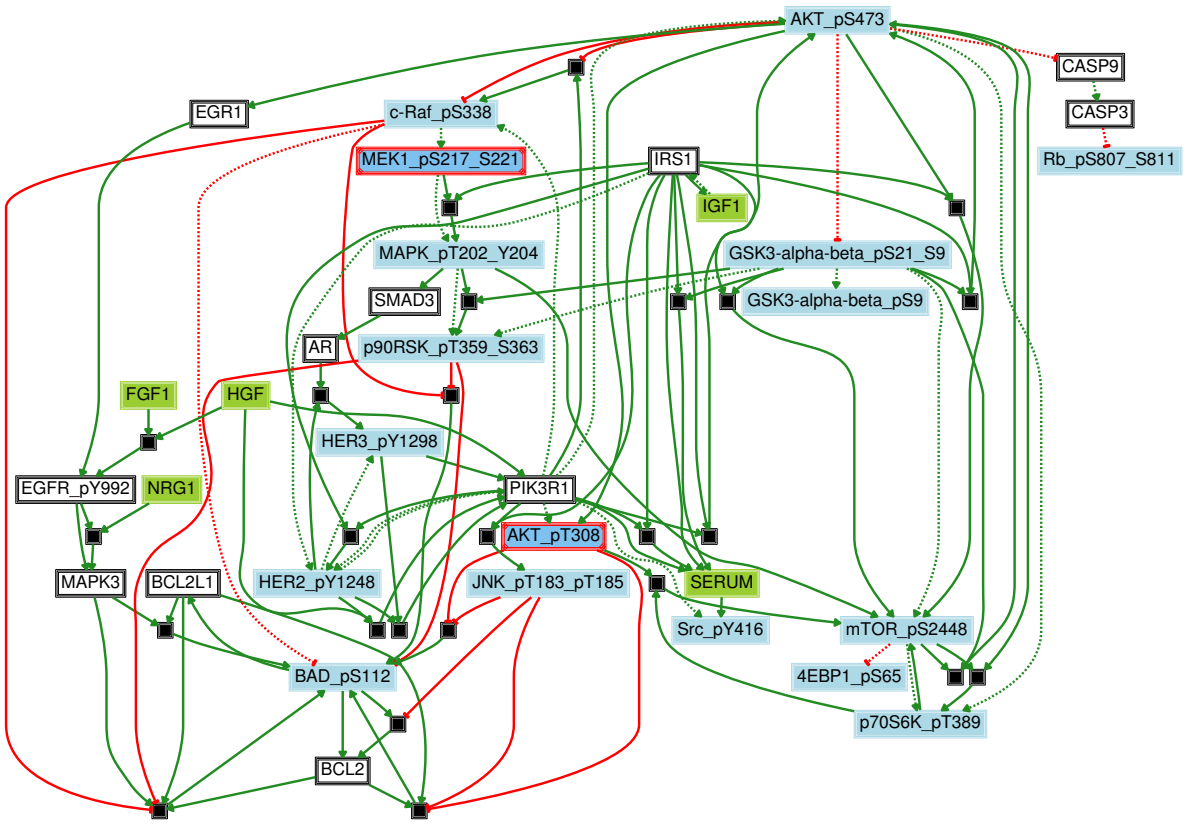


Figure 5.6 – *caspo-ts*^{D}: 10 optimal TP BNs concatenated for cell line UACC812. The dashed edges are used to represent clauses that were also discovered by *caspo-ts*. AND gates are represented by black boxes.

of UACC812 cell line. The average RMSE score for the false positive networks is 0.0119 while the score is 0.0073 for the true positive networks. The minimum and maximum RMSE score for the false positive networks is 0.0094 and 0.0127 respectively. The minimum and maximum RMSE score for the true positive networks is 0.0021 and 0.0114 respectively. These results show that the true positive networks have lower RMSE score and provide the minimum distance of time series of BNs to the phosphoproteomic time series.

5.3.4 Improvements in computation time of a BN's model checking

Here we analyze the computational time spent on verifying the reachability of all time series traces using the previous and new parallel model checking implementation. We created 16 samples of the $BT549$ cell line by slightly modifying the data in it. We learned 231 BNs for each sample and verified the reachability properties in case of previous and parallel model checking process. Table 5.2 illustrates the achieved results. The verified BNs column shows the number of BNs verified by the model checker. The TPs column shows the number of TP BNs. The FPs column represents the number of FP BNs. The time column shows the time spend on checking reachability on the respective sample. The difference column represents the

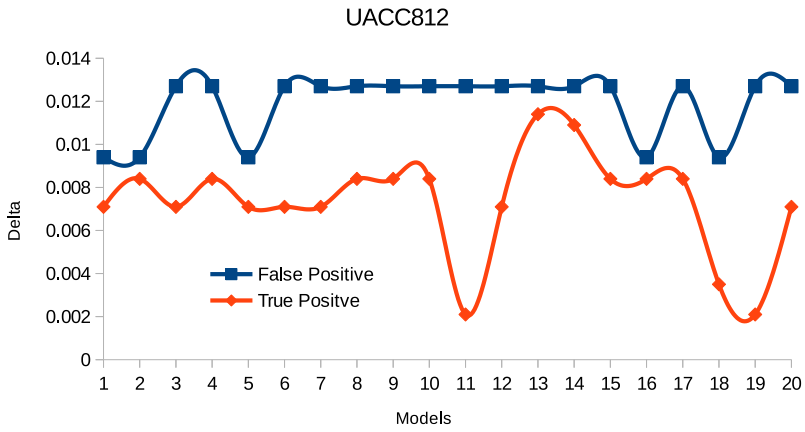


Figure 5.7 – RMSE score for true and false positive networks of UACC812 cell line.

improvement in time by using parallel model checking. As expected we see huge improvement in the computation time of FP BNs, moreover we have also observed improvements in the computational time of TP BNs (sample 12 and 14). In case of sample 7 where the previous model checking process only verified 22 BNs in 90 days, parallel model checking verified all 231 BNs within 6 hours. In case of sample 4 and 13, with the previous model checking approach, it took days to verify all 231 BNs while parallel model checking took only hours. The previous model checking process could not finish verifying all 231 BNs for 7 out of 16 samples while parallel model checking finished verifying all BNs for all 16 samples. The total time spent on verifying 16 samples using the previous model checking approach is more than 1016 days while the total time spent on verifying reachability using parallel model checking is 148 days. This analysis shows the efficiency of the presented algorithm, where it is extremely time consuming (in some cases more than 3 months) to verify the BNs using the previous implementation.

5.4 Discussion

We have presented an algorithm to enumerate diverse optimal solutions with the *caspo-ts* system. The new algorithm extends the approach of [RSW16] for computing diverse optimal solutions. The novelty of this extension is that by modifying the heuristic of the solver we manage to enumerate solutions that are both *optimal* and *diverse*. There are other approaches for computing diverse solutions [EEEF09, HHOW05b, Nad11b] but they do not consider optimization problems.

Our key findings suggest that we retrieved a more complete set of mechanisms explaining the experimental data and better approximate biological reality, by sampling the large solution space of BNs. We were able to discover almost all behaviors (48 out of 49) existing in the

Table 5.2 – Comparison of previous and parallel model checking.

BT549	Previous Model Checking				Parallel Model Checking				Difference
	Verified BNs	TPs	FPs	Time	Verified BNs	TPs	FPs	Time	
1	231	0	231	64d	231	0	231	7.2d	56.8d
2	221	0	221	>90d	231	0	231	15.7d	>74d
3	173	0	173	>90d	231	0	231	11.9d	>78d
4	231	0	231	7.2d	231	0	231	17h	6.5d
5	37	0	37	>90d	231	0	231	20.4d	>70d
6	231	0	231	78.5d	231	0	231	14d	64.5d
7	22	0	22	>90d	231	0	231	6h	>84d
8	172	0	172	>90d	231	0	231	8.5d	>82d
9	231	0	231	76.7d	231	0	231	9.5d	67.2d
10	121	0	121	44d	231	0	231	2.5d	41.5d
11	135	0	135	>90d	231	0	231	7.5d	>82d
12	231	231	0	16.2d	231	231	0	4.1d	12.1d
13	231	0	231	9.3d	231	0	231	10h	8.9d
14	231	231	0	50d	231	231	0	11.8d	38.2d
15	231	0	231	40d	231	0	231	3.8d	36.2d
16	105	0	105	>90d	231	0	231	29.7d	>60d

complete solution space of 68338 BNs of TCR signaling dataset by exploring only 100 BNs using *caspo-ts*^{D}. We also identified TP BNs in a case (MCF7) where *caspo-ts* could not find any TP BN among 46 solutions. Our method is applicable to gene or protein expression time series datasets measured upon different perturbations. Moreover, the proposed Algorithm (4) is not specific to our biological application. It computes diverse subset minimal solutions in ASP, and therefore can be applied to any problem modeled in ASP.

We have also improved the issue of the huge amount of time required for categorizing the BN as TP or FP using model checking. An algorithm is presented to check reachability in a parallel fashion. The substantial improvements have been shown in computation time by demonstrating the huge time difference (868 days) between the previous and the new implementation of the model checking process.

5.5 Conclusion

In this chapter, we have presented an extended version of the *caspo-ts* system to enumerate diverse solutions. We applied *caspo-ts*^{D} on an artificial dataset (TCR signaling) as well as a real case study (HPN-DREAM) to learn diverse BNs. We compared the results with the *caspo-ts* system, showing a substantial improvement in solution quality. For one, we discovered more signaling behaviors (clauses) comparing solutions enumerated with both systems. For another,

we were able to find solutions for cell line MCF7, where *caspo-ts* could not find solutions before. Furthermore, we have presented an algorithm to parallelize the model checking process. The results show substantial improvements in computation time of the model checking process.

6

Conclusions and Future Work

In this chapter, we summarize the work and outline future perspectives.

6.1 Summary

Protein signaling networks are static views of dynamic processes where proteins go through many biochemical modifications such as ubiquitination and phosphorylation to propagate signals that regulate cells and can act as feedback systems. Understanding the precise mechanisms underlying protein interactions can elucidate how signaling and cell cycle progression occur within cells in different diseases such as cancer. In this thesis, we have advanced our understanding in breast cancer by using the *caspo-ts* method to learn BNs from multiple perturbation phosphoproteomic time series data given a Prior Knowledge Network. We have improved and adapted *caspo-ts* to deal with a midscale PKN with 64 nodes and 178 edges in order to learn the BNs of four breast cancer cell lines (BT20, BT549, MCF7, UACC812) from their time series phosphoproteomic datasets. Importantly, the PKN did not contain any information about the temporal changes or dynamic properties of the proteins. This information was learned from a dataset describing the dynamics of signaling processes for those breast cancer cell lines as part of the HPN-DREAM challenge. The initial results highlighted several characteristics of this method when applied on HPN-DREAM dataset (a real case study):

1. high false positive rate,
2. large solution space (billions of BNs),
3. high computation time of *caspo-ts*-model checking component.

The **high false positive rate** of BNs highlighted issues existing in a real dataset, i.e., missing time points and inconsistent or incompatible measurements of proteins (see Chapter 4 Section 4.5.2). It also pointed us to the fact that a limited number of perturbations is not enough to derive a unique model representing one specific cell line, but rather leads billions of solutions (BNs). We also discovered that the shuffling of perturbations leads to the generation of solutions in a different order. By resolving these problems, we managed to generate true positive BNs for the case study of the HPN-DREAM challenge. Our results point to measurements in the time series phosphoproteomic dataset that contradict the experimental setting and to perturbations that show contradictory dynamics. This analysis highlighted the erroneous experiments, so this method can be used as complementary to the existing approaches to discover errors (see Chapter 4).

A detailed analysis of the **large solution space** generated by the *caspo-ts* method led to the discovery of clustered solutions. The *caspo-ts* method uses the *clingo* ASP solver, which is able to exhaustively enumerate all solutions. The *clingo* solver by default uses an enumeration scheme, in which, once a solution is found, it backtracks to the first point from where the next solution can be found. This typically leads to the situation where successive solutions only change in a small part. As a result, *caspo-ts* may enter a solution space where BNs are clustered together. We have observed that given the size of the PKN and the small number of perturbations in the experimental data, the solution space of the *caspo-ts* can be very large containing billions of BNs making it difficult to enumerate true positive BNs in reasonable time if it gets stuck in a cluster of false positive BNs. We have improved the ASP-based *caspo-ts* system to resolve the above mentioned issue. We have extended the *caspo-ts* system using heuristics to enumerate diverse solutions. Our results showed that this extension proved to be very useful to breakup cluster of solutions and to get the better impression of the solution space (see Chapter 5).

The **high computation time** in case of a real case study (HPN-DREAM challenge) directed us to analyze the model checking component of the *caspo-ts* system. Since the *caspo-ts* system uses an over-approximation criteria, it can lead to false positive BNs. These false positive BNs are ruled out using a model checker by verifying a reachability property. We observed that the properties are verified in a manner, which can cost the expensive computation time spent on verification of properties. To resolve this issue, we proposed a new parallel verification of the properties. Our results showed that the parallel verification of properties saves us months of computation time (see Chapter 5).

6.2 Future perspectives

6.2.1 Simulation of Boolean models

It has been shown that the HER2 amplified breast cancer demonstrates significant increase in BCL2 and ER expression when treated with anti HER2 therapy. The co-regulation of ER or BCL2 with anti HER2 drug is shown to be useful for decreasing the BCL2 and ER over-expression in HER2 positive UACC812 cell line. These molecular entities (BCL2 and HER2/3) exist in our Boolean model of UACC812 cell line. In the near future, we plan to execute the Boolean model of UACC812 cell line derived using *caspo-ts*^{D}presented in Chapter 4 (Figure 5.6), to study the impact of inhibiting these molecules in our Boolean models and compare the results with literature. We believe this study will be useful for final validation of our models. We studied some software to simulate boolean models but we faced a lot of trouble configuring different parameters for the toy example. Different software produces different simulation results. We also had problem reproducing the results published with these software. Therefore, we plan to develop our own tool to simulate the Boolean models and embed it in the *caspo-ts*^{D}system .

6.2.2 Extension of diversity algorithm

For the first time, with our study, we observe that solutions are not randomly distributed along the search tree of *caspo-ts*, as shown in Chapter 5. In the near future, we plan to extend the diversity algorithm in two directions. First, we are planning to experiment with solver parameters in order to introduce some randomness into the search. Second, we intend to extend the algorithm to call the model-checker only on answer sets which are diverse (according to some measure to be defined). This is possible because the time to enumerate over-approximated solutions using the ASP solver is much lower than the time needed to check solutions using the model-checker. We expect both enhancements to further improve the diversity of the discovered solutions.

6.2.3 Extension of model checking algorithm

In the current implementation, we parallelize the properties on the perturbation level. It can be improved further by parallelizing the traces within a perturbation as well. This is possible because we can prove in a parallel fashion that one time point is reachable from the preceding time point, too. Typically, it can be quickly determined if a property does not hold but it might take a long time to verify that property holds. A BN is a false positive if one of the reachability property does not hold, and a lot of computation time might be spent before detecting a false positive BN. However by parallelizing verification at time point level, we can stop the model

checker from verifying reachability of other time points and other perturbations as soon as one transition cannot be verified. We believe that this improvement will help in gaining computation time particularly in case of many time points.

6.2.4 Experimental design

Currently, our research is in the context of identifying BNs representing a part of a signaling pathway from real time series of phosphoproteomics data using caspo time series method which is based on answer set programming and model checking. Phosphoproteomics data contain a lot of noise, and also our approach is exhaustive, so it results in learning a family of Boolean Networks. This highlights the need for designing efficient experimental strategy. To deal with this issue, we proposed to: (1) identify a way to characterize different dynamical behaviors among a family of Boolean networks and (2) develop an algorithm based on the maximum entropy measure to select the efficient experimental perturbation to make this family of dynamic behaviors less variable.

7

Scientific Activities

7.1 Publications

Peer reviewed journal

Misbah Razzaq, Loïc Paulevé, Anne Siegel, Julio Saez-Rodriguez, Jérémie Bourdon, and Carito Guziolowski: Computational Discovery of Dynamic Cell Line Specific Boolean Networks from Multiplex Time-Course Data, Plos Computational Biology, Published 2018.

Book chapter

Misbah Razzaq, Lokmane Chebouba, Pierre Le Jeune, Hanen Mhamdi, Carito Guziolowski, and Jérémie Bourdon: Logic and Linear programs to understand cancer response, Automated Reasoning for Systems Biology and Medicine (ARSBM), Accepted 2018.

Conference paper

Misbah Razzaq, Roland Kaminski, Javier Romero, Torsten Schaub, Jérémie Bourdon, and Carito Guziolowski: Computing Diverse Boolean Networks from Phosphoproteomic Time Series Data, 16th International Conference on Computational Methods in System Biology (CMSB), Published 2018.

7.2 Academic visits

1. Short Research Stay, University of Potsdam, Germany, Oct-Dec 2017.
2. Short Research Visit, University of Manchester, UK, 11-13 July 2017.
3. Summer School, Ile de Porquerolles, France, 6-10 June 2016.

7.3 Scientific Communications

7.3.1 Presentations

1. Presentation, Seminar, LRI: Laboratoire de Recherche en Informatique, l'Université Paris-Sud, France, 8 Nov 2018.
2. Presentation, 16th International Conference on Computational Methods in System Biology (CMSB), Brno, Czech Republic, 12-14 Sept 2018.
3. Presentation, ASSTABIO: Apprentissage de modèles Statistiques et Stochastiques A partir de données BIOlogiques, Rennes, France, 22-23 March 2018.
4. Presentaion, 11èmes journées du Cancéropôle Grand Ouest, Vannes, France, 29-30 June 2017.
5. Flash Presentation, La Journée de doctorants (JDOC), l'Ecole doctorale STIM, Nantes, France, May 4 2017 (Best Presentation Award).

7.3.2 Posters

1. Poster, ISMB/ECCB 2017, Prague, Czech Republic July 21 - July 25 2017.
2. Poster, La Journée de doctorants (JDOC), l'Ecole doctorale STIM, Nantes, France, 4 May 2017.
3. Poster, Formal Modeling of Biological Regulatory Networks, Summer School, Ile de Porquerolles, France, June 2016.

7.4 Teaching

Teaching Assistant, Oct 2016 - Sept 2018, Computer Science, Ecole Centrale de Nantes, France.

1. C++, 102 Hours, Oct 2016 - March 2018.
2. Databases, 8 Hours, Jan 2017.
3. Matlab, 12 Hours, Feb-Jun 2018.

7.5 Supervision

7.5.1 Simulator for Boolean models

I have supervised three students to develop a tool in python which can simulate and visualize boolean models. We have managed to develop a first version of the tool. This tool takes a Boolean model as input, simulates it under given experimental settings and visualize the results. We have used the Epidermal Growth Factor (EGF) and Tumor Necrosis Factor Alpha (TNFα) as benchmark [MTH$^+$12]. Our results suggest that the initial condition affect the evolution of the dynamics of the Boolean model.

7.6 Collaborative Research

7.6.1 Interaction graph for dream 11 challenge

I also participated in the Dream 11 challenge focusing on discovering molecular signatures to virus exposures [Sie16]. More precisely, I worked on generating PKNs for the Dream 11 challenge for merging knowledge driven methods with data driven methods. An influence graph is extracted from different literature databases based on the selection of core variables. I inferred six influence graphs from literature curated databases i.e BioCarta, Go Ontology, Kegg, NCI, Reactome, and Wiki Pathways using the enrichnet web tool. In order to cover all core variables existing in dataset, I merged influence graphs to generate a new enriched influence graph. According to the results obtained, there is still a lot of room for improvement in the data driven algorithm to improve overall performance.

Bibliography

[ABLS06] Bree B Aldridge, John M Burke, Douglas A Lauffenburger, and Peter K Sorger. Physicochemical modelling of cell signalling pathways. *Nature cell biology*, 8(11):1195, 2006. 34

[AGW14] Amir Abboud, Fabrizio Grandoni, and Virginia Vassilevska Williams. Subcubic equivalences between graph centrality problems, apsp and diameter. In *Proceedings of the twenty-sixth annual ACM-SIAM symposium on Discrete algorithms*, pages 1681–1697. SIAM, 2014. 77

[Alo06] Uri Alon. *An introduction to systems biology: design principles of biological circuits*. CRC press, 2006. 13, 14

[BB15] Stefanie Boellner and Karl-Friedrich Becker. Reverse phase protein arrays—quantitative assessment of multiple biomarkers in biopsies for clinical use. *Microarrays*, 4(2):98–114, 2015. 28

[BCT+04] Chitta Baral, Karen Chancellor, Nam Tran, NL Tran, Anna Joy, and Michael Berens. A knowledge based approach for representing and reasoning about signaling networks. *Bioinformatics*, 20(suppl_1):i15–i22, 2004. 43

[BDRS15] Gerhard Brewka, James Delgrande, Javier Romero, and Torsten Schaub. Implementing preferences with asprin. In F. Calimeri, G. Ianni, and M. Truszczyński, editors, *Proceedings of the Thirteenth International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR'15)*, volume 9345 of *Lecture Notes in Artificial Intelligence*, pages 158–172. MIT Press, 2015. 93

[BFA+10] Mara A Bonelli, Claudia Fumarola, Roberta R Alfieri, Silvia La Monica, Andrea Cavazzoni, Maricla Galetti, Rita Gatti, Silvana Belletti, Adrian L Harris, Stephen B Fox, et al. Synergistic activity of letrozole and sorafenib on breast cancer cells. *Breast cancer research and treatment*, 124(1):79–88, 2010. 21

[BGN06] Marcello Balduccini, Michael Gelfond, and Monica Nogueira. Answer set based design of knowledge systems. *Annals of Mathematics and Artificial Intelligence*, 47(1-2):183–219, 2006. 43

[BHvM09] Armin Biere, Marijn Heule, and Hans van Maaren. *Handbook of satisfiability*, volume 185. IOS press, 2009. 44, 90

[BL13] Marcello Balduccini and Yuliya Lierler. Integration schemas for constraint answer set programming: a case study. *Theory and Practice of Logic Programming*, 13(4-5), 2013. 45

[Blo09] James Blondin. Particle swarm optimization: A tutorial. *Availaible from: http://cs. armstrong. edu/saad/csci8100/pso tutorial. pdf*, 2009. 34

[BMH+13] Kieran Brennan, Elaine A McSherry, Lance Hudson, Elaine W Kay, Arnold DK Hill, Leonie S Young, and Ann M Hopkins. Junctional adhesion molecule-a is co-expressed with her2 in breast tumors and acts as a novel regulator of her2 protein degradation and signaling. *Oncogene*, 32(22):2799, 2013. 25

[BSE05] Anna Birgersdotter, Rickard Sandberg, and Ingemar Ernberg. Gene expression perturbation in vitro—a growing case for three-dimensional (3d) culture systems. In *Seminars in cancer biology*, volume 15, pages 405–412. Elsevier, 2005. 21

[BSS+92] Christopher C Benz, Gary K Scott, Jay C Sarup, Randolph M Johnson, Debasish Tripathy, Ester Coronado, H Michael Shepard, and C Kent Osborne. Estrogen-dependent, tamoxifen-resistant tumorigenic growth of mcf-7 cells transfected with her2/neu. *Breast cancer research and treatment*, 24(2):85–95, 1992. 23

[C+04] Gene Ontology Consortium et al. The gene ontology (go) database and informatics resource. *Nucleic acids research*, 32(suppl 1):D258–D261, 2004. 28

[Car14] Daniel E Carlin. *Computational evaluation and derivation of biological networks in cancer and stem cells*. University of California, Santa Cruz, 2014. 85

[CCG+02] Alessandro Cimatti, Edmund Clarke, Enrico Giunchiglia, Fausto Giunchiglia, Marco Pistore, Marco Roveri, Roberto Sebastiani, and Armando Tacchella. Nusmv 2: An opensource tool for symbolic model checking. In *International Conference on Computer Aided Verification*, pages 359–364. Springer, 2002. 63

[CEP95] Allan Cheng, Javier Esparza, and Jens Palsberg. Complexity results for 1-safe nets. *Theoretical Computer Science*, 147(1):117 – 136, 1995. 61

[CGL10] Kathryn J Chavez, Sireesha V Garimella, and Stanley Lipkowitz. Triple negative breast cancer cell lines: one tool in the search for better treatment of triple negative breast cancer. *Breast disease*, 32(1-2):35, 2010. 21, 22

[CH15] Chad J Creighton and Shixia Huang. Reverse phase protein arrays in signaling pathways: a data integration perspective. *Drug design, development and therapy*, 9:3519, 2015. 27, 28

[CPM+12] Angelo J Casa, Adam S Potter, Simeen Malik, ZaWaunyka Lazard, Isere Kuiatse, Hee-Tae Kim, Anna Tsimelzon, Chad J Creighton, Susan G Hilsenbeck, Powell H Brown, et al. Estrogen and insulin-like growth factor-i (igf-i) independently down-regulate critical repressors of breast cancer growth. *Breast cancer research and treatment*, 132(1):61–73, 2012. 23

[DC03] Rina Dechter and David Cohen. *Constraint processing*. Morgan Kaufmann, 2003. 44

[DCBL17] Xiaofeng Dai, Hongye Cheng, Zhonghu Bai, and Jia Li. Breast cancer cell line classification and its relevance with breast tumor subtyping. *Journal of Cancer*, 8(16):3131, 2017. 20, 21, 22

[DGH92] Paul Dagum, Adam Galper, and Eric Horvitz. Dynamic network models for forecasting. In *Proceedings of the eighth international conference on uncertainty in artificial intelligence*, pages 41–48. Morgan Kaufmann Publishers Inc., 1992. 37

[DGH+95] Paul Dagum, Adam Galper, Eric Horvitz, Adam Seiver, et al. Uncertain reasoning and forecasting. *International Journal of Forecasting*, 11(1):73–87, 1995. 37

[DLL62] Martin Davis, George Logemann, and Donald Loveland. A machine program for theorem-proving. *Communications of the ACM*, 5(7):394–397, 1962. 90

[DP60] Martin Davis and Hilary Putnam. A computing procedure for quantification theory. *Journal of the ACM (JACM)*, 7(3):201–215, 1960. 90

[DW15] Guangyou Duan and Dirk Walther. The roles of post-translational modifications in the context of protein interaction networks. *PLoS Comput Biol*, 11(2):e1004049, 2015. 14, 28

[EEEF09] Thomas Eiter, Esra Erdem, Halit Erdoğan, and Michael Fink. Finding similar or diverse solutions in answer set programming. In *International Conference on Logic Programming*, pages 342–356. Springer, 2009. 54, 56, 89, 102

[EHN+06] Fons Elstrodt, Antoinette Hollestelle, Jord HA Nagel, Michael Gorin, Marijke Wasielewski, Ans van den Ouweland, Sofia D Merajver, Stephen P Ethier, and Mieke Schutte. Brca1 mutation analysis of 41 human breast cancer cell lines reveals three new deleterious mutants. *Cancer research*, 66(1):41–45, 2006. 22

[Eth96] Stephen P Ethier. Human breast cancer cell lines as models of growth regulation and disease progression. *Journal of mammary gland biology and neoplasia*, 1(1):111–121, 1996. 21

[FAC13] Daniela Ferreira, Filomena Adega, and Raquel Chaves. The importance of cancer cell lines as in vitro models in cancer methylome analysis and anticancer

drugs testing. In *Oncogenomics and cancer proteomics-novel approaches in biomarkers discovery and therapeutic targets in cancer*. InTech, 2013. 21

[FLNP00] Nir Friedman, Michal Linial, Iftach Nachman, and Dana Pe'er. Using bayesian networks to analyze expression data. *Journal of computational biology*, 7(3-4):601–620, 2000. 37

[FMR98] Nir Friedman, Kevin Murphy, and Stuart Russell. Learning the structure of dynamic probabilistic networks. In *Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence*, pages 139–147. Morgan Kaufmann Publishers Inc., 1998. 37

[Fre06] R Ian Freshney. Basic principles of cell culture. *Culture of cells for tissue engineering*, pages 11–14, 2006. 21

[GHW+15] Mario Giuliano, Huizhong Hu, Yen-Chao Wang, Xiaoyong Fu, Agostina Nardone, Sabrina Herrera, Sufeng Mao, Alejandro Contreras, Carolina Gutierrez, Tao Wang, et al. Upregulation of er signaling as an adaptive mechanism of cell survival in her2-positive breast tumors treated with anti-her2 therapy. *Clinical Cancer Research*, 21(17):3995–4003, 2015. 25

[GKKS12] Martin Gebser, Roland Kaminski, Benjamin Kaufmann, and Torsten Schaub. Answer set solving in practice. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 6(3):1–238, 2012. 43, 44

[GKKS14] Martin Gebser, Roland Kaminski, Benjamin Kaufmann, and Torsten Schaub. Clingo= asp+ control: Preliminary report. *arXiv preprint arXiv:1405.3694*, 2014. 62, 87, 90

[GKS12] Martin Gebser, Benjamin Kaufmann, and Torsten Schaub. Conflict-driven answer set solving: From theory to practice. *Artificial Intelligence*, 187:52–89, 2012. 94

[GL88] M. Gelfond and V. Lifschitz. The stable model semantics for logic programming. In R. Kowalski and K. Bowen, editors, *Proceedings of the Fifth International Conference and Symposium of Logic Programming (ICLP'88)*, pages 1070–1080. MIT Press, 1988. 45

[GMN+12] Anita Grigoriadis, Alan Mackay, Elodie Noel, Pei Jun Wu, Rachel Natrajan, Jessica Frankum, Jorge S Reis-Filho, and Andrew Tutt. Molecular characterisation of cell line models for triple-negative breast cancers. *BMC genomics*, 13(1):619, 2012. 24

[GNEJ80] Geoffrey L Greene, Chris Nolan, J Petér Engler, and Elwood V Jensen. Monoclonal antibodies to human estrogen receptor. *Proceedings of the National Academy of Sciences*, 77(9):5115–5119, 1980. 23

[GSTV11] Martin Gebser, Torsten Schaub, Sven Thiele, and Philippe Veber. Detecting inconsistencies in large biological networks with answer set programming. *Theory and Practice of Logic Programming*, 11(2-3):323–360, 2011. 43

[GVE+13] Carito Guziolowski, Santiago Videla, Federica Eduati, Sven Thiele, Thomas Cokelaer, Anne Siegel, and Julio Saez-Rodriguez. Exhaustively characterizing feasible logic models of a signaling network using answer set programming. *Bioinformatics*, 29(18):2320–2326, 2013. 29, 53

[HCM75] KB Horwitz, ME Costlow, and WL McGuire. Mcf-7: a human breast cancer cell line with estrogen, androgen, progesterone, and glucocorticoid receptors. *Steroids*, 26(6):785–795, 1975. 23

[Hei16] Laura Heiser. Hpn-dream breast cancer network inference challenge, April 2016. 58, 68

[HHC+16] Steven M Hill, Laura M Heiser, Thomas Cokelaer, Michael Unger, Nicole K Nesser, Daniel E Carlin, Yang Zhang, Artem Sokolov, Evan O Paull, Chris K Wong, et al. Inferring causal molecular networks: empirical assessment through a community-based effort. *Nature methods*, 13(4):310–318, 2016. 16, 28, 42, 58, 67, 68, 69, 98

[HHOW05a] Emmanuel Hebrard, Brahim Hnich, Barry O'Sullivan, and Toby Walsh. Finding diverse and similar solutions in constraint programming. In *AAAI*, volume 5, pages 372–377, 2005. 54

[HHOW05b] Emmanuel Hebrard, Brahim Hnich, Barry O'Sullivan, and Toby Walsh. Finding diverse and similar solutions in constraint programming. In *Proceedings of the Twentieth National Conference on Artificial Intelligence (AAAI'05)*, pages 372–377. AAAI Press, 2005. 102

[HLM+12] Steven M Hill, Yiling Lu, Jennifer Molina, Laura M Heiser, Paul T Spellman, Terence P Speed, Joe W Gray, Gordon B Mills, and Sach Mukherjee. Bayesian inference of signaling network topology in a cancer cell line. *Bioinformatics*, 28(21):2804–2810, 2012. 28, 37, 39

[HMPL+04] Henning Hermjakob, Luisa Montecchi-Palazzi, Chris Lewington, Sugath Mudali, Samuel Kerrien, Sandra Orchard, Martin Vingron, Bernd Roechert, Peter Roepstorff, Alfonso Valencia, et al. Intact: an open source molecular interaction database. *Nucleic acids research*, 32(suppl_1):D452–D455, 2004. 28

[HNJC+17] Steven M Hill, Nicole K Nesser, Katie Johnson-Camacho, Mara Jeffress, Aimee Johnson, Chris Boniface, Simon EF Spencer, Yiling Lu, Laura M Heiser, Yancey Lawrence, et al. Context specificity in causal signaling networks re-

vealed by phosphoprotein profiling. *Cell systems*, 4(1):73–83, 2017. 16, 28, 38, 42, 58, 67, 68, 69, 85

[HRS+09] Michael Hartmann, Johan Roeraade, Dieter Stoll, Markus F Templin, and Thomas O Joos. Protein microarrays for diagnostic assays. *Analytical and bioanalytical chemistry*, 393(5):1407–1416, 2009. 26

[HS11] Deborah L Holliday and Valerie Speirs. Choosing the right cell line for breast cancer research. *Breast cancer research*, 13(4):215, 2011. 20

[Hus03] Dirk Husmeier. Sensitivity and specificity of inferring genetic regulatory interactions from microarray experiments with dynamic bayesian networks. *Bioinformatics*, 19(17):2271–2282, 2003. 37

[IGT10] Solomon Itani, Joe Gray, and Claire J Tomlin. An ode model for the her2/3-akt signaling pathway in cancers that overexpress her2. In *American Control Conference (ACC), 2010*, pages 1235–1241. IEEE, 2010. 32

[ILM+10] Sergio Iadevaia, Yiling Lu, Fabiana C Morales, Gordon B Mills, and Prahlad T Ram. Identification of optimal drug combinations targeting cellular networks: integrating phospho-proteomics and computational network analysis. *Cancer research*, 70(17):6704–6714, 2010. 31

[Ing] Ingenuity Pathway Analysis. `https://www.qiagenbioinformatics.com/products/ingenuity-pathway-analysis/`. 33, 35

[Ino11] Katsumi Inoue. Logic programming for boolean networks. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence - Volume Volume Two*, volume 22 of *IJCAI'11*, pages 924–930. AAAI Press, 2011. 60

[JSB+98] Meei-Huey Jeng, Margaret A Shupnik, Timothy P Bender, Eric H Westin, Debdutta Bandyopadhyay, Rakesh Kumar, Shigeru Masamura, and Richard J Santen. Estrogen receptor expression and function in long-term estrogen-deprived human breast cancer cells. *Endocrinology*, 139(10):4164–4174, 1998. 23

[JYL+17] Zhiwei Ji, Ke Yan, Wenyang Li, Haigen Hu, and Xiaoliang Zhu. Mathematical and computational modeling in complex biological systems. *BioMed research international*, 2017, 2017. 30, 31, 34

[Kau69] S. A. Kauffman. Metabolic stability and epigenesis in randomly constructed genetic nets. *Journal of theoretical biology*, 22(3):437–467, 1969. 60

[KG00] Minoru Kanehisa and Susumu Goto. Kegg: kyoto encyclopedia of genes and genomes. *Nucleic acids research*, 28(1):27–30, 2000. 28, 33

[KHJ+06] Neil Kumar, Bart S Hendriks, Kevin A Janes, David de Graaf, and Douglas A Lauffenburger. Applying computational modeling to drug discovery and development. *Drug discovery today*, 11(17-18):806–811, 2006. 29

[KSRL+06] Steffen Klamt, Julio Saez-Rodriguez, Jonathan A Lindquist, Luca Simeoni, and Ernst D Gilles. A methodology for the structural and functional analysis of signaling and regulatory networks. *BMC bioinformatics*, 7(1):56, 2006. 53, 88, 96

[KSW17] Roland Kaminski, Torsten Schaub, and Philipp Wanko. A tutorial on hybrid answer set solving with clingo. In *Reasoning Web International Summer School*, pages 167–203. Springer, 2017. 93

[KvIH+12] Thomas Kelder, Martijn P. van Iersel, Kristina Hanspers, Martina Kutmon, Bruce R. Conklin, Chris T. Evelo, and Alexander R. Pico. Wikipathways: building research communities on biological pathways. *Nucleic Acids Research*, 40(D1):D1301, 2012. 28

[LBC+11] Brian D Lehmann, Joshua A Bauer, Xi Chen, Melinda E Sanders, A Bapsi Chakravarthy, Yu Shyr, and Jennifer A Pietenpol. Identification of human triple-negative breast cancer subtypes and preclinical models for selection of targeted therapies. *The Journal of clinical investigation*, 121(7):2750–2767, 2011. 24

[LC81] EY Lasfargues and WG Coutinho. Human breast tumor cells in culture; new concepts in mammary carcinogenesis. In *New Frontiers in Mammary Pathology*, pages 117–143. Springer, 1981. 24

[LG87] Jérome Lebeau and Gérard Goubin. Amplification of the epidermal growth factor receptor gene in the bt20 breast carcinoma cell line. *International journal of cancer*, 40(2):189–191, 1987. 22

[Lif08] Vladimir Lifschitz. What is answer set programming? In *AAAI*, pages 1594–1597. AAAI Press, 2008. 64

[LL04] Marc Lacroix and Guy Leclercq. Relevance of breast cancer cell lines as models for breast tumours: an update. *Breast cancer research and treatment*, 83(3):249–289, 2004. 21

[LO58] Etienne Y Lasfargues and Luciano Ozzello. Cultivation of human breast carcinomas. *Journal of the National Cancer Institute*, 21(6):1131–1147, 1958. 22

[LOD15] Adrian V Lee, Steffi Oesterreich, and Nancy E Davidson. Mcf-7 cells—changing the course of breast cancer research and care for 45 years. *JNCI: Journal of the National Cancer Institute*, 107(7), 2015. 23

[LR06] Vladimir Lifschitz and Alexander Razborov. Why are there so many loop formulas? *ACM Transactions on Computational Logic (TOCL)*, 7(2):261–268, 2006. 44

[Mas00] John RW Masters. Human cancer cell lines: fact and fantasy. *Nature reviews Molecular cell biology*, 1(3):233, 2000. 21

[MBC+15] Shital K. Mishra, Sourav S. Bhowmick, Huey Eng Chua, Fan Zhang, and Jie Zheng. Computational cell fate modelling for discovery of rewiring in apoptotic network for enhanced cancer drug sensitivity. *BMC Systems Biology*, 9(1):S4, Jan 2015. 35, 36

[MLD+91] P Meltzer, A Leibovitz, W Dalton, H Villar, T Kute, J Davis, R Nagle, and J Trent. Establishment of two new cell lines derived from human breast carcinomas with her-2/neu amplification. *British journal of cancer*, 63(5):727, 1991. 24

[MLE10] Claudius Mueller, Lance A Liotta, and Virginia Espina. Reverse phase protein microarrays advance to use in clinical trials. *Molecular oncology*, 4(6):461–481, 2010. 26

[MMM+11] Ioannis N Melas, Alexander Mitsos, Dimitris E Messinis, Thomas S Weiss, and Leonidas G Alexopoulos. Combined logical and data-driven models for linking signalling pathways to cellular response. *BMC systems biology*, 5(1):107, 2011. 41, 42

[MMS+09] Alexander Mitsos, Ioannis N Melas, Paraskeuas Siminelakis, Aikaterini D Chairakaki, Julio Saez-Rodriguez, and Leonidas G Alexopoulos. Identifying drug effects via pathway alterations using an integer linear programming optimization formulation on phosphoproteomic data. *PLoS computational biology*, 5(12):e1000591, 2009. 40, 41, 42

[MR02] Kevin Patrick Murphy and Stuart Russell. Dynamic bayesian networks: representation, inference and learning. 2002. 37

[MS00] Gavin MacBeath and Stuart L Schreiber. Printing proteins as microarrays for high-throughput function determination. *Science*, 289(5485):1760–1763, 2000. 26

[MTH+12] Aidan MacNamara, Camille Terfve, David Henriques, Beatriz Peñalver Bernabé, and Julio Saez-Rodriguez. State–time spectrum of signal transduction logic models. *Physical biology*, 9(4):045003, 2012. 29, 34, 53, 111

[MW07] Wayne Materi and David S Wishart. Computational systems biology in drug discovery and development: methods and applications. *Drug discovery today*, 12(7-8):295–303, 2007. 34

[N+12] Cancer Genome Atlas Network et al. Comprehensive molecular portraits of human breast tumours. nature 490, 61e70, 2012. 77

[Nad11a] Alexander Nadel. Generating diverse solutions in sat. In *International Conference on Theory and Applications of Satisfiability Testing*, pages 287–301. Springer, 2011. 54

[Nad11b] Alexander Nadel. Generating diverse solutions in SAT. In *Proceedings of the Fourteenth International Conference on Theory and Applications of Satisfiability Testing (SAT'11)*, volume 6695, pages 287–301. Springer, 2011. 102

[NBG+01] Monica Nogueira, Marcello Balduccini, Michael Gelfond, Richard Watson, and Matthew Barry. An a-prolog decision support system for the space shuttle. In *International symposium on practical aspects of declarative languages*, pages 169–183. Springer, 2001. 43

[NBK12] E Shannon Neeley, Keith A Baggerly, and Steven M Kornblau. Surface adjustment of reverse phase protein arrays using positive control spots. *Cancer informatics*, 11:CIN–S9055, 2012. 26

[NCF+06] Richard M Neve, Koei Chin, Jane Fridlyand, Jennifer Yeh, Frederick L Baehner, Tea Fevr, Laura Clark, Nora Bayani, Jean-Philippe Coppe, Frances Tong, et al. A collection of breast cancer cell lines for the study of functionally distinct cancer subtypes. *Cancer cell*, 10(6):515–527, 2006. 21

[Nie99] Ilkka Niemelä. Logic programs with stable model semantics as a constraint programming paradigm. *Annals of mathematics and Artificial Intelligence*, 25(3-4):241–273, 1999. 44

[Nis01] Darryl Nishimura. Biocarta. *Biotech Software & Internet Report: The Computer Software Journal for Scient*, 2(3):117–120, 2001. 28

[NLSBW08] Aleksandra Nita-Lazar, Hideshiro Saito-Benz, and Forest M White. Quantitative phosphoproteomics by mass spectrometry: past, present, and future. *Proteomics*, 8(21):4433–4443, 2008. 25

[NSS+08] Domenico Napoletani, T Sauer, Daniele C Struppa, E Petricoin, and L Liotta. Augmented sparse reconstruction of protein signaling networks. *Journal of theoretical biology*, 255(1):40–52, 2008. 39

[NW88] George L Nemhauser and Laurence A Wolsey. Integer programming and combinatorial optimization. *Wiley, Chichester. GL Nemhauser, MWP Savelsbergh, GS Sigismondi (1992). Constraint Classification for Mixed Integer Programming Formulations. COAL Bulletin*, 20:8–12, 1988. 39

[OPS+15] Max Ostrowski, Loïc Paulevé, Torsten Schaub, Anne Siegel, and Carito Guziolowski. Boolean network identification from multiplex time series data. In

Computational Methods in Systems Biology, volume 9308, pages 170–181. Springer, 2015. 15, 16, 29, 43, 52, 58

[OPS+16] Max Ostrowski, Loïc Paulevé, Torsten Schaub, Anne Siegel, and Carito Guziolowski. Boolean network identification from perturbation time series data combining dynamics abstraction and logic programming. *Biosystems*, 149:139–153, 2016. 15, 58, 62, 64, 66, 88, 89

[OS11] C Kent Osborne and Rachel Schiff. Mechanisms of endocrine resistance in breast cancer. *Annual review of medicine*, 62:233–247, 2011. 23

[OSM+74] L Ozzello, B Sordat, C Merenda, S Carrel, J Hurlimann, and JP Mach. Transplantation of a human mammary carcinoma cell line (bt 20) into nude mice. *Journal of the National Cancer Institute*, 52(5):1669–1672, 1974. 22

[OZG+01] Steffi Oesterreich, Ping Zhang, Rebecca L Guler, Xiuhua Sun, Edward M Curran, Wade V Welshons, C Kent Osborne, and Adrian V Lee. Re-expression of estrogen receptor α in estrogen receptor α-negative mcf-7 cells restores both estrogen and insulin-like growth factor-mediated signaling and growth. *Cancer research*, 61(15):5771–5777, 2001. 23

[PCB+01] Cloud P Paweletz, Lu Charboneau, Verena E Bichsel, Nicole L Simone, Tina Chen, John W Gillespie, Michael R Emmert-Buck, Mark J Roth, Emanuel F Petricoin III, and Lance 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, 2001. 26

[Per11] Charles M Perou. Molecular stratification of triple-negative breast cancers. *The oncologist*, 16(Supplement 1):61–70, 2011. 21

[PNK+04] Suraj Peri, J Daniel Navarro, Troels Z Kristiansen, Ramars Amanchy, Vineeth Surendranath, Babylakshmi Muthusamy, TKB Gandhi, KN Chandrika, Nandan Deshpande, Shubha Suresh, et al. Human protein reference database as a discovery resource for proteomics. *Nucleic acids research*, 32(suppl_1):D497–D501, 2004. 28

[Pol11] Kornelia Polyak. Heterogeneity in breast cancer. *The Journal of clinical investigation*, 121(10):3786–3788, 2011. 19

[PZS+10] Jadwiga Pietkiewicz, Katarzyna Zielińska, Jolanta Saczko, Julita Kulbacka, Michał Majkowski, and Kazimiera A Wilk. New approach to hydrophobic cyanine-type photosensitizer delivery using polymeric oil-cored nanocarriers: hemolytic activity, in vitro cytotoxicity and localization in cancer cells. *European journal of pharmaceutical sciences*, 39(5):322–335, 2010. 21

[RCAdS15] Ana Rodriguez, Isaac Crespo, Ganna Androsova, and Antonio del Sol. Discrete logic modelling optimization to contextualize prior knowledge networks using prunet. *PloS one*, 10(6):e0127216, 2015. 28

[RJC+02] Anthony Rhodes, Bharat Jasani, Jérôme Couturier, Mark J McKinley, John M Morgan, Andrew R Dodson, Hossein Navabi, Keith D Miller, and André J Balaton. A formalin-fixed, paraffin-processed cell line standard for quality control of immunohistochemical assay of her-2/neu expression in breast cancer. *American journal of clinical pathology*, 117(1):81–89, 2002. 22

[RKR+18] Misbah Razzaq, Roland Kaminski, Javier Romero, Torsten Schaub, Jeremie Bourdon, and Carito Guziolowski. Computing diverse boolean networks from phosphoproteomic time series data. In *International Conference on Computational Methods in Systems Biology*, pages 59–74. Springer, 2018. 17, 87

[RMD08] Sabry Razick, George Magklaras, and Ian M Donaldson. irefindex: a consolidated protein interaction database with provenance. *BMC bioinformatics*, 9(1):405, 2008. 28

[RPO18] M Razzaq, L Paulevé, and M Ostrowski. Caspo-ts. `https://github.com/misbahch6/caspo-ts`, 2018. 15

[RPS+18] Misbah Razzaq, Loïc Paulevé, Anne Siegel, Julio Saez-Rodriguez, Jérémie Bourdon, and Carito Guziolowski. Computational discovery of dynamic cell line specific boolean networks from multiplex time-course data. *PLoS computational biology*, 14(10):e1006538, 2018. 17, 57

[RSW16] Javier Romero, Torsten Schaub, and Philipp Wanko. Computing diverse optimal stable models. In *ICLP (Technical Communications)*, volume 52 of *OASICS*, pages 3:1–3:14. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2016. 55, 56, 86, 89, 102

[RvJH+13] Muhammad Riaz, Marijn TM van Jaarsveld, Antoinette Hollestelle, Wendy JC Prager-van der Smissen, Anouk AJ Heine, Antonius WM Boersma, Jingjing Liu, Jean Helmijr, Bahar Ozturk, Marcel Smid, et al. mirna expression profiling of 51 human breast cancer cell lines reveals subtype and driver mutation-specific mirnas. *Breast cancer research*, 15(2):R33, 2013. 19

[SBR+06] Chris Stark, Bobby-Joe Breitkreutz, Teresa Reguly, Lorrie Boucher, Ashton Breitkreutz, and Mike Tyers. Biogrid: a general repository for interaction datasets. *Nucleic acids research*, 34(suppl_1):D535–D539, 2006. 28

[Sch98] Alexander Schrijver. *Theory of linear and integer programming*. John Wiley & Sons, 1998. 44

[SDS+10] Ganesh N Sharma, Rahul Dave, Jyotsana Sanadya, Piush Sharma, and KK Sharma. Various types and management of breast cancer: an overview. *Journal of advanced pharmaceutical technology & research*, 1(2):109, 2010. 19

[SFG+14] Renaud Sabatier, Pascal Finetti, Arnaud Guille, José Adelaide, Max Chaffanet, Patrice Viens, Daniel Birnbaum, and François Bertucci. Claudin-low breast cancers: clinical, pathological, molecular and prognostic characterization. *Molecular cancer*, 13(1):228, 2014. 21

[SFL+09] Özgür Sahin, Holger Fröhlich, Christian Löbke, Ulrike Korf, Sara Burmester, Meher Majety, Jens Mattern, Ingo Schupp, Claudine Chaouiya, Denis Thieffry, et al. Modeling erbb receptor-regulated g1/s transition to find novel targets for de novo trastuzumab resistance. *BMC systems biology*, 3(1):1, 2009. 53

[SGTP17] Christoph Sachse, Berthold Gierke, Markus F Templin, and Michael Pawlak. Reverse phase protein arrays (rppa): The assay platform of choice for pathway mapping, drug mode-of-action and multiplex biomarker analyses. 2017. 26

[SHGAL+08] Katherine Stemke-Hale, Ana Maria Gonzalez-Angulo, Ana Lluch, Richard M Neve, Wen-Lin Kuo, Michael Davies, Mark Carey, Zhi Hu, Yinghui Guan, Aysegul Sahin, et al. An integrative genomic and proteomic analysis of pik3ca, pten, and akt mutations in breast cancer. *Cancer research*, 68(15):6084–6091, 2008. 77

[SHM+15] Simon EF Spencer, Steven M Hill, Sach Mukherjee, et al. Inferring network structure from interventional time-course experiments. *The Annals of Applied Statistics*, 9(1):507–524, 2015. 39

[Sie16] Solveig Sieberts. Discovering dynamic molecular signatures in response to virus exposure, May 2016. 111

[Sim99] P. Simons. Extending the stable model semantics with more expressive rules. In M. Gelfond, N. Leone, and G. Pfeifer, editors, *Proceedings of the Fifth International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR'99)*, volume 1730 of *Lecture Notes in Artificial Intelligence*, pages 305–316. Springer, 1999. 46

[SK13] Regina Samaga and Steffen Klamt. Modeling approaches for qualitative and semi-quantitative analysis of cellular signaling networks. *Cell communication and signaling*, 11(1):43, 2013. 34

[SLB+10] Kristina Subik, Jin-Feng Lee, Laurie Baxter, Tamera Strzepek, Dawn Costello, Patti Crowley, Lianping Xing, Mien-Chie Hung, Thomas Bonfiglio, David G

Hicks, et al. The expression patterns of er, pr, her2, ck5/6, egfr, ki-67 and ar by immunohistochemical analysis in breast cancer cell lines. *Breast cancer: basic and clinical research*, 4:117822341000400004, 2010. 20, 21

[SMC+17] Damian Szklarczyk, John H Morris, Helen Cook, Michael Kuhn, Stefan Wyder, Milan Simonovic, Alberto Santos, Nadezhda T Doncheva, Alexander Roth, Peer Bork, et al. The string database in 2017: quality-controlled protein–protein association networks, made broadly accessible. *Nucleic acids research*, 45(D1):D362–D368, 2017. 28

[SMO+03] Paul Shannon, Andrew Markiel, Owen Ozier, Nitin S Baliga, Jonathan T Wang, Daniel Ramage, Nada Amin, Benno Schwikowski, and Trey Ideker. Cytoscape: a software environment for integrated models of biomolecular interaction networks. *Genome research*, 13(11):2498–2504, 2003. 28

[SN99] Timo Soininen and Ilkka Niemelä. Developing a declarative rule language for applications in product configuration. In *International Symposium on Practical Aspects of Declarative Languages*, pages 305–319. Springer, 1999. 43

[SPJ+13] Hongwei Shao, Tao Peng, Zhiwei Ji, Jing Su, and Xiaobo Zhou. Systematically studying kinase inhibitor induced signaling network signatures by integrating both therapeutic and side effects. *PLoS One*, 8(12):e80832, 2013. 33

[SRAE+09] Julio Saez-Rodriguez, Leonidas G Alexopoulos, Jonathan Epperlein, Regina Samaga, Douglas A Lauffenburger, Steffen Klamt, and Peter K Sorger. Discrete logic modelling as a means to link protein signalling networks with functional analysis of mammalian signal transduction. *Molecular systems biology*, 5(1), 2009. 29, 35, 36, 51, 52, 53

[SSK+17] Jodi M Saunus, Chanel E Smart, Jamie R Kutasovic, Rebecca L Johnston, Priyakshi Kalita-de Croft, Mariska Miranda, Esdy N Rozali, Ana Cristina Vargas, Lynne E Reid, Eva Lorsy, et al. Multidimensional phenotyping of breast cancer cell lines to guide preclinical research. *Breast cancer research and treatment*, pages 1–13, 2017. 24

[SVL+73] HD Soule, J Vazquez, A Long, S Albert, and M Brennan. A human cell line from a pleural effusion derived from a breast carcinoma. *Journal of the National Cancer Institute*, 51(5):1409–1416, 1973. 23

[SW18] Torsten Schaub and Stefan Woltran. Answer set programming unleashed! *KI-Künstliche Intelligenz*, pages 1–4, 2018. 44

[SWF+05] S Chandra Shekar, Haiyan Wu, Zheng Fu, Shu-Chin Yip, Sean M Cahill, Mark E Girvin, Jonathan M Backer, et al. Mechanism of constitutive phos-

phoinositide 3-kinase activation by oncogenic mutants of the p85 regulatory subunit. *Journal of Biological Chemistry*, 280(30):27850–27855, 2005. 77

[TCJ+10] Natalie Tulchin, Monique Chambon, Gloria Juan, Steven Dikman, James Strauchen, Leonard Ornstein, Blase Billack, Nicholas T Woods, and Alvaro NA Monteiro. Brca1 protein and nucleolin colocalize in breast carcinoma tissue and cancer cell lines. *The American journal of pathology*, 176(3):1203–1214, 2010. 21

[The18] TheraLink. `http://medfusionservices.com/`, 2018. 28

[TWK+10] Cullen M Taniguchi, Jonathon Winnay, Tatsuya Kondo, Roderick T Bronson, Alexander R Guimaraes, José O Alemán, Ji Luo, Gregory Stephanopoulos, Ralph Weissleder, Lewis C Cantley, et al. The phosphoinositide 3-kinase regulatory subunit p85α can exert tumor suppressor properties through negative regulation of growth factor signaling. *Cancer research*, 70(13):5305–5315, 2010. 77

[VGE+12] Santiago Videla, Carito Guziolowski, Federica Eduati, Sven Thiele, Niels Grabe, Julio Saez-Rodriguez, and Anne Siegel. Revisiting the training of logic models of protein signaling networks with asp. In *Computational Methods in Systems Biology*, pages 342–361. Springer, 2012. 29, 42, 43, 51, 53, 61

[VGR07] Tracy Vargo-Gogola and Jeffrey M Rosen. Modelling breast cancer: one size does not fit all. *Nature Reviews Cancer*, 7(9):659, 2007. 21

[WBB+99] Ignacio I Wistuba, David Bryant, Carmen Behrens, Sara Milchgrub, Arvind K Virmani, Raheela Ashfaq, John D Minna, and Adi F Gazdar. Comparison of features of human lung cancer cell lines and their corresponding tumors. *Clinical cancer research*, 5(5):991–1000, 1999. 21

[WBM+98] Ignacio I Wistuba, Carmen Behrens, Sara Milchgrub, Salahuddin Syed, Mohsen Ahmadian, Arvind K Virmani, Venkatesh Kurvari, Thomas H Cunningham, Raheela Ashfaq, John D Minna, et al. Comparison of features of human breast cancer cell lines and their corresponding tumors. *Clinical Cancer Research*, 4(12):2931–2938, 1998. 21

[WDD+14a] Guanming Wu, Eric Dawson, Adrian Duong, Robin Haw, and Lincoln Stein. Reactomefiviz: a cytoscape app for pathway and network-based data analysis. *F1000Research*, 3, 2014. 16, 28, 70

[WDD+14b] Guanming Wu, Eric Dawson, Adrian Duong, Robin Haw, and Lincoln Stein. Reactomefiviz: a cytoscape app for pathway and network-based data analysis. *F1000Research*, 3, 2014. 28, 70

[WGG+85] Philippe Walter, Stephen Green, Geoffrey Greene, ANDRtE Krust, Jean-Marc Bornert, Jean-Marc Jeltsch, Adrien Staub, Elwood Jensen, Geoffrey Scrace, and Mike Waterfield. Cloning of the human estrogen receptor cdna. *Proceedings of the National Academy of Sciences*, 82(23):7889–7893, 1985. 23

[WMG08] Steven Watterson, Stephen Marshall, and Peter Ghazal. Logic models of pathway biology. *Drug discovery today*, 13(9):447–456, 2008. 34

[WMG+11] Yen-Chao Wang, Gladys Morrison, Ryan Gillihan, Jun Guo, Robin M Ward, Xiaoyong Fu, Maria F Botero, Nuala A Healy, Susan G Hilsenbeck, Gail Lewis Phillips, et al. Different mechanisms for resistance to trastuzumab versus lapatinib in her2-positive breast cancers-role of estrogen receptor and her2 reactivation. *Breast Cancer Research*, 13(6):R121, 2011. 25

[XRS+00] Ioannis Xenarios, Danny W Rice, Lukasz Salwinski, Marisa K Baron, Edward M Marcotte, and David Eisenberg. Dip: the database of interacting proteins. *Nucleic acids research*, 28(1):289–291, 2000. 28

[YXW+10] Y Ye, Y Xiao, W Wang, K Yearsley, JX Gao, B Shetuni, and SH Barsky. Erα signaling through slug regulates e-cadherin and emt. *Oncogene*, 29(10):1451, 2010. 21

[ZAUH16] Peican Zhu, Hamidreza Montazeri Aliabadi, Hasan Uludağ, and Jie Han. Identification of potential drug targets in cancer signaling pathways using stochastic logical models. *Scientific reports*, 6, 2016. 89

[ZDX+12] Chi Zhang, Xuening Duan, Ling Xu, Jingming Ye, Jianxin Zhao, and Yinhua Liu. Erythropoietin receptor expression and its relationship with trastuzumab response and resistance in her2-positive breast cancer cells. *Breast cancer research and treatment*, 136(3):739–748, 2012. 25

[ZMPQ+02] Andreas Zanzoni, Luisa Montecchi-Palazzi, Michele Quondam, Gabriele Ausiello, Manuela Helmer-Citterich, and Gianni Cesareni. Mint: a molecular interaction database. *FEBS letters*, 513(1):135–140, 2002. 28

[ZT13] Ying Zhu and Miroslaw Truszczynski. On optimal solutions of answer set optimization problems. In *International Conference on Logic Programming and Nonmonotonic Reasoning*, pages 556–568. Springer, 2013. 55, 56

Titre : Intégration de données de séries temporelles phosphoprotéomiques dans des réseaux de connaissances antérieurs

Mots clés : réseaux de signalisation, programmation logique, vérification de modèles, lignées cellulaires

Résumé: Les voies de signalisation canoniques traditionnelles aident à comprendre l'ensemble des processus de signalisation à l'intérieur de la cellule. Les données phosphoprotéomiques à grande échelle donnent un aperçu des altérations entre différentes protéines dans différents contextes expérimentaux. Notre objectif est de combiner les réseaux de signalisation traditionnels avec des données de séries temporelles phosphoprotéomiques complexes afin de démêler les réseaux de signalisation spécifiques aux cellules. Côté application, nous appliquons et améliorons une méthode de séries temporelles caspo conçue pour intégrer des données phosphoprotéomiques de séries temporelles dans des réseaux de signalisation de protéines. Nous utilisons une étude de cas réel à grande échelle tirée du défi HPN-DREAM Breast Cancer. Nous déduisons une famille de modèles booléens à partir de données de séries temporelles de perturbations multiples de quatre lignées cellulaires de cancer du sein, compte tenu d'un réseau de signalisation protéique antérieur. Les résultats obtenus sont comparables aux équipes les plus performantes du challenge HPN-DREAM. Nous avons découvert que les modèles similaires sont regroupés dans l'espace de solutions. Du côté informatique, nous avons amélioré la méthode pour découvrir diverses solutions et améliorer le temps de calcul.

Title : Integrating Phosphoproteomic Time Series Data into Prior Knowledge Networks

Keywords : signaling networks, logic programming, model checking, cell lines

Abstract: Traditional canonical signaling pathways help to understand overall signaling processes inside the cell. Large scale phosphoproteomic data provide insight into alterations among different proteins under different experimental settings. Our goal is to combine the traditional signaling networks with complex phosphoproteomic time-series data in order to unravel cell specific signaling networks. On the application side, we apply and improve a caspo time series method conceived to integrate time series phosphoproteomic data into protein signaling networks. We use a large-scale real case study from the HPN-DREAM Breast Cancer challenge. We infer a family of Boolean models from multiple perturbation time series data of four breast cancer cell lines given a prior protein signaling network. The obtained results are comparable to the top performing teams of the HPN-DREAM challenge. We also discovered that the similar models are clustered together in the solutions space. On the computational side, we improved the method to discover diverse solutions and improve the computational time.