Functional Significance Checking in Biological Networks: Theory and Implementation

Submitted in partial fulfillment of the requirements for the degree of

MTech PhD Dual Degree

by

Sukanya Basu
Roll No : 09305908

under the guidance of

Prof. Supratik Chakraborty
Prof. Akshay S

Department of Computer Science and Engineering
Indian Institute of Technology, Bombay

2019

Declaration

I declare that this written submission represents my ideas in my own words and where others' ideas or words have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Sukanya Basu

Sukanya Basu
Roll no. 09305908

Date 20.06.2020

i

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Biological scientists try to understand the mechanism behind specific phenomena. Why do cells behave in a certain way? What signals among the cells cause disease-like behavior? Can these signals be altered? How does a drug work? Why does a drug not work? There are myriads of such questions in biology that are still not well understood.

To answer such questions, biologists perform a lot of wet-lab experiments and make various observations. Additionally, the advent of high-throughput experimental techniques has made it possible to collect data about interactions amongst thousands of genes, biochemical molecules, proteins, or other entities, in a single biological experiment. This large amount of accumulated data has hugely increased the size and complexity of models used to represent information obtained from such experiments.

A gene is a sequence of specific organic molecules called nucleotides. Genes encode the genetic information inherited from parents and is responsible for manifestation of biological characteristics of a person. Each gene has one or more specific functions. *Expression* is the process by which a gene undergoes a tranformation into gene products, mostly proteins. The extent of expression is measured by the amount of the corresponding protein being produced.

Fig. 1.1 shows a schematic sketch of the process of expression by which a gene transforms into a functional product, most often a protein. Over-expression (or up-regulation) of a gene means excessive expression that produces too much of its product. Repression (or down-regulation) of a gene indicates the suppression of its effect, which results in less than usual production of the protein.

Proteins are functional units of a cell and it is the behavior of the proteins that manifest as a biological function. Aberrant behavior of some proteins might result in a condition that we identify as a disease. We might be able to understand the cause of a disease by measuring the expression of genes and by studying the interactions of genes with one another.

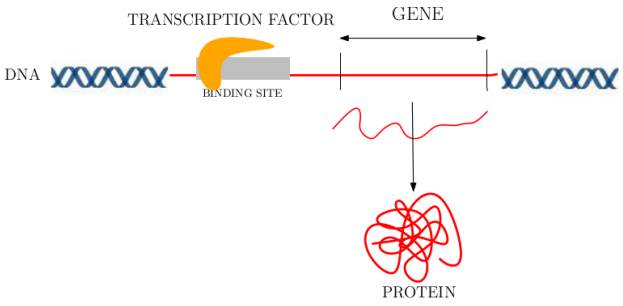


Figure 1.1: Schematic representation of expression of a gene into protein

Microarray experiments One widely popular group of wet-lab experiments is gene over- or under-expression experiments. In an over(under)-expression experiment, the expression of one gene is increased(decreased) externally, and the biologists observe expressions of the other genes. One way of capturing this phenomenon is through microarrays. Single timepoint microarray experiments measure the expression of thousands of proteins simultaneously. It represents a snapshot of the system in great detail, but this observation is limited to a single time point. Even a single timepoint microarray data can indicate individual profiles like an allergic disease, or a specific kind of cancer, which is suggestive of how the whole system functions.

Scientists are accumulating vast amounts of data using increasingly sophisticated techniques, like microarrays. However, there is a lack of understanding of the internal workings of a system. Hence, all of these experimental observations and readings are not getting exploited fully. The challenge to extract useful knowledge from such rich data has now become a task that is best to automate using a machine or software. This opportunity has spurred much exciting research over the last few decades, and entirely new areas of study like bio-informatics and systems biology have emerged and flourished.

1.1 Motivation

Motivation of this thesis is rooted in a practical problem faced in wet-lab experiments. In a typical single point molecular biology experiment, a stimulus is given to specially prepared cells under controlled conditions and expressions of different entities are observed. Time-series gene expression data provides a unique glimpse into the dynamics of underlying processes. They can be used to identify new causal relationships between entities. However, there are practical difficulties as-

sociated with obtaining multiple gene expression snapshots. Firstly, the spectrum of time constants involved in the many interactions participating in the biological process under investigation may be broad, and not known a priori. The full spectrum makes it difficult to decide an optimal sampling period for observing all the relevant dynamics. Secondly, despite significant improvements in technology, observing a full gene expression profile today is still at a price point where a scientist needs to decide and budget the measurement plan carefully. In the vast majority of cases, complete gene expressions are available for only one or two time points for a biological experiment.

A large amount of public and validated knowledge on how complex sequences of gene regulatory mechanisms work in different contexts are freely available online. Though information on regulatory pathways is incomplete in a large number of contexts, a collection of such pathways provides an excellent source of information on gene interactions. In this work, the highly curated and publicly available KEGG database [1] was used to compensate for the causal interaction information that is lacking in a single timepoint microarray dataset.

In this work, the goal was to computationally find an explanation of sparse snapshots of gene expression data, compensating for the lack of a detailed time series by consulting well-curated databases of regulatory pathways.

The goal of this thesis was to design computational techniques that would cover the following requirements.

(i) An approach was to be developed to work well with a sparse, probably even a single snapshot of gene expression profile. It is practical to work with sparse data for the limitations mentioned earlier. All of the current validations mentioned in this work are on human cell lines. This approach could work with sparse data, and in the future, can even be extended to single patient data.

(ii) A method was to be designed to handle a limited quantity of noise in both the expression profile and in the publicly-available gene regulatory pathway information. The need for noise tolerance is two-ways. Firstly, there might be errors in both the array readings and the network curation. Secondly, culminating edges from pathways of very different contexts would, in principle, string together a path, but this path may not be functional or feasible. Allowing limited noise handles both these issues.

(iii) A potentially involved actor was to be identified that played a *functionally significant* role in plausible explanations of the gene expression profile. From a biological point of view, the only way to answer this question is by doing a

knock-out experiment, where the biologists methodically remove the potential actor and repeat the experiment. However, each such experiment costs non-trivial time and money. The hope was to provide a relatively cheap and fast computational alternative, that can also provide valuable insights to biologists, to design better experiments with lesser effort.

1.2 Main contributions

Here is presented a summary of the main contributions of the thesis.

1.2.1 Identifying *functionally significant* nodes

First, is presented a new problem formulation for checking whether an actor(a biological entity) plays a significant functional role in explaining an observed gene expression data profile.

At the start, explanation subgraphs are queried for from a given stimulus node to a target observation node. The explanation subgraphs must be consistent with the observed gene expression profile.

Noise and inconsistencies are inherent in the networks and microarray data. Most often, the KEGG pathways do not match precisely with the experimental profile. Hence a consistent subgraph from the KEGG pathways, explaining all of the experimental observation data from the microarray profile might not exist. Then, often no such explanation subgraph is obtained. In such cases, to yield an explanation, minimal nodes and edges can be changed (relaxed) within a user-defined bound, following some heuristics.

Furthermore, when solutions exist, a second query is generated by excluding some nodes from the solutions. Results of the second query might lead to two different outcomes:

(i) There is no solution at precisely the same number of node and edge relaxations. More of the nodes or more of the edges or both, need to be relaxed to obtain solutions. This outcome indicates that the excluded node was essential to obtain a solution at that level of noise.

(ii) There continue to be solutions at the same number of node and edge relaxations. This result implies, either the excluded node was not necessary. Alternatively, on its exclusion, some other compensatory mechanisms kicked in and produced solutions at the same level of noise.

Informally, a node is declared to be *functionally significant* if its absence requires to admit more noise for obtaining an explanation.

In earlier work, SAT-solving techniques have been used to find few small explanatory pathways assuming either zero or a few known perturbations in the experimental observations. Unfortunately, these approaches do not work in many situations, where the noise or perturbation may not be known a priori and the number of possible pathways generated by a SAT-solver is too large to be analyzed by enumeration. In such settings, determining if an actor plays a functionally significant role towards explaining an experiment is very difficult using existing SAT-based techniques.

In this work, the problem of functional significance checking in gene-regulatory pathways in the presence of unknown but bounded perturbations is formalised. We show that this problem is Δ_2^P-hard in general and hence cannot be efficiently encoded as a SAT problem. An algorithm is proposed that uses a polynomial number of SAT-oracle invocations to solve a practically useful version of this problem.

Finally, the algorithm presented in this work detects functional significance of a gene without actually enumerating the potentially explosively many explanations of the observed gene expressions while admitting bounded noise. Pareto-optimality and comparison of two Pareto-optimal curves are used in achieving this. This way it is possible to analyze much larger systems of gene interactions than in earlier work [2]. This approach is discussed in detail later in Chapter 5.5.

1.2.2 Complexity results

The next contribution of this thesis is to show the complexity of the problems addressed here. The following are the results.

1. The problem of checking for the existence of a solution subgraph is NP-complete.

2. Checking the existence of an (n, e)-relaxed explanation subgraph is NP-complete.

3. Checking *functional significance* within a user-defined window W is coNP-complete.

4. If the relaxation window W is part of the input, checking for *functional significance* is Δ_2^P-hard.

The thesis presents complete characterizations of the computational complexity of the problems under different settings. NP-completeness of the problems can be shown even under relatively liberal assumptions. These results imply that no existing polynomial-time algorithm for identifying essential nodes in a network can solve the problem unless P = NP. The problem admits efficient algorithmic solutions only in exceptional cases, perhaps unlikely to occur in practice.

1.2.3 Approach implemented as a tool

Finally, a tool is developed that implements the techniques described above and checks for the *functional significance* of specific genes in some targeted biological experiments.

The prediction of the approach is indeed borne out in wet-lab experiments wherein the specific genes were deliberately inhibited or deactivated. Interestingly, this approach is also able to identify actors that are unlikely to play a significant role or have compensatory pathways, even when the actor is deactivated.

1.3 Chapter outline

Chapter 2 gives a literature survey of the various techniques that are relevant to the main aspects of the approach. It also discusses in detail the background required for an easy read of the rest of the thesis. Chapter 3 presents the formal problem definition. Computational results on the hardness of the problems along with the hardness proofs are in Chapter 4. Chapter 5 gives details of the steps of the approach. Chapter 6 gives the computational experiments and results, along with a brief discussion on the biological validation. The overall conclusions are in the final Chapter 7.

Chapter 2

Preliminaries and literature survey

The layout of the literature survey section is as follows. It begins with a few surveys that extensively cover the representation of biological systems as computational models. Section 2.1 will discuss influence graphs that are the most common representation used for small biological networks. It also presents a discussion on the sign-consistency models on influence graphs, that try to match an experimental profile to a network structure. Next there is a brief description of the answer set programming used in this context. Then, we discuss several existing tools that handle biological data at various levels of detail.

Next we present a brief description of the methods and results of our initial experience of using a few tools. These tools try to learn the network structure from a given microarray data automatically. It is followed by a discussion on the reasons for choosing manually curated databases for the analysis. Following that is a detailed description of the semantics of KEGG database pathways.

Finally, there is a short discussion on graph theoretic techniques like centrality measures and minimum cut that handle and solve various network problems quickly. However, as discussed later on, these techniques were not sufficient for quickly identifying functionally significant candidate nodes. The query is modeled as a multi-objective optimization problem, and Pareto-optimal curves are used to make predictions. A review of techniques used to obtain Pareto-optimal curves in various circumstances concludes the literature survey.

2.1 Representation of biological networks

The sequence of steps in a biological process that culminate in a common outcome with a specific function is often termed as a biological *pathway*, eg., calcium

metabolism pathway, pancreatic cancer pathway, etc. In other words, a biological pathway is a component of a biological process.

Biological processes have been modeled in various ways, using Petri-nets, ODEs, set of rules, Boolean networks, and others. Many surveys are available on this topic [3,4]. Of these, the network representation was most suitable for the work in this thesis, and hence, the approach here interprets biological pathways as graphs or networks. Section 3.1 discusses the exact representation.

Several network models show the mechanism of a certain biological process. These representations are called *pathways* or *maps*. The terms *pathways*, *maps*, *networks*, *graphs*, and in some instances, *subgraphs*, will be used interchangeably in this document. The terms *nodes* or *vertices* would mean genes or gene products, while the terms *edges* are the same as *interactions* depending on the context.

Influence graphs

Before going into the graph representation of biological networks used in this thesis, here we discuss a similar but very simplistic, yet common representation of biological systems called *influence graphs* [5].

An influence graph is defined as a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \sigma)$, where,

$\mathcal{V}$: set of vertices representing the genes (or proteins),

$\mathcal{E}$: set of edges representing the interactions between those genes (or proteins),

$\sigma : \mathcal{E} \to \{+, -\}$ is a possibly partial labeling of the edges.

An edge between nodes $u, v \in \mathcal{V}$ is represented as $u \longrightarrow v$. An edge $u \longrightarrow v$ with a $+$ label represents activation of gene transcription (or protein). An edge $u \longrightarrow v$ with a $-$ label represents inhibition of gene transcription or inactivation of protein.

Sign consistency model

Sign consistency model (SCM) is a simple, qualitative model based on influence graph that is used to constraint the GRN with experimental profiles [6]. An experimental profile, say, $\mu : \mathcal{V} \to \{+, -\}$ is a (partial) labeling of the vertices of the graph which can be incomplete or even noisy. Each vertex of the graph is assigned as either input or non-input. For every non-input vertex $v \in \mathcal{V}$, the sign $\mu(v)$ is *consistent*, if there is some edge $u \to v$ in $\mathcal{E}$ such that $\mu(u) = \mu(v) \cdot \sigma(u \to v)$, where the multiplication of signs $\cdot$ is same as that of numbers. The idea of consistency does not apply to input vertices.

There is a notion of *mutual consistency* between an influence graph or model, and an experimental profile or data. A model $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \sigma)$ and a data profile μ, are mutually consistent iff there exists total labelings $\sigma' : \mathcal{E} \to \{+, -\}$ and $\mu' : \mathcal{V} \to \{+, -\}$, which are total extensions of σ and μ, respectively, such that $\mu'(v)$ is consistent for every non-input vertex $v \in \mathcal{V}$ [8]. When μ is undefined for every vertex of the graph, it is interpreted as the experimental profile being absent. This check is equivalent to checking for the self-consistency of the graph [7].

Checking for consistency

Now comes the question of how to check if an experiment profile is consistent with a given influence graph. Though there are other approaches, in relation to this thesis, we are interested in the ones using satisfiability checking [8, 9].

The Boolean satisfiability problem (SAT) looks for an assignment to the variables of a Boolean formula that satisfies it. [8] used an influence graph to encode a gene regulatory network and validated it against an experimental profile using SAT and MaxSAT solvers.

Problem instance using SAT

Of various methods, the approach in this thesis is closest to [8]. Hence this section discusses their model in detail. According to [8], an influence graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \sigma)$ and respective experimental profile μ can be encoded into a SAT query as follows.

Variables

First, they introduce three types of Boolean variables for the nodes and the edges.

(i) For each vertex $v \in \mathcal{V}$,

  (a) A Boolean variable inp_v which takes the value $true$ if v is an input vertex, and $false$ otherwise.

  (b) A Boolean variable $lvtx_v$ (label vertex) which takes the value $true/false$ if the corresponding label $\mu(v)$ is $+/-$.

(ii) For each edge $u \to v \in \mathcal{E}$,

  (a) A Boolean variable $ledg_{uv}$, such that $ledg_{uv}$ takes value $true/false$ if the corresponding label $\sigma(u \to v)$ is $+/-$.

(iii) Another type of variable is used to denote the influence between vertices. For each edge $u \to v \in \mathcal{E}$, a Boolean variable $infl_{uv}$ is used such that $infl_{uv}$ takes value $true/false$ if $\mu(v) \cdot \sigma(u \to v)$ evaluates to $+/-$.

Constraints

The constraints corresponding to unit clauses are,

(i) For each vertex $v \in \mathcal{V}$, a unit clause (inp_v) is introduced if v is an input vertex. Otherwise, the unit clause $(\neg inp_v)$ is introduced.

(ii) Given labellings μ/σ, one unit clause is introduced for each vertex/edge that has a label.

- For each vertex $v \in \mathcal{V}$ with $\mu(v) = +/-$ a unit clause $(lvtx_v)/(\neg lvtx_v)$ is introduced.
- For each edge $u \to v \in \mathcal{E}$ with $\sigma(u \to v) = +/-$ a unit clause $(ledg_{uv})/(\neg ledg_{uv})$ is included.

Non-unit clauses are required for defining the value of the variables $infl$. The value of these variables is given by $\mu(v) \cdot \sigma$.

(iii) For each edge $u \to v$, the following constraints are added:

$$infl_{uv} \longrightarrow (lvtx_u \wedge ledg_{uv}) \vee (\neg lvtx_u \wedge \neg ledg_{uv})$$
$$\neg infl_{uv} \longrightarrow (lvtx_u \wedge \neg ledg_{uv}) \vee (\neg lvtx_u \wedge ledg_{uv})$$

These are the final constraints for checking consistency between the network and the data. As discussed earlier, an influence graph and an experimental profile are mutually consistent if total extensions for μ and σ can be found such that all non-input vertices labels are consistent. When u may be any vertex to which v is adjacent, then use of variables $infl_{uv}$ ensures consistency of vertex v.

(iv) For each vertex v, the following constraints are added:

$$inp_v \vee (lvtx_v \longrightarrow \bigvee_u infl_{uv})$$
$$inp_v \vee (\neg lvtx_v \longrightarrow \bigvee_u \neg infl_{uv})$$

Finally, a SAT solver takes the entire CNF encoding and generates an output that reveals whether or not a model is consistent. Another call along with the experimental profile shows if the model is consistent with the experimental profile.

Initially, the approach in this thesis might seem to be similar to [8]. However, the approach described in this work has multiple advantages over the one outlined here. Out work can scalably handle large system level networks at gene level details to find consistent solution subgraphs. It is robust towards possible noise in the experimental data and can even identify individual key players from possibly incompletely curated pathways. So far, such a balance of robustness and accuracy is not known to the best of our knowledge.

Answer set programming

Answer set programming (ASP) is yet another technique for obtaining models to a set of logical rules. The initial introduction of ASP was in the context of planning and scheduling strategies or actions in artificial intelligence [10]. It is a form of declarative programming focused on searching for models, primarily for NP-hard problems [11]. It also finds uses for detecting inconsistencies, repair, and prediction in large biological networks [12, 13].

2.2 Existing computational tools

This section will discuss different computational tools that handle similar problems at various levels of detail. Some analyses reduce networks to a list of genes. [14] showed a novel way to group genes that share common biological function, chromosomal location, or regulation. It used the Kolmogorov-Smirnov test and listed pathways in decreasing order of involvement or enrichment. [15–18] explore genes as a group at the level of a pathway.

[19] and [20] retain the underlying topology structure and display expression values as colors on nodes. [21–24] try to distinguish pathways using topology and expression information alone. The approaches in [25–31] try to find enriched pathways based only on the gene information, whereas [32–41] use both gene-expression and pathway topology to select candidates for enriched pathways.

Most of these tools do not take into account the wealth of information present in each interaction of a pathway. Exploiting full annotation on the interactions is a complicated task. Still, some tools [42–46] have tried to take this information into account. Very recently [47–49] have stringently analysed relations between genes. From each pathway, they try to find sub-pathways consistent with the data profile.

The work presented in this thesis treats network information and extensive wet-lab validation on human cell-lines with rigor and is novel in its exhaustiveness and robustness. Human system-level studies, with sparse data that yields such specific output have not been performed before.

2.3 Learning networks from microarray data

Mutual information

Most of the mutual information based techniques used for extracting gene regulatory networks from microarray expression data follow two main steps:

(i) Finding a relatedness measure between entities.

(ii) Considering interactions with relatedness above a certain threshold.

However, the methods differ in ways they perform these two crucial steps and sometimes performing some further analyses. The following are some well-known algorithms.

Relevance Network

The relevance network is a technique, originally developed with the goal of exploiting existing databases of laboratory test experiments to ascertain known physiological phenomena [50]. The variables (nodes in the final network) represent physiological entities like serum osmolarity, serum sodium, whole blood sodium, among others. The process involves (i) computing, for each pair of variables, a correlation coefficient r, by clustering patient and experiment data, (ii) selecting links with r^2 above a certain threshold. The resulting network gave islands of nodes called the *relevance networks*.

Mutual information (MI) measures the dependency between two variables. For discrete variables X and Y, MI is defined as

$$I(X,Y) = -\sum_{x\in X, y\in Y} p(x,y)\,\log\frac{p(x,y)}{p(x)p(y)}.$$

For constructing relevance networks, mutual information is calculated for every pair of genes. A value on every pair of genes in the network gives a completely connected graph with some value of mutual information on each edge. At the next step, a threshold mutual information (TMI) value is chosen. In the resulting network, only the genes that have a link to others with mutual information higher than the TMI are selected. The resulting network is some, often disjoint, cluster(s) of genes, called the relevance network.

Experiments on benchmarks

The experimental data is the ultimate truth and would have to be explained by any model claiming to represent the system accurately. Given that, it is a naturally

intuitive step to try and obtain graphical models from the experimental data alone. Experiments were performed with some tools (eg., ARACNe [51, 52], TINGe [53, 54]) that tried to arrive at the interactions between genes starting solely from the microarray data.

However, it was seen that these tools were not accurate, especially in the large scale networks that were of interest to this thesis. The outputs generated from these tools did not only include false positive results but also left out several true positive ones.

Two popular tools are compared here. The automated network learning experiments were conducted to evaluate the performance of the tool TINGe [53, 54] and ARACNe [51, 52]. These experiments were performed on networks obtained from the DREAM5 Network Inference Challenge [55].

Network 1

This network came with artificially generated microarray data.

- *in-silico* network generated using GeneNetWeaver (GNW) version 3.0
- 1643 genes on 805 chips
- Gold standard file contained 4012 interactions

The network learning tools expected some p-value and tolerance of error margin. Various values were tried for both p-value and tolerance. Results showed here used the Bonferroni corrected p-value, that is, 0.05/(number of genes in the input file) and a few values for error tolerances.

In Table 2.1, the header is to be read as follows. The "Tolerance" value represents an allowance for error. The "Nodes" and "Edges" are the numbers of nodes and edges learned for a particular solution. The column "TP" represents true positives, the interactions or edges learned that are present in the standard file. The "FP" or false positives column indicates edges that are deduced by the tool but are not present in the original network. The "FN" or false negatives are the interactions that are present in the actual network but not deduced by the tool. The "TN" or true negatives are the edges that are not present and also not reported by the tool.

P-value = 1.56e-07

	Tolerance	Nodes	Edges	TP	FP	FN	TN
Tinge	0.05	1625	9969	700	9269	3296	1210565
Aracne		1611	9743	650	9093	3346	1210741
Tinge	0.001	1629	5333	545	4788	3451	1215046
Aracne		1611	6089	492	5597	3504	1214237
Tinge	0.0001	1618	5982	521	5461	3475	1214373
Aracne		1611	6029	490	5539	3506	1214295

Table 2.1: Network 1 results

While the high value of TN is encouraged, a relatively small value of TP indicates that many edges occurring in the original network were missed out by the tools. Such behavior is not very desirable as it will not be able to infer all the actual biological interactions. Also, the high value of FP means there were many interactions reported that were not part of the actual network. These inaccuracies are undesirable in the setting of biology. While these can miss out on a possible explanation, spurious solutions might lead the user down the wrong track.

Network 2

- Based on real microarray data in E. coli
- Gold standard constructed from RegulonDB Release 6.8
- 4511 genes on 805 chips
- Contains 2066 interactions

Similar results for a second network are presented in Table 2.2. This is a real network with the above details.

P-value = 1.1e-05

	Tolerance	Nodes	Edges	TP	FP	FN	TN
Aracne	0.01	4437	10510	55	10455	2000	571230
Tinge		4346	10497	59	10438	1996	571247
Aracne	0.05	4437	17850	68	17782	1987	563903
Tinge		4347	18063	71	17992	1984	563693

Table 2.2: Results from network 2

From these experiments, the observation was that the networks thus learned were far from accurate. The tools could not retrieve all interactions, while there were many false positive interactions reported. The level of accuracy was not sufficient to use them for practical purposes. It was decided to take the help of networks curated by experts.

2.4 Manually curated databases

There are several databases, eg., KEGG [56], BioCarta [57], Reactome [58], among others, that host many pathways depicting relationships between genes, proteins, diseases, drugs, and other chemical substances at a system level. Most of these databases provide an intuitive graphical representation of the process/pathway, and some of them also provide a textual description that can be used to automate the analysis. The KEGG database was chosen for this work for the high level of trust in its curation, its easily manageable format, and freely available content.

2.4.1 KEGG database and semantics

The Kyoto Encyclopedia of Genes and Genomes (KEGG) database [1, 59] is a reference knowledge base for systems-level interpretation of data resources. The KEGG pathways are available online [56] and also for download. It is developed and maintained since 1995 by Kanehisa Laboratories at Kyoto University and the University of Tokyo. Experts mostly create this database with some modules generated computationally.

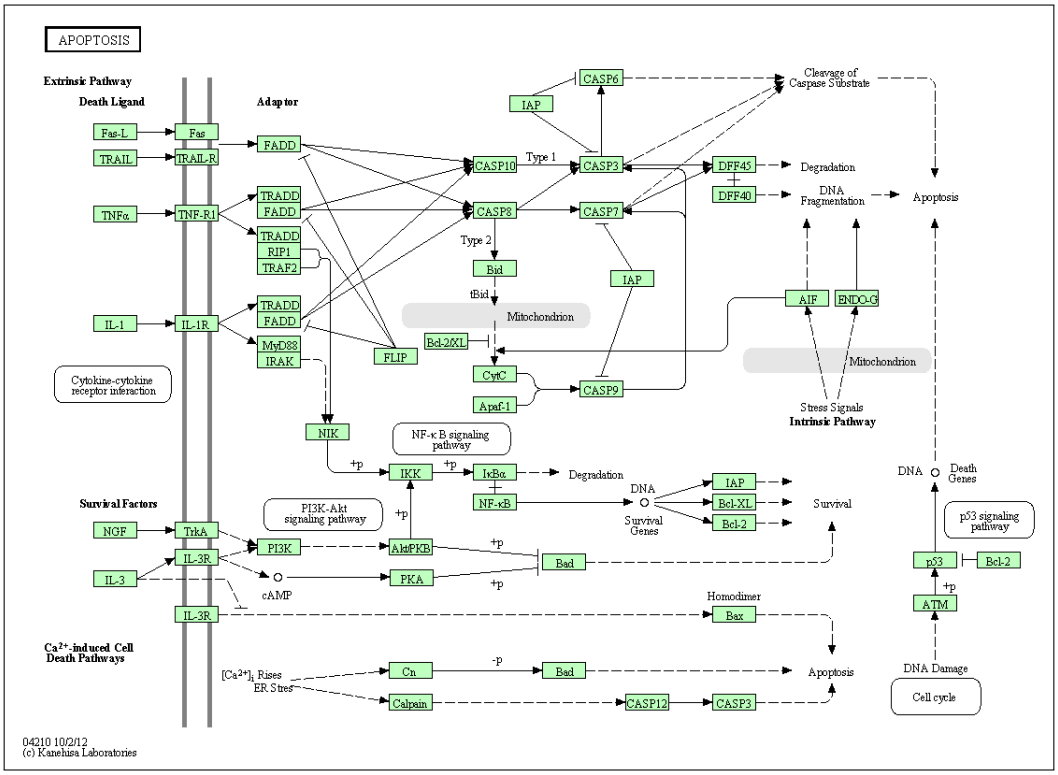


Figure 2.1: KEGG apoptosis pathway for humans

It contains pathways for humans and other microscopic and macroscopic organisms as well. It stores and regularly updates reference and individual pathways for a multitude of species. An entity in KEGG is identified with a code with a three letter prefix denoting the species (for humans it is "hsa"). Fig. 2.1 shows the apoptosis pathway for humans[1].

KEGG provides the pathways to be downloaded in an XML like format called KGML. Of the various pathways in KEGG depicting processes in human beings, some present the information at a molecular level, while some represent a higher level structure at the gene/protein level. Several of the molecular level details were purely diagrammatic, and information was not available in the form that can be automatically handled at this moment. Hence this work is with only the information available in the background. Sometimes the background file information is short of or different from the image shown. In such cases, the strategy was to restrict ourselves to the background information only (except in cases mentioned later).

[1] https://www.genome.jp/kegg-bin/show_pathway?hsa04210

Nodes

The nodes in KEGG may represent different entities, like, genes or proteins, RNAs, protein complexes, chemical compounds, or DNA molecules. In the graphic representation, the nodes provide a link to a page dedicated to the said gene, compound, and others, which contain further information on them. Some nodes are linked to other pathways. The legend of KEGG provided alongside shows how these nodes may look like in a network.

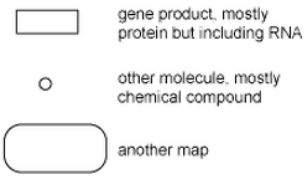


Figure 2.2: Nodes legend in KEGG

Edges

The nodes are connected among themselves by edges that represent an interaction. KEGG provides several types and subtypes of edges with different semantics for each of them.

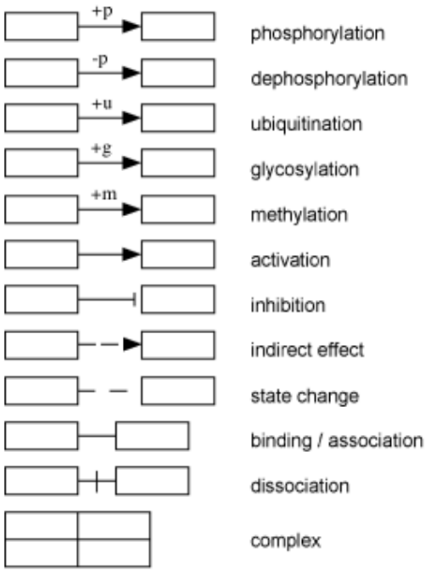


Figure 2.3: PPrel edge subtypes in KEGG

Protein-protein relation or PPrel This type of edge exists between two proteins or protein complexes and is the most abundant. It can have the following subtypes.

- **Phosphorylation (or dephosphorylation):** Interaction denoting that the source entity adds (or removes) a phosphate group from the target entity
- **Ubiquitination:** The source node attaches a ubiquitin molecule to the target node.
- **Glycosylation:** A reaction in which the source attaches a carbohydrate, i.e., a glycosyl donor, to the target node.
- **Methylation:** The source node attaches a methyl group to the target node.
- **Activation:** This kind of edge suggests that the source node activates, or enables, the target to perform its function. In this kind of interaction, the source node might act as a transcription factor (TF) of the target node.
- **Inhibition:** Reverse of activation, if the source node is active, an inhibition edge disables the target from performing its action.
- **Indirect effect:** It is known that the source affects the target, but details of this molecular interaction are missing.
- **State change:** The activity of the source node results in a change of state of the target node.
- **Binding/association:** An undirected edge usually represents this. It denotes that the components structurally bind together and perform some task.
- **Dissociation:** Opposite of association, the components separate from each other.
- **Complex:** The components come together to act as a complex of different proteins.

Protein-compound relation or PCrel This is similar to the PPrel type, but here one of the source or target nodes is a chemical compound, eg., SO_2, NO_2, Ca, and others. A protein either affects or is affected by the compound. The subtypes are also similar to the above type PPrel, eg., activation, inhibition.

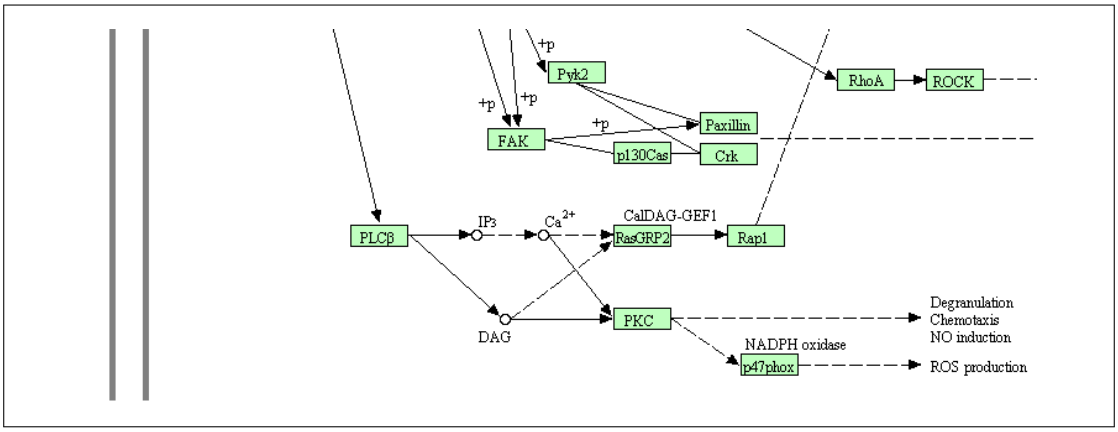
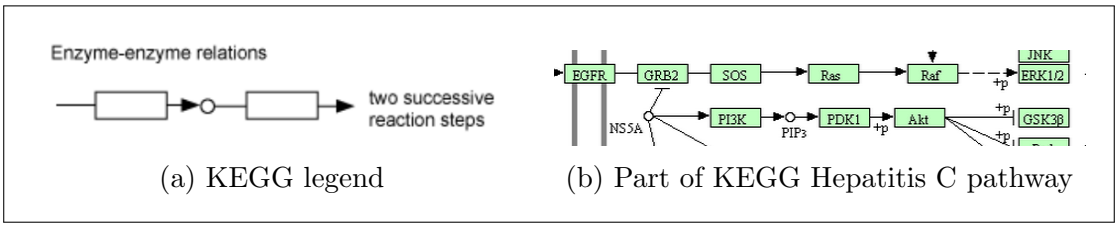


Figure 2.4: Part of KEGG Chemokine signaling pathway

Examples can be seen in the Figure 2.4, a snippet from the Chemokine signaling pathway[2].

Enzyme-enzyme relation or ECrel These edges represent two successive enzyme catalysis reactions via a compound.



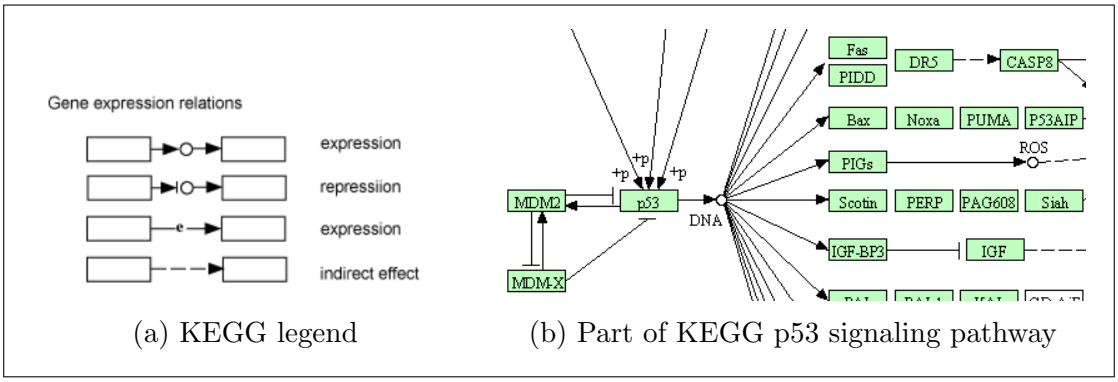
(a) KEGG legend (b) Part of KEGG Hepatitis C pathway

Figure 2.5: Enzyme-enzyme relation or ECrel edge

The example shows a part of the KEGG Hepatitis C pathway[3] where the edge between PI3K and PDK1 goes through the compound Phosphatidylinositol-3,4,5-trisphosphate (PIP3).

Gene expression relation or GErel When a protein expresses or represses a gene, this is represented by the GErel relation. The source protein (usually a TF for the target) expresses the gene that is responsible for producing the target protein. Since this expression occurs at the gene level, it involves the DNA and is made intuitive by showing this by a small circle.

[2] https://www.genome.jp/kegg-bin/show_pathway?hsa04062
[3] https://www.genome.jp/kegg-bin/show_pathway?hsa05160



(a) KEGG legend (b) Part of KEGG p53 signaling pathway

Figure 2.6: Gene expression relation or GErel edge

In the Figure 2.6, part of p53 signaling pathway[4], p53 is shown to express a bunch of genes, like FAS, BAX, Scotin, and many more.

Modeling generalizations in this thesis Different types of edges are grouped into two main groups. (i) Expression (negative as repression) that deal with the expression of a gene into a protein, and (ii) Activation (negation form as inhibition). All other directed edges were considered as activating or inhibiting based on the context. Edges like association and dissociation. whose semantics were unclear, have been ignored.

Expression vs activation For the model in this work, it was necessary to distinguish between expression and activation of a gene.

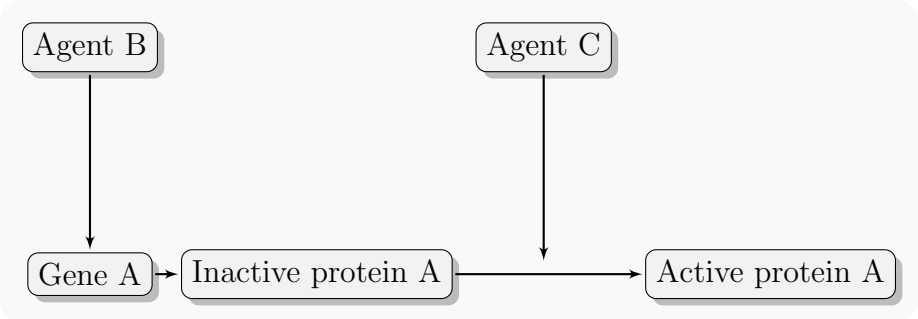


Figure 2.7: Expression vs activation

A gene needs to be expressed for its product to be active. For example, in Fig. 2.7, agent B expresses gene A, which forms protein A.

[4] https://www.genome.jp/kegg-bin/show_pathway?hsa04115

It is possible that this protein A, will be in its inactive form and say, it needs agent C to get active. This agent could be a compound, a protein, or even some external factor, like a light. It would hence be said that B expresses A, and C activates A. However, if protein A is active, gene A must have been expressed. Fig. 2.8 shows how this would be represented in KEGG.

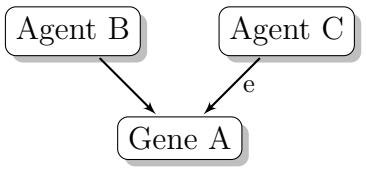


Figure 2.8: In KEGG representation

2.4.2 Selecting KEGG pathways

One hundred sixty-three pathways were identified from KEGG for which it was possible to parse the KGML files and construct a network in the internal representation. The list of pathways is available in Appendix A. The tool developed as a part of this thesis read these files, constructed the graphs for each of the networks, and merged them into one *master network*. When two graphs are merged, a union is taken of the nodes in the two graphs. If there are two edges in two different graphs, between the same pair of nodes and with the same annotation (e.g. activation, phosphorylation, possibly others), then only one edge was retained in the merged graph. However, if the annotations on these two edges were different, both the edges were retained between these pairs of nodes as parallel edges.

2.5 Graph theoretical techniques and centrality measures

The problem of identifying important actors and connections in a network has been extensively studied in multiple contexts, including (but not limited to) the world-wide-web, social media networks, communication networks and gene regulatory and protein-protein interaction networks. Some of the more essential metrics and ranking schemes that have evolved as a result of these studies are mentioned here. Deficiencies of these metrics in the context of explanation subgraphs for a specific source node s and target node t are also elaborated here.

Centrality measures

Degree centrality in undirected graphs uses the degree of a node, or edges connected to the node, to determine the "importance" of a node [60]. Thus, nodes with a higher degree are considered more valuable. For directed graphs, the *outdegree centrality* measures the number of outgoing edges from a node, and the *indegree*

prestige measures the number of incoming edges to a node. In protein-protein interaction networks and gene regulatory networks, nodes with high degree centrality are also called *hubs*. Nodes with high measures of degree centrality potentially participate in multiple paths. However, they are not specific to the source and target nodes of interest in the explanation subgraph problem. Also, these metrics do not take into account the fact that a node with indegree one and outdegree one can still have *all* paths from the source to the target passing through it, even though there are other nodes with high degree centrality measures.

The *closeness centrality* of a node in an undirected graph is defined as the inverse of its peripherality [61,62], which is simply the sum of its shortest distances to all other nodes in the graph. A node with a high value of closeness centrality is closer to most nodes in the graph. As before, this metric is insensitive to the specific source-target pair. Moreover, two nodes with the same closeness centrality can have two completely different distribution of distances to other nodes in the graph adding the distances to all other nodes obliterates these differences when computing the metric.

Betweenness centrality is a widely used metric that measures the fraction of all-pairs *shortest paths* in a network that passes through a given node [63]. This metric is effectively computed by considering each pair of source-target vertices (s, t), computing the shortest paths between them and finding the fraction of these shortest paths that pass through a chosen vertex v. By summing this fraction over all (s, t) pairs, the betweenness centrality of v is obtained. Nodes with high betweenness centrality are often identified as *bottlenecks* in protein-protein interaction networks and gene regulatory networks. Since betweenness centrality is aggregated over all (s, t) pairs in the network, it is easy to see that it is insensitive to the specific choice of source and target nodes in the explanation subgraph problem.

Furthermore, the use of betweenness centrality and any derivative metric is premised on the assumption that the shortest paths are the only ones that play an essential role in a regulatory or interaction network. Unfortunately, this is not always true, and hence, betweenness centrality can miss key players that participate in the regulatory mechanism through short, but not necessarily shortest paths [64].

Eigenvector based centrality measures have been intensely studied in the context of understanding the structure of the world-wide-web and social media networks. These measures assign scores to nodes based on the idea that all connections of a node may not be equally meaningful in determining the score of the node. Specifically, a connection to a high-scoring node contributes more to the score of the node in question than a connection to a low-scoring node. If an adjacency matrix A represents the interconnections of nodes in the network, the eigenvector centrality of nodes in the network is given by the (non-negative) eigenvector cor-

responding to the highest eigenvalue of the adjacency matrix. The components of the eigenvector can be further normalized if one wishes to assign an absolute eigenvector centrality score to each node. The PageRank algorithm [65] and its variants used by internet search engines employ variants of eigenvector centrality measures.

Betweenness for bottleneck detection

Such graph-theoretic metrics have been widely used to detect crosstalk in large biological networks [64, 66]. These techniques usually, rank the nodes by a particular centrality and ascertain a top fixed percentage of the sorted nodes as hubs or bottlenecks. The biological community primarily uses [64]'s definition of a bottleneck as the top 20% of the nodes sorted by betweenness centrality. A discussion about the use of betweenness centrality as a first level filter can be found later in Section 5.3.

2.6 SAT encoding of biological networks

In [67], SMT-solvers have been used to analyze robustness under mutations of gene regulatory networks. While encoding the problem of finding an explanation from known pathways and expression data as a SAT problem has similarities with other work in literature [2, 68], such an encoding often does not explain subgraphs or too many of them as solutions to the SAT problem. Crucially it does not solve functional significance checking when expression data and knowledge about pathways are noisy unless experts examine every explanation subgraph for all noise perturbations. Checking this is too intensive to be realistically performed widely. The primary differentiator of this work vis-a-vis these earlier work is in the way this approach models noise and implicitly considers all possible noisy inputs subject to the bounded weighted noise, while still requiring a polynomial number of SAT invocations.

Finally, identifying important actors in a network has been studied in multiple contexts, including the web, social media networks, gene regulatory, and protein-protein interaction networks. Various graph-theoretic metrics have been used to detect crosstalk and identify hubs and bottlenecks in large biological networks [64, 66]. The work in this thesis can be used in tandem with these techniques by first obtaining potential candidates for functional significance checking using graph theoretic techniques, and then actually checking their functional significance using the proposed approach.

2.7 Multi-objective optimization problem

Multi-objective optimization is a variant of decision-making problems where multiple criteria are to be optimized at the same time. Many real-world problems are multi-objective in behavior, where the different objectives often conflict with each other. eg., cost vs. quality, speed vs. accuracy.

In a multi-objective optimization problem, the solution is a vector of decision variables. These different decision variables must satisfy all given constraints where the conditions are often inter-dependent. At the same time, the solution must also optimize a vector function whose elements represent the, often conflicting, objective functions. Optimization looks for only those solutions, which give values of the objective functions, all of which are acceptable to the user.

The multi-objective optimization problem (MOOP) takes the general form:

For n decision variables, we need to find the solution vector

$$\bar{x}^* = [x_1^*, x_2^*, \ldots, x_n^*]^T$$

which satisfy m inequality constraints

$$g_i(\bar{x}) \geq 0,\ i = 1, 2, \ldots, m$$

and p equality constraints

$$h_i(\bar{x}) = 0,\ i = 1, 2, \ldots, p$$

while optimizing the vector function

$$\bar{f}(\bar{x}) = [f_1(\bar{x}), f_2(\bar{x}), \ldots, f_k(\bar{x})]^T$$

Each objective function either needs to be minimized or maximized. A feasible solution is a solution that satisfies all the constraints.

$$g_i(\bar{x}) \geq 0,\ i = 1, 2, \ldots, m$$

$$h_i(\bar{x}) = 0,\ i = 1, 2, \ldots, p$$

define the *feasible region* F, which is the set of all the feasible solutions [110].

Example

Suppose Sam is buying a car. Car A is cheap but uncomfortable; car B is comfortable but pricey. If price and comfort are the only two criteria he is judging a car by, then he cannot make an informed decision about which car, A or B, is better. Hence in this optimization problem, there are more than one objectives:

1. Minimize cost
2. Maximize comfort

Also the objectives are conflicting:

- Reducing cost implies reducing comfort
- Increasing comfort causes an increase in cost

For Sam, cars A and B are incomparable. To make an informed decision, he then goes to a car expert, who from his experience chooses car B because of mileage, maintenance, lifespan, and many things that Sam does not quite understand. Here the car expert is the domain expert who has the knowledge to choose one option over the other.

Multi-objective optimization is well-studied over four decades. Here is a brief outline of the various techniques for tackling this problem and obtaining the Pareto-optimal curve.

2.7.1 Methods with *a priori* preference articulation

This group of techniques assumes it is desirable to achieve some goal, or there is a preferred ordering of the objectives. They take into account specific parameters, like coefficients, exponents, bounds, and combine them to help in decision making before performing a search.

1. **Weighted global criterion method** Here the aim is to minimize a *weighted global criterion* function. Few standard forms of this function and some popular coefficient and exponent values are given by [69–79], though other functions and values can be used.
2. **Weighted sum method** Highly intuitive and one of the earliest, this is the most common approach for solving multi-objective optimization problems [76–85]. Usually, it is seen as a particular case of the previous method, where the exponent of the function to be minimized is set to 1, and one can obtain the weighted sum of the different parameters. However, many shortcomings of this approach have been identified [77–79, 85].
3. **Lexicographic method** Objective functions are ordered by importance, and individual optimization problems are solved with a single objective function, in the sequence in which they are ordered. Note that, after solving the problem with the first objective function, the set of constraints might change in the subsequent problems [86–88].

4. **Weighted min-max method** This method also called *weighted Tchebycheff* method, introduces an additional unknown parameter λ, moreover, tries to minimize λ while trying to maximize each objective function [76, 78, 79, 89]. [90–92] present modified or augmented versions of the weighted min-max method.

5. **Exponentially weighted criterion** To overcome the shortcoming of the weighted sum method to capture points in the non-convex area, [77] introduced this method were the exponential is weighted by a parameter. Minimizing this criterion is shown to be sufficient for obtaining the Pareto-optimal curve.

6. **Weighted product method** Weights, according to relative significance, are put as the exponent on the objective functions and their product, instead of sum, is considered, making it potentially non-linear [93]. This approach is not a popular method due to the computational difficulties associated with solving the non-linear function.

7. **Goal programming method** In this method, a goal is specified for each objective function and the total deviation of the goals, usually, a summation function of the individual goals is minimized [72, 73, 89, 94–98].

8. **Bounded objective function method** This method minimizes only the most crucial objective function. Other objective functions form additional constraints within lower and upper bounds l and ϵ respectively [72]. A popular variant of this method is the ϵ-*constraint* approach which excludes the lower bound l and and considers only constraints involving the upper bound ϵ [86, 89, 99–103].

9. **Physical programming** Goals, objectives, and preferences are grouped into individual metrics by the general form like monotonically increasing, monotonically decreasing, and others [78, 104, 105]. Additional constraints define numerical ranges that are desirable, tolerable, or undesirable for these metrics.

2.7.2 Methods with *a posteriori* preference articulation

In some instances, having *a priori* preference function might not be feasible. It might be more intuitive to select from the available solutions. This group of methods is called the *cafeteria* or *generate-first-choose-later* approaches [105].

1. **Physical programming** Though initially developed as an *a priori* approach, physical programming can effectively find the entire Pareto-optimal

set even for non-convex regions. In this case, instead of representing predetermined preferences of the decision-maker, the constant constraints are modified systematically to find the complete Pareto-optimal set [105, 106].

2. **Normal boundary intersection (NBI) method** It finds an even distribution of Pareto-optimal points for a consistent variation in the parameters supplied by the user [107].

3. **Normal constraint (NC) method** This provides an improvement over the previous method. It uses normalized objective functions with a Pareto-filter to eliminate the points that are not Pareto optimal [108].

2.7.3 No preference articulation

Often the user may not be clear about their requirements, and it might be challenging to articulate the preferences. Many of the methods in the two previous categories have been extended to work without specific method parameters [69, 72, 73, 86, 109].

2.8 Pareto-optimality

While a unique minimum or maximum is expected in case of a problem involving a single objective, for multiple objectives, it is quite rare to have a single $\mathbf{x}^*$, such that

$$f_1(\bar{x}^*) <= f_1(\bar{x}) \; \forall \bar{x} \in F, \; i = 1, 2, \ldots, k$$

It usually represents a front or a curve, depending on the dimension of the objective function. The criteria that will determine what constitutes an *optimal* solution need to be established.

Definition 2.8.1. Pareto-optimum *According to the definition in [110], a point $\bar{x}^*$ is Pareto-optimal if, for every $\bar{x} \in F$, either*

$$f_i(\bar{x}) = f_i(\bar{x}^*), \; i = 1, 2, \ldots, k$$

or there is at least one $i \in \{1, 2, \ldots, k\}$, such that $f_i(\bar{x}) > f_i(\bar{x}^*)$.

A point is called Pareto-optimal (PO) if none of the criteria can be improved in value without making some other criteria worse off at the same time. Without domain knowledge to lead to preference information, all Pareto optimal solutions are considered equally good.

2.8.1 Domination

Definition 2.8.2. Domination *According to the definition in [110], A solution* $\mathbf{x_1}$ *is said to dominate the other solution* $\mathbf{x_2}$*, if both the following conditions hold:*

1. *The solution* $\mathbf{x_1}$ *is* no worse *than* $\mathbf{x_2}$ *in all objectives, for all* $j = \{1, 2, \ldots, M\}$.
2. *The solution* $\mathbf{x_1}$ *is* strictly better *than* $\mathbf{x_2}$*, in at least one objective, for* at least *one* $\bar{j} \in \{1, 2, \ldots, M\}$.

If any of the above condition is violated, the solution $\mathbf{x_1}$ *does not dominate the solution* $\mathbf{x_2}$.

2.8.2 Pareto-optimal curve

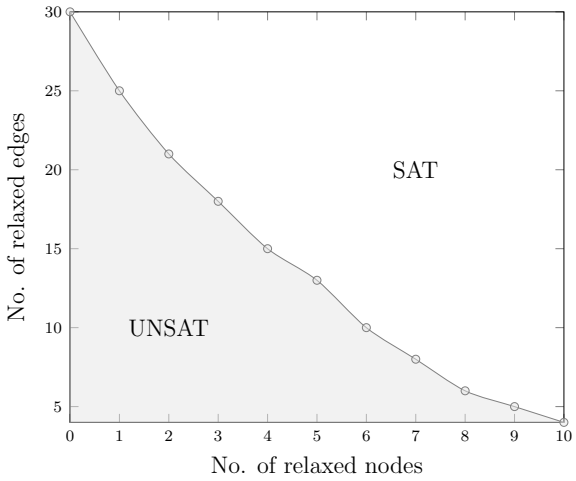


Figure 2.9: Pareto optimal curve for relaxation bounds

As shown in Fig. 2.9, a curve is obtained from the Pareto-optimal points. Every point below the curve is UNSAT, i.e., there are no solutions with those bounds, and every point above the curve is SAT, i.e., they have solutions. The points on the differentiating front constitute the Pareto-optimal curve.

Chapter 3

Problem formulation

Here is the formal statement of the problem and its different variants addressed in this thesis. Expected input are: (i) biological pathway networks from KEGG database (ii) a single time-point gene expression profile (obtained from microarray, RNA sequencing, or other techniques) (iii) a stimulus node s and target observation node t, and (iv) a node of interest i, whose functional significance is a matter of investigation.

The source is usually an upstream player, like membrane receptors, which can be easily perturbed to bring about a cascade of changes. The target node is usually a gene that represents a phenotype; for example, morphological changes in the cell. The node of interest can be anywhere along the cascade. If this node is too high up in the cascade, it will most likely affect too many downstream players, thus bringing in inadvertent changes that obscure the phenomenon that the user is trying to observe. On the other hand, if the central node is too downstream, in its absence alternate pathways might kick in to restore the behavior. Hence identifying a node of interest with just the right amount of influence on the system presents itself as a non-trivial problem.

While the results in this thesis have used KEGG [1] to encode domain knowledge of gene interactions in the experiments, the abstract problem formulation is not specific to KEGG. To keep the description simple, assume that there are only two types of edges: activating ($\mathcal{A}$) and their semantically negative counterparts inhibiting ($\mathcal{I}$). The domain knowledge of gene interactions is given as an edge labeled graph $G_{dom} = (V, E, \mu)$, where V is the set of genes, $E \subseteq V \times V$ is the set of interactions (directed edges) between genes, and $\mu : E \longrightarrow L_e$ is a labeling of edges with $L_e = \{\mathcal{A}, \mathcal{I}\}$. The interpretation of edges is as follows: For an edge $e = (u, v)$, if $\mu(e) = \mathcal{A}$, then gene v must be activated whenever u is active. Also, an activating edge (u, v) is consistent if both u and v are in the inhibited state. Similarly, if $\mu(e) = \mathcal{I}$, v must be inhibited whenever u is active. An inhibiting edge (u, v) is consistent with u being inhibited and v being active.

3.1 A node and edge labeled input graph

Information from the gene expression profile (data) is incorporated as "influence" on the nodes, using labels $L_v = \{+, -, ?\}$. Here, $+$ stands for overexpressed (and by implication, active) nodes, $-$ represents underexpressed (and by implication, inhibited) nodes and ? is used for unknown values or non-differentially expressed nodes. For simplicity of description, $*$ denotes either $+$ or $-$, but not both. The labels on the edges are from a finite set $L_e = \{\mathcal{A}, \mathcal{I}\}$

The domain knowledge and gene expression profile can be represented together as a node-labeled and edge-labeled graph $G = (V, E, \lambda, \mu)$, where V is the set of nodes, E is the set of edges, $\lambda : V \to L_v$ and $\mu : E \to L_e$.

Suppose, three nodes $s, t, i \in V$ are given. s represents a stimulus gene, the effect of whose activation we wish to study, t represents a target gene that is eventually activated or inhibited (possibly through a chain of interactions) due to activation of s, and i represents a suspect gene whose functional significance in the activation or inhibition of t by s is the subject of our investigation. For simplicity, we fix $\lambda(s) = \lambda(t) = +$; other combinations of $\lambda(s)$ and $\lambda(t)$ are handled similarly.

As discussed in section 2.4.1, KEGG considers many more labels, but the types are restricted to the subset $\{\mathcal{A}, \mathcal{I}\}$ for simplicity. Undirected interactions like associations and dissociation are ignored since their semantics is unclear, and they lack directionality. In general, interactions like phosphorylation, ubiquitination, and others are considered as activation.

Taking the union of all pathways in KEGG gives a large *master network*. The master network is an edge labeled graph. During the union, if there are edges on different graphs between the same pair of nodes, having non-matching sets of labels, those edges are replicated to retain each set of labels. The annotation is preserved but at the cost of possibly increasing the total number of solutions. This model of the master network is called an *input graph*.

Definition 3.1.1. Input graph *An* input graph *is an L_v-node and L_e-edge labeled directed graph $G = (V, E, \lambda, \mu)$, where, V is the set of interacting entities (nodes), $E \subseteq V \times V$ is the set of interactions (edges) between entities, L_v is the set of vertex labels as defined above, L_e is the set of edge labels as defined above, $\lambda : V \longrightarrow L_v$ is a labeling of vertices, and, $\mu : E \longrightarrow L_e$ is a labeling of edges.*

3.2 Explanation subgraph

In order to formally define *functional significance*, an *explanation subgraph* needs to be defined first. Informally, this is a subgraph of G that contains s-t paths along with a labeling of nodes that "explains" the observed gene expression profile

while being consistent with the domain knowledge. An example is presented before proceeding to the definition.

Example

Consider a hypothetical wet-lab experiment. Suppose, cancer cells are treated with a drug known to activate gene A, and how it affects the activation of another gene G in cancerous cells is to be determined.

For simplicity, assume that only seven genes named A, B, C, D, E, F and G potentially play any role in the outcome of the experiment. Let the gene expression profile obtained at an appropriate time instant during the experiment be as follows: A, B, G over-expressed, C under-expressed and D, E, F did not show any significant difference in the expressions relative to that of a typical (non-cancerous) cell. From this, it can be inferred that A, B and G are activated and C is inhibited in the context of the experiment. Genes D, E and F in the cancerous cells could either be in their respective ground states (as in a normal cell), or could even be in mildly activated or mildly inhibited states (mild enough so that they do not express their effect overwhelmingly in the gene expression profile).

Suppose, domain knowledge indicates that mutual interactions of genes A through G are as shown in the graph in Fig. 3.1 (sans the +/- labelings). In this figure, a $\rightarrow$ denotes an activating interaction ($A \rightarrow F$ implies that if A is active, F must also be active) and a $\dashv$ denotes an inhibiting edge ($A \dashv B$ implies that if A is active, B must be inactive). Since the ground state (in a normal cell) of a gene may itself be activated or inhibited, we must be careful in interpreting the edges. For example, the edge $A \rightarrow F$ not only admits both A and F are activated, but also admits both are inhibited. To see why this makes sense, note that if F has an inhibited ground state, then an inhibited A cannot activate F through an edge $A \rightarrow F$. Similarly, $B \dashv C$ not only admits B activated and C inhibited, but also vice versa, i.e., C has an activated ground state, and B, being itself inhibited, cannot inhibit C.

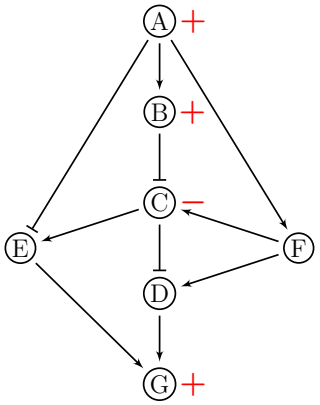
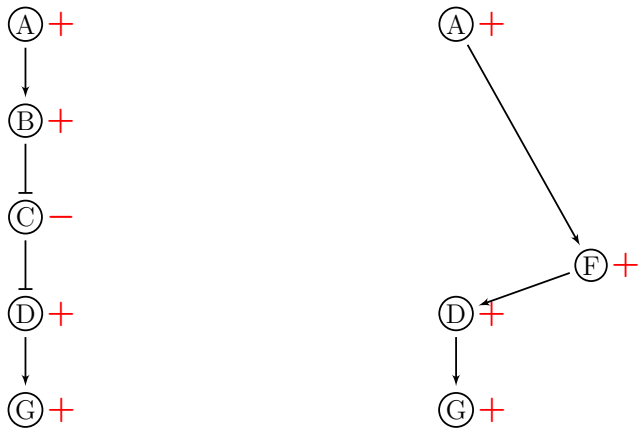


Figure 3.1: Example encoding of gene interactions

Domain knowledge is encoded in a graph like Fig. 3.1, we represent activation levels of genes in the experiment under study by +/- labelings of nodes, where the activated genes are labeled "+" and the inhibited ones are labeled "-".

The first goal is to determine if there exists a set of paths from A to G in Fig. 3.1, and a +/- labeling of nodes along these paths, such that the labeling is consistent



(a) One path (b) Another path

Figure 3.2: Source-target paths with no noise in the example graph

with both the observed gene expressions and domain knowledge. Informally, such a set of paths explains the experimental observations consistently with domain knowledge. In this example, it is indeed possible to find such an explanation with two paths from A to G, namely: $A(+) \to B(+) \dashv C(-) \dashv D(+) \to G(+)$ as shown in Fig. 3.2a and $A(+) \to F(+) \to D(+) \to G(+)$ as in Fig. 3.2b.

The formal definition of the *explanation subgraph* given below was obtained after considering the requirements of credible explanation paths specified by molecular biology experts.

Definition 3.2.1. Explanation subgraph *Let $G = (V, E, \lambda, \mu)$ be as defined above, and let s and t be nodes in V, s.t. $\lambda(s) = \lambda(t) = +$. An* explanation subgraph *of (G, s, t) is a node- and edge-labeled graph $G' = (V', E', \lambda', \mu')$ s.t.,*

1. ***Subgraph containing* s, t***: We require $V' \subseteq V$, $E' \subseteq E \cap (V' \times V')$, μ' is the restriction of μ to E', and $s, t \in V'$.*

2. ***Labels consistent with observed expressions***: *$\lambda'(v) \in \{+, -\}$ for all $v \in V'$, and $\lambda'(v) = \lambda(v)$ if $\lambda(v) \neq ?$.*

3. ***No floating nodes***: *Every $v \in V'$ is reachable from s in G'.*

4. ***Activity condition***: *Every s-t path of length > 1 in G' passes through some node $v \notin \{s, t\}$ with $\lambda(v) = +$. Effectively, for a pathway to credibly explain how s eventually activates t, it must be supported by at least one other active node along the pathway. This condition ensures that completely*

spurious paths without any support from the observed gene expressions are not included.

5. ***Compatible labeling**: For every edge $e = (u, v)$ in E', if $\mu'(e) = \mathcal{A}$, then $\lambda'(u) = \lambda'(v)$, and if $\mu'(e) = \mathcal{I}$, then $\lambda'(u) \neq \lambda'(v)$. Moreover, every node other than s in G' must have at least one incoming compatible edge.*

It is to be noted that edge type $\mathcal{A}$ and $\mathcal{I}$ are sematically opposite. Hence the simplication mentioned earlier, which restricts the edge labels to the set $L_e = \{\mathcal{A}, \mathcal{I}\}$, does not allow multiple edges between the same pair of nodes in the input or solution graphs. This restriction is imposed only to keep the formal definitions simple. Few other edge labels are allowed in the implementation. Hence multiple parallel edges are also admitted. This distinction of edges enables multiple types of outcome at the target node as will be seen in some experimental results in Chapter 6.

Example

In Fig. 3.1, the path $A(+) \rightarrow F(+) \rightarrow D(+) \rightarrow G(+)$ does not constitute an explanation subgraph because the activity condition is violated. However, $A(+) \rightarrow B(+) \dashv C(-) \dashv D(+) \rightarrow G(+)$ is an explanation subgraph.

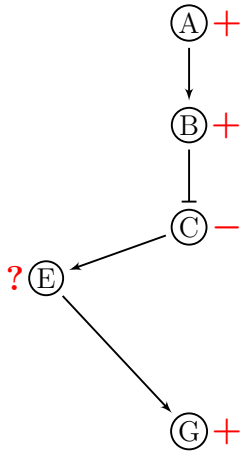


Figure 3.3: Topological path that is not a solution

A few points can be observed in the example of Fig. 3.1. Although F was not differentially expressed in the observed profile, it is fine to assign label "+" to F in the explanation, since F could indeed have been in a mildly activated state that did not result in strong gene expression. Also, though F and C are included in

the explanation, the induced edge $F \to C$ is not included since the labelings of C and F are not consistent with $F \to C$.

Finally, the presence of a topological path from A to G through C does not necessarily imply that this path explains the experimental observations consistently with domain knowledge. For example, although there is a topological path $A(+) \to B(+) \dashv C(-) \to E(?) \to G(+)$ as shown in Fig. 3.3, there is no way of assigning a label ("+" or "-") to E that is consistent with the interpretation of activating and inhibiting edges. Thus, finding explanations is significantly harder than finding topological paths or induced subgraphs.

3.3 Relaxation constraints

The initial experiments with real microarray data and KEGG pathways gave an unexpected result. In many cases, no explanation subgraphs could be found at all. After a more in-depth investigation into the results, some explanations were obtained. There were two primary reasons for lack of any solutions:

(i) the pathway information in KEGG did not fully relate to the context in which the microarray experiments were performed, and

(ii) there was noise in the microarray data.

The notion of relaxation was introduced to handle such noisy input.

3.3.1 Modeling noise

Specifically, an integer relaxation weight is associated with each node and edge and, node and edge labels are allowed to be changed when finding an explanation subgraph. The total node noise (resp. edge noise) introduced to obtain an explanation subgraph is simply the sum of the relaxation weights of all nodes (resp. edges) whose labels had to be ignored or changed to obtain the explanation subgraph. The admissible noise is restricted by specifying an upper bound n and e for node and edge noise, respectively. For notational convenience, (n, e) denote the relaxation bounds in the subsequent discussion.

(i) **Noise from the microarray data.** Suppose, a node x is present in the master network on a path from source s to target t but is not in a form that is consistent with the explanation subgraph. If there is high trust in the pathway, then an aberration in the microarray data need to be acknowledged. This aberration is the noise or error on the nodes.

(ii) **Noise from the network interaction.** On the other hand, suppose, the wet-lab data can be trusted totally. Now in the explanation subgraph, if there is an edge that, in its original form, is not matching the profile, then it is to be said that there is improper or incomplete curation in KEGG, in other words, a noise in the pathways or edges.

The ideal subgraph is likely to be a tradeoff between the two types of noise, which can only be determined after careful inspection. Hence the formulation in this work was developed to be robust to both kinds of noise.

As an example, if all relaxation weights are assigned to 1, then the node relaxation weight is simply the number of nodes relaxed, and the edge relaxation weight is simply the number of edges relaxed. Node and edge labels are allowed to be changed up to specified *relaxation bounds* to admit the possibility of changes due to noise in the database and difference in experimental contexts. A relaxation bound of (n, e) permits at most n nodes and e edges to be changed (or relaxed).

Formally, given a subgraph G' of G, a node in G is said to be *relaxed* in G' if one of the following holds:

(i) it is labeled $+$ in G but is absent in G', i.e., a node active in G is excluded from G',

(ii) it is labeled $+$ in G, and is present and labeled $-$ in G'

(iii) it is labeled $-$ in G and is present and labeled $+$ in G'.

Note that if a node is inhibited in G but excluded in G', it is not treated as relaxed.

Similarly, an edge $e = (u, v) \in G$ is *relaxed* in G' if $u, v \in V'$ and $e \in E'$ and one of the following holds:

(i) $\mu(e) = \mathcal{A}$ and $\mu'(e) = \mathcal{I}$

(ii) $\mu(e) = \mathcal{I}$ and $\mu'(e) = \mathcal{A}$

This brings the discussion to the formal definition of *relaxed explanation* or *relaxation*.

Definition 3.3.1. Relaxed explanation *Given an input graph $G = (V, E, \lambda, \mu)$, source node s, target node t, a relaxation weight $R : V \cup E \longrightarrow \mathbb{N} \setminus \{0\}$ and $(n, e) \in \mathbb{N}^2$, we call H an* (n, e)-relaxed explanation *of (G, s, t) under R if (a) there exists a subgraph G' of G that is obtained by relaxing nodes and/or edges, (b) $\sum_{\{v \in V | v \text{ relaxed in } G'\}} R(v) \leq n$, (c) $\sum_{\{e \in E | e \text{ relaxed in } G'\}} R(e) \leq e$, and (d) H is an explanation subgraph of (G', s, t).*

3.4 Functional significance

As mentioned earlier, often there are no explanation subgraphs with 0 node and 0 edge relaxations. Interestingly, however, the experiments performed here indicate that there is a vast multiplicity (literally 1000s) of explanation subgraphs if we allow a small node and/or edge relaxation. In this context, solutions obtained with large values of node and/or edge relaxations may not be meaningful. Relaxing too many nodes allows the activation status of many nodes to differ from the observed gene expression profile. Similarly, relaxing too many edges amounts to making significant modifications to a curated database of regulatory pathways. None of these are desirable. Indeed, if all nodes or all edges are allowed to be relaxed, an explanation subgraph can always be found. However, this relaxed subgraph may hardly relate to the wet-lab experiment under investigation. Therefore, it makes sense to ask for minimal node and edge relaxations that yield at least one explanation subgraph.

Not surprisingly, increasing node relaxations reduces the requirement of edge relaxations and vice versa. Therefore, this becomes a multi-objective optimization problem, and the only hope is to obtain a set of minimal or Pareto-optimal (n, e) values. Furthermore, since large node and edge relaxations are undesirable, node and edge relaxation values within given bounds are the only interesting ones. The restriction of the number of relaxed nodes/edges to specific bounds was the motivation behind defining a window of relaxation W. This window is given as a pair of intervals, denoted $\langle [n_l, n_u], [e_l, e_u] \rangle$, for node and edge relaxations respectively, shown in Fig. 3.4. A point $(n, e) \in W$ iff $n \in [n_l, n_u]$ and $e \in [e_l, e_u]$.

Figure 3.4: User defined window

Consider the partial order $\sqsubseteq$ on $\mathbb{N} \times \mathbb{N}$ defined by $(n', e') \sqsubseteq (n, e)$ iff $n' \leq n$ and $e' \leq e$. It is said that (n, e) *dominates* (n', e') if $(n', e') \sqsubseteq (n, e)$ and that (n, e) *strictly dominates* (n', e') if $(n', e') \sqsubseteq (n, e)$, but $(n', e') \neq (n, e)$. Given an input graph G with nodes s, t, a relaxation weight R and a relaxation window W, let $Sol(G, s, t, W, R)$ denote the set of $(n, e) \in W$ such that there exists an (n, e)-relaxed explanation of (G, s, t) under R. If $(n, e) \in Sol(G, s, t, W, R)$ but both $(n - 1, e)$ and $(n, e - 1)$ are not in $Sol(G, s, t, W, R)$, it is said that (n, e) is

on the solution curve of (G, s, t, R) within window W. Observe that the set of points on the solution curve forms a Pareto-optimal curve; any point in W that dominates a point on the curve is in $Sol(G, s, t, W, R)$ and any point in W that is strictly dominated by a point on the curve is not in $Sol(G, s, t, W, R)$.

Two reasonable yet essential assumptions need to be made here.

A1: The "golden truth" pathway for the wet-lab experiment under study, henceforth called true explanation subgraph, is present, modulo relaxations and inter-pathway crosstalk, in the input graph $G = (V, E, \lambda, \mu)$.

A2: The true explanation subgraph corresponds to a solution at a Pareto-optimal point (n^*, e^*) within the relaxation window of interest W, for the given relaxation weight function R. It is reasonable to expect $(n^\star, e^\star)$ to be a Pareto-optimal point. Otherwise, there would be an alternative explanation of the microarray data with fewer relaxations than that required for the correct explanation subgraph to provide a plausible explanation.

Definition 3.4.1. Functional significance *Under assumptions A1 and A2, a node v is said to be* functionally significant *in (G, s, t, W, R) if its removal from G leaves no $(n^\star, e^\star)$-relaxed explanation subgraph. In other words, $Sol(G \setminus \{v\}, s, t, \langle [n^\star, n^\star], [e^\star, e^\star] \rangle) = \emptyset$.*

Example

In context to Fig 3.1, a question can be asked: *Does gene D play a functionally significant role in explaining the observed expressions consistently with domain knowledge?* While the precise notion of functional significance will be introduced soon, informally an explanation graph of the observed gene expressions consistent with domain knowledge can be found even on the removal of node D from Fig. 3.1.

It is easy to see from Fig. 3.1 that the answer to the question in the previous paragraph is in the negative. In contrast, if node E or F (or both) is (are) removed, the path $A(+) \rightarrow B(+) \dashv C(-) \dashv D(+) \rightarrow G(+)$ continues to explain the observed gene expressions.

Therefore, it can be assumed that if all gene expression measurements are free of any noise, node D is functionally significant, while E and F are not.

However, even if one gene expression measurement is allowed to be noisy, then D ceases to be functionally significant. Indeed, with D

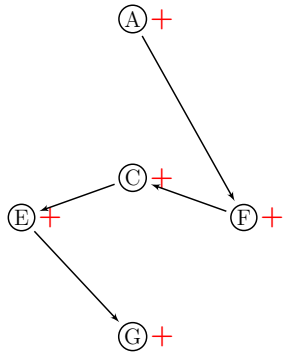


Figure 3.5: Solution with noisy label of C

removed as in Fig. 3.5, the paths $A(+) \to F(+) \to C(+) \to E(+) \to G(+)$ and $A(+) \to B(+)$ explain the observed gene expressions with the (noisy) label of C changed from "-" to "+". Thus the functional significance of a gene can vary depending on the admissible noise.

Unfortunately, Defn 3.4.1 does not yield a practical algorithm for checking the functional significance of a node, due to two reasons. (i) First, for a given experiment, the values of $n^\star$ and $e^\star$ are not known beforehand. (ii) Second, the studies done as part of this work show that there are thousands of explanation subgraphs at each Pareto-optimal point in the window of relaxation of interest. So, even if the value $(n^\star, e^\star)$ were known, it would be practically impossible to examine all $(n^\star, e^\star)$-relaxed explanation subgraphs and identify a common node.

Thus, it was necessary to find a way to decide the functional significance of a node without knowing $(n^\star, e^\star)$ precisely, and without generating all explanation subgraphs corresponding to Pareto-optimal pairs. The following lemma provides a sufficient condition to surmount the hurdles mentioned above.

Lemma 3.4.1. *Suppose* $Sol(G, s, t, W, R) \neq \emptyset$ *and either* $Sol(G \setminus \{i\}, s, t, W, R) = \emptyset$ *or for every* $(n, e) \in Sol(G \setminus \{i\}, s, t, W, R)$*, there exists* $(n', e')) \in Sol(G, s, t, W, R)$ *such that* (n', e') *strictly dominates* (n, e)*. Then node* i *is functionally significant in* (G, s, t, W, R) *under assumptions A1 and A2.*

Informally, Lemma 3.4.1 says that i is functionally significant if the Pareto-optimal curve for $(G \setminus \{i\}, s, t, W, R)$ lies above that for (G, s, t, W, R). This lemma yields a practical algorithm for functional significance checking (see Section 5.4).

Chapter 4

Complexity results

The main question being addressed in this thesis is that, given (i) a graph $G = (V, E, \lambda, \mu)$, (ii) vertices s and t, such that $\lambda(s) = \lambda(t) = (+, +)$, and (iii) certain constraints, does there exist a non-empty explanation subgraph H? This is called the *explanation finding problem* and such a resulting graph is called an explanation or a solution. In this chapter, the computational complexity of this problem is examined.

A crucial feature of the problem is the third point above, that is, the constraints. There are three types of constraints, which are:

1. Relaxation constraints
2. Graph conditions
3. Implication constraints

The first two types of constraints arise from the activity and compatibility conditions of Defn 3.2.1. The activity condition ensures that every s-t path in the solution has at least one active node along it. The compatibility conditions put certain restrictions on the labels of pairs of nodes or edges. These have been explored in the Section 3.3 and Section 3.2 respectively. Their encodings will be discussed later on.

The *Implication* constraints, denoted IC, express that when one edge (or node) is present, then another edge (or node respectively) must also either be present or absent. Formally $IC \subseteq (E \times E) \cup V \times V$ is a set of pairs of edges and nodes from G such that $IC = \eta_{presence} \cup \eta_{absence} \cup \gamma_{presence} \cup \gamma_{absence}$ are implications, where,

(i) $\eta_{presence} \subseteq E \times E$ is a set of ordered pairs of edges. If $(e_i, e_j) \in \eta_{presence}$ we denote $(e_i \implies e_j) = \texttt{true}$.

(ii) $\eta_{absence} \subseteq E \times E$ is a set of ordered pairs of edges. If $(e_i, e_k) \in \eta_{absence}$ we denote $(e_i \implies \neg e_k) = \texttt{true}$.

(iii) $\gamma_{presence} \subseteq V \times V$ is a set of ordered pairs of nodes. If $(v_i, v_j) \in \gamma_{presence}$ we denote $(v_i \implies v_j) = \texttt{true}$.

(iv) $\gamma_{absence} \subseteq V \times V$ is a set of ordered pairs of nodes. If $(v_i, v_k) \in \eta_{absence}$ we denote $(v_i \implies \neg v_k) = \texttt{true}$.

The η are *edge implications*, while γ are *node implications*. Further, the first and third above are called *positive implications*, while the second and fourth above are called *negative implications*. Let IC be a set of implication constraints on G.

These constraints enable the definition of a third *implication condition* on the explanation graph G'; namely, it is required that IC is satisfied by V', E'. For instance$(\forall (e, e') \in \eta_{presence}$, if $e \in E'$, then $e' \in E') \wedge (\forall (e, e') \in \eta_{absence}$, if $e \in E'$, then $e' \notin E')$.

4.1 Significant attributes of the graph

A careful examination of the constraints above shows that there are few essential features in the input graph whose inter-dependencies present different variations resulting in various subproblems. An immediate question is the complexity of these subproblems. Also, whether relaxing any of the conditions could lead to polynomial time variants of the main problem. Before going into the subproblems, we discuss the features in detail.

Positive and negative edge constraints A positive edge constraint is when one edge is present, another edge has to be present, that is, if present they have to appear together. On the other hand, a negative edge constraint is a restriction on at least two edges that when one is present the other cannot be present. We discuss exampls with help of part of the TNFα signaling pathway[1] as shown in Figure 4.1.

[1] https://www.genome.jp/kegg/pathway/hsa/hsa04668.png

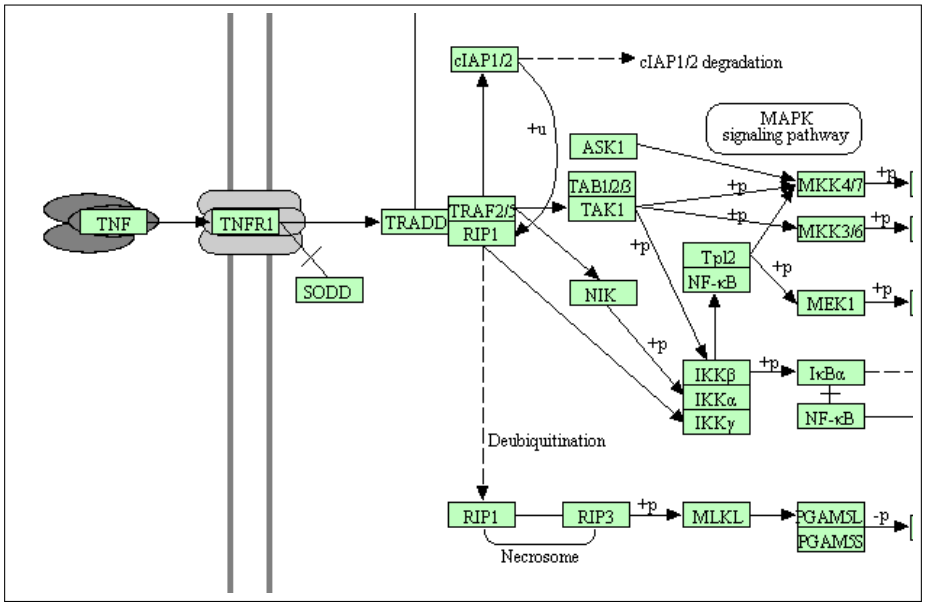


Figure 4.1: Part of NFκB signaling pathway

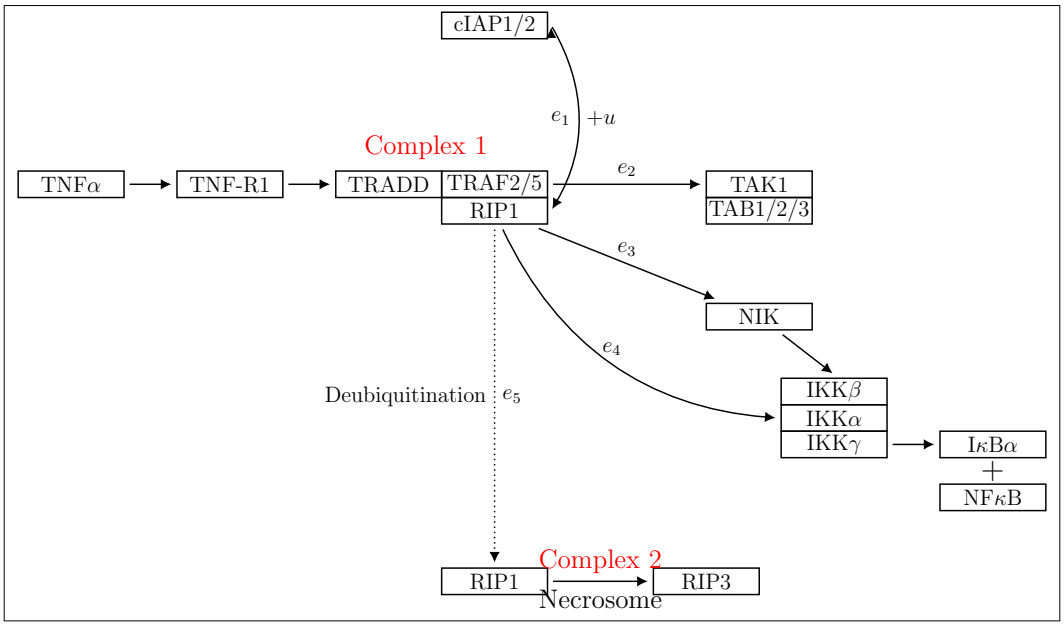


Figure 4.2: Schematic diagram of path from TNFα to IκBα

Figure 4.2 shows a schematic diagram of part of the NFκB signaling pathway. Stimulation of TNF-R1 by TNFα results in interaction or TRADD and RIP1 proteins with the TNF signaling complex. Ubiquitin ligases are recruited

by the complex where they add multiple ubuquitin linkages on RI 1 (Complex 1). This ubiquitinated RIP1 interacts with downstream TAK1 kinase and IKK kinase complexes to ctivate NFκB which results in the downstream activation of many prosurvival genes.

Alternatively, inhibition of RIP1 by some NFκB targets results in de-ubiquitination and formation of the RIP1 necrosome complex (Complex 2). Hence RIP1-based complex 1 and complex 2 are mutually exclusive. Thus edge e_5 occurs at a very different scenario as edges e_2, e_3, and e_4. We have examples of negative edge constraints: $e_3 \implies \neg e_2$, $e_4 \implies \neg e_2$ and $e_5 \implies \neg e_2$,

cIAP's ubiquitination of RIP1 is essential for downstream activation of TAK/TAB, NIK, or IKK complexes which go on to activate NFκB. Thus examples of positive edge constraints are $e_2 \implies e_1$, $e_3 \implies e_1$ and $e_4 \implies e_1$.

Positive and negative node constraints Positive node constraints ensure that two nodes always occur together in the solution. Figure 4.3 example is shown in part of PPAR signaling pathway[2].

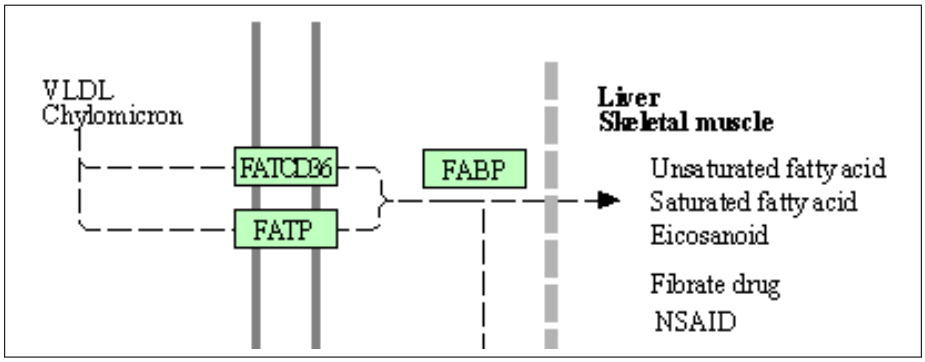


Figure 4.3: Part of PPAR signaling pathway

The example shows two receptors on the cell membrane, FATCD36 and FATP. It was pointed out that both these receptors have to be engaged for FABP to be activated. Here is an example of positive node constraints: FATCD36 $\implies$ FATP and vice versa.

[2] https://www.genome.jp/kegg-bin/show_pathway?hsa03320

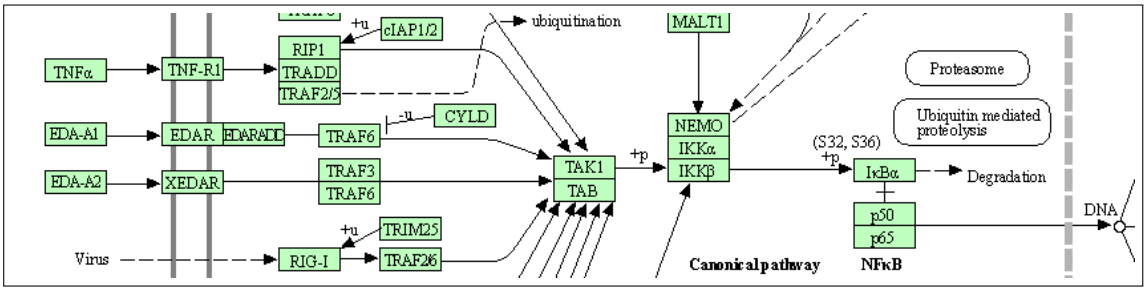


Figure 4.4: Part of NFκB signaling pathway

The snippet shown in Figure 4.5, shows a path "TNFα $\longrightarrow$ TNF-R1 $\longrightarrow$ [RIP1 + TRADD + TRAF2/5] $\longrightarrow$ [TAK1 + TAB] $\longrightarrow$ [NEMO + IKKα + IKKβ] $\longrightarrow$ IκBα". Though this entire path contains only activating edges (phosphorylation is also taken semantically as activating), yet the biological domain experts pointed out that if TNFα is active, then IKBα must be absent or inactive. They constitute the constraints "TNFα $\implies$ $\neg$ IKBα". This is an example of negative node constraint.

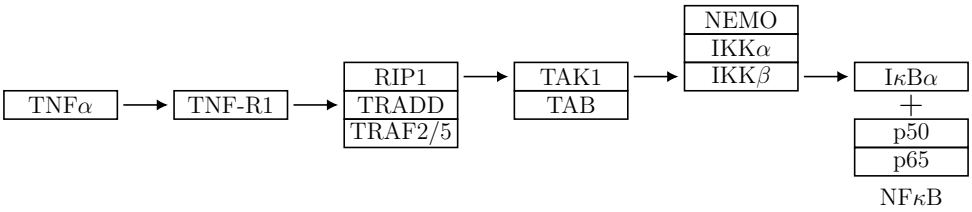


Figure 4.5: Schematic diagram of path from TNFα to IκBα

Inhibition edges present As described in the definition of a graph, the edges have labels that corresponds to different types of functions in the KEGG pathways. Two main types of edges being considered are activation and inhibition. An activation edge has a positive correlation between its source and target nodes where both source and target nodes are either present or absent. An inhibition edge encodes a negative correlation between its source and target nodes. If the source node is active the target node is inhibited and vice versa.

In the setup of our biological experiments, we needed a final node NFκB to be activated. This can only be achieved if the inhibitor of NFκB, IκB, is inhibited. The inhibitor being inhibited allows NFκB node to be active. Thus to inhibit IκB, at least one inhibition edge incoming to IκB must be present. This is a requirement imposed from domain knowledge. Otherwise the solver was free to choose. This gives an example of the constraint that inhibition edges are present.

Positive (+) labels on internal nodes Depending on the data profile used, some nodes are up-regulated. These nodes have a node label +. All such nodes other than source and target nodes are considered as internal nodes with positive (+) label.

Negative (-) labels on internal nodes Similarly, the down-regulated nodes in a data profile have a label −. Such nodes other than source and target are called internal nodes with negative (−) label.

4.2 Explanation subgraph finding problem is NP-complete

Various combinations of the attributes discussed above give rise to different cases. It is now the goal to analyze complexities of these cases. Most of the cases are found to be NP-complete.

Theorem 4.2.1. *Checking the existence of an explanation subgraph, even without relaxations, is* NP*-complete.*

Proof. Follows from proofs of lemmas 4.3.1 through 4.3.6. □

4.3 Variants of the problem

The different cases can be grouped into eight categories that cover all combinations possible with the various constraints. The next section discusses these different cases and gives proofs for each of them.

Before going into the first lemma and its proof, we discuss a small gadget that will be used hereafter. Let this gadget be called A_i. The diagramatic representation is given in Figure 4.6. There will be one for each clause c of the formula. It is constructed using total of six nodes. Three nodes are denoted by l_i^j, that is i-th literal of the j-th clause. For example, say, $l_1^1 = x_1$ or $\neg x_1$. Two edges from nodes l_i^c and $l_{i'}^c$ converge on node l_u^c. Similarly two edges from $l_{i'}^c$ and $l_{i''}^c$ converge on the node l_v^c. Finaly edges from l_u^c and l_v^c converge on the nodes l_w^c.

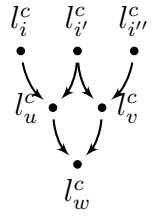


Figure 4.6: Gadget A_i for clause c

Claim 1. *In Gadget A_c, for clause c, l_i^c, $l_{i'}^c$ or $l_{i''}^c$ is in active form if and only if final l_w^c node is active.*

Proof. If any of the nodes l_i^c, $l_{i'}^c$ or $l_{i''}^c$ is in active form, then they will, by compatibility condition, make either l_u^c or l_v^c, or both active. If either of these nodes is active, then by compatibility condition, node l_w^c is also active. On the other hand, if node l_w^c is active, then by compatibility conditions, either l_u^c or l_v^c, or both must be active. To make l_u^c active either l_i^c or $l_{i'}^c$ or both must be active. Similary, to make l_v^c active either $l_{i'}^c$ or $l_{i''}^c$ or both must be active. □

Case *i*: With negative edge constraints

Here we state the first lemma and its proof that will eventually prove NP-completeness of the existence of an explanation subgraph problem. This case considers presence of negative edge constraints in the original network.

Lemma 4.3.1. *Checking the existence of an explanation subgraph in presence of negative edge constraints is* NP*-complete.*

Proof. It is easy to see that the problem is in NP, since the explanation subgraph can be simply guessed, and the solution can be checked in polynomial time.

For proving NP-hardness, 3-SAT is reduced to the current problem. Let φ be an instance of 3-SAT in CNF, with ℓ variables $x_1, \ldots, x_\ell$ and m clauses. The variables and clauses need to be encoded in the form of a graph. Then, an explanation subgraph of the graph exists if and only if the formula is satisfiable.

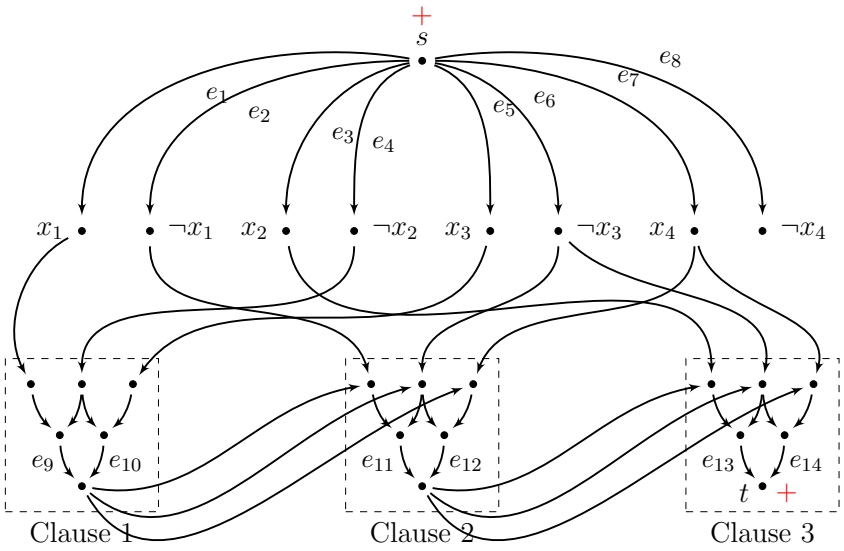


Figure 4.7: Construction for reduction from 3-SAT

We discuss the construction here. For each variable, two nodes denote its positive and negative forms, such as x_1, $\neg x_1$, x_2, $\neg x_2$, x_3, $\neg x_3$, x_4 and $\neg x_4$, as depicted in Figure 4.7. For each clause c, a gadget A_i is constructed as described previously. Now, if $l_1^1 = x_1$, or respectively $\neg x_1$, then an edge is added from x_1 (or $\neg x_1$) to l_1^1.

Each clause of the original instance is replicated by edges from a variable or its negation as appropriate, which finally converge at l_w^c, emulating the disjunctions within each clause. Finally, from each l_w^c, edges are added to the nodes l_i^c, $l_{i'}^c$ and $l_{i''}^c$ of the next clause, except the last clause. In the final clause the node l_w^c is same as target node t. Recall that the target is also labeled $+$.

Additionally, there is the requirement that presence of certain edges disallows presence of some other edges. These restrictions are used to make sure that the variable and its negation both are not picked. Also, inside the gadget A_c for clause c, if the edge outgoing from the node l_u^c is absent, then the edge outgoing from l_v^c must be present and vice versa. These restrictions make sure that the each l_w^c node is true, that is, each clause evaluates to true.

$$
\begin{array}{llll}
e_1 \implies \neg e_2 & e_2 \implies \neg e_1 & \qquad \neg e_9 \implies e_{10} & \neg e_{10} \implies e_9 \\
e_3 \implies \neg e_4 & e_4 \implies \neg e_3 & \neg e_{11} \implies e_{12} & \neg e_{12} \implies e_{11} \\
e_5 \implies \neg e_6 & e_6 \implies \neg e_5 & \neg e_{13} \implies e_{14} & \neg e_{14} \implies e_{13} \\
e_7 \implies \neg e_8 & e_8 \implies \neg e_7 &
\end{array}
$$

Table 4.1: Negative edge implications

The claim is that an explanation subgraph from s to t exists iff the formula is satisfiable. In one direction, if there is an explanation from s to t, the additional node l_w^c at clause c for every clause must be active and each variable is assigned a unique value. Further, to make this active, by the proof of Claim 1, one of the literals in that clause gadget must be active. In turn to make that literal active, node corresponding to the literal should be active and this gives the satisfying assignment. Conversely, if the formula is satisfiable, then the satisfying assignment defines an explanation subgraph. This completes the proof of NP-hardness. An example is the formula $\varphi = (x_1 \vee \neg x_2 \vee x_3) \wedge (\neg x_3 \vee x_4 \vee \neg x_1) \wedge (x_2 \vee \neg x_3 \vee x_4)$, whose graph is shown in Figure 4.7 with a source s and target t. The negative edge implications for the example given are shown in Table 4.1. □

Case *ii*: With negative node constraints

Here we present the second lemma and its proof for the variation that negative node constraints are present in the master network.

Lemma 4.3.2. *Checking the existence of an explanation subgraph in presence of negative node constraints is* NP*-complete.*

Proof. It is easy to see that the problem is in NP, since the explanation subgraph can simply be guessed and the solution can be checked in polynomial time.

For proving NP-hardness, 3-SAT is reduced to the current problem. Similarly as before, let φ be an instance of 3-SAT in CNF, with ℓ variables $x_1, \ldots, x_\ell$ and m clauses. The variables and clauses need to be encoded in the form of a graph. Then, an explanation subgraph of the graph exists if and only if the formula is satisfiable.

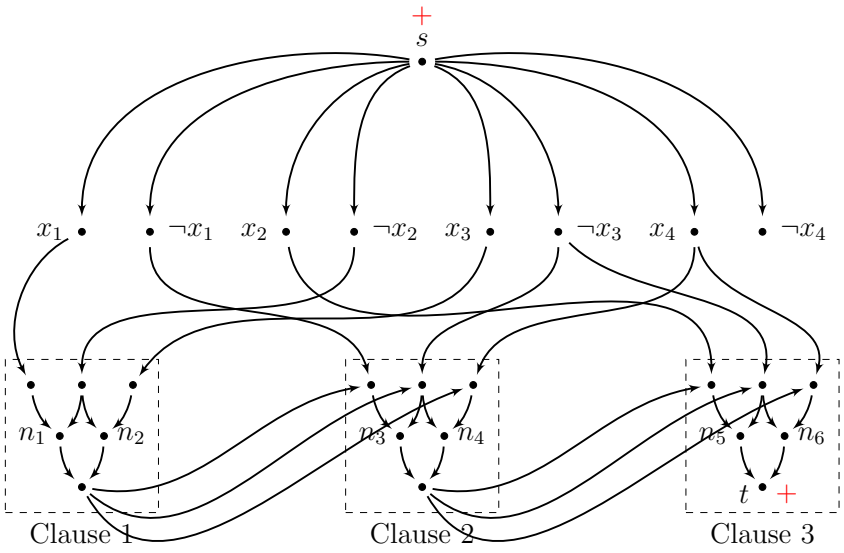


Figure 4.8: Construction for reduction from 3-SAT

For each variable, two nodes denote its positive and negative forms, such as x_1, $\neg x_1$, x_2, $\neg x_2$, x_3, $\neg x_3$, x_4 and $\neg x_4$, as depicted in Figure 4.8. For each clause c, the gadget A_c is constructed as described previously. Rest of the construction is same as that described in Figure 4.7.

Only difference is, instead of naming some edges in gadgets A_i, in this case we name certain nodes in A_i. We name node l_u^c and l_v^c in each of gadget A_i, and out constraints on these nodes, as described in Figure 4.8.

This case permits presence of negative node constraints. Here, presence of a variable node disallows presence of the negation of that variable. Also, within the gadget A_c for clause c, absence of node l_u^c means node l_v^c has to be active, and vice versa.

x_1	$\Longrightarrow$	$\neg\neg x_1$	$\neg x_1$	$\Longrightarrow$	$\neg x_1$	$\neg n_1$	$\Longrightarrow$	n_2	$\neg n_2$	$\Longrightarrow$	n_1
x_2	$\Longrightarrow$	$\neg\neg x_2$	$\neg x_2$	$\Longrightarrow$	$\neg x_2$	$\neg n_3$	$\Longrightarrow$	n_4	$\neg n_4$	$\Longrightarrow$	n_3
x_3	$\Longrightarrow$	$\neg\neg x_3$	$\neg x_3$	$\Longrightarrow$	$\neg x_3$	$\neg n_5$	$\Longrightarrow$	n_6	$\neg n_6$	$\Longrightarrow$	n_5
x_4	$\Longrightarrow$	$\neg\neg x_4$	$\neg x_4$	$\Longrightarrow$	$\neg x_4$						

Table 4.2: Negative node implications

The claim is that an explanation subgraph from s to t exists iff the formula is satisfiable. In one direction, if there is an explanation from s to t, the additional node l_w^c at clause c for every clause must be active and each variable is assigned a unique value. Further, to make this active, by the compatibility condition, one of the literals in that clause gadget must be active. In turn to make that literal active, node corresponding to the literal should be active and this gives the satisfying assignment. Conversely, if the formula is satisfiable, then the satisfying assignment defines an explanation subgraph. This completes the proof of NP-hardness. An example is the formula $\varphi = (x_1 \vee \neg x_2 \vee x_3) \wedge (\neg x_3 \vee x_4 \vee \neg x_1) \wedge (x_2 \vee \neg x_3 \vee x_4)$, whose graph is shown in Figure 4.8 with a source s and target t. The negative edge implications for the example given are shown in Table 4.2. □

Case *iii*: With positive edge constraints and inhibitory edges

The third case allows for presence of positive edge constraints and inhibitory edges in the main network. The corresponding lemma is stated below along with its proof.

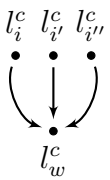


Figure 4.9: Gadget B_c for clause c

Here onwards, for each clause c, a gadget B_c is used which is simpler than gadget A_c. Gadget B_c, as shown in Figure 4.9, is constructed with four nodes, one for each of the three literals that occur in the clause. The nodes are called l_i^c, $l_{i'}^c$ and $l_{i''}^c$. They take their values from the literals in the clause, like, if the literal is positive (resp. negative) in c, i.e., $l_i^c = x_i$ (resp. $= \neg x_i$), then an edge is added from x_i to the node l_i^c, else an edge is added from $\neg x_i$. Nodes l_i^c, $l_{i'}^c$ and $l_{i''}^c$ are all connected to the culminating node l_w^c.

Claim 2. *In Gadget B_c, for clause c, l_i^c, $l_{i'}^c$ or $l_{i''}^c$ is in active form iff l_w^c is in active form.*

Proof. Proof is obvious from the construction and the compatiblity conditions. □

Another small gadget, as shown in Figure 4.10, is used only for the next proof. Like in the previous proofs, for each variable one node represents its positive form x_i, and another one represents its negative form $\neg x_i$. In the new gadget, C_{x_i}, two inhibiting edges are added from x_i to $\neg x_i$ and from $\neg x_i$ to x_i. Consider the positive edge constraints $e_1 \implies e_3$ and $e_2 \implies e_4$. The gadget C_{x_i} along with the edge constraints ensure that the nodes x_i and its negation $\neg x_i$ are not active at the same time.

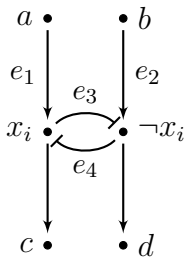


Figure 4.10: Gadget C_{x_i} for clause c

Claim 3. *Gadget C_i for each variable in the formula, along with certain positive edge constraints, ensures that both the variable and its negation are not active simultaneously.*

Proof. Proof is straightforward from the construction and the discussion above. □

Now, we state the lemma for the case with positive edge constraints and inhibitory edges.

Lemma 4.3.3. *Checking the existence of an explanation subgraph in presence of positive edge constraints and inhibitory edges is* NP*-complete.*

Proof. As in the previous cases, it is easy to show that the problem is in NP, since the explanation subgraph can be guessed and the solution can be checked in polynomial time.

The proof of NP-hardness is given next. Again, 3-SAT is reduced to the problem. Let φ be an instance of 3-SAT in CNF, with ℓ variables $x_1, \ldots, x_\ell$ and m clauses. The variables and clauses need to be encoded in the form of a graph. Then, an explanation subgraph of the graph exists if and only if the formula is satisfiable.

For each variable one node represents its positive form x_i, and the other one represents its negative form $\neg x_i$. Activating edges are added from source s to x_i and $\neg x_i$ for all i. In this proof, we use the gadget x_i. for each i, ensure that a variable x_i and its negation $\neg x_i$ are not active at the same time.

For each clause c, the gadget B_c is used and the necessary edges are added as discussed before. Similar to the previous proofs, the l_w^c node from one clause is connected to the l_i^c, $l_{i'}^c$ and $l_{i''}^c$ of the subsequent clause. Finally, the l_w^c node, is indicated as the target node t and labeled $+$. Source s is also labeled $+$.

This case permits presence of positive edge constraints and inhibitory edges. We have already discussed how inhibitory edges are added between a variable and its negation nodes. By putting positive edge constraints we ensure that both a

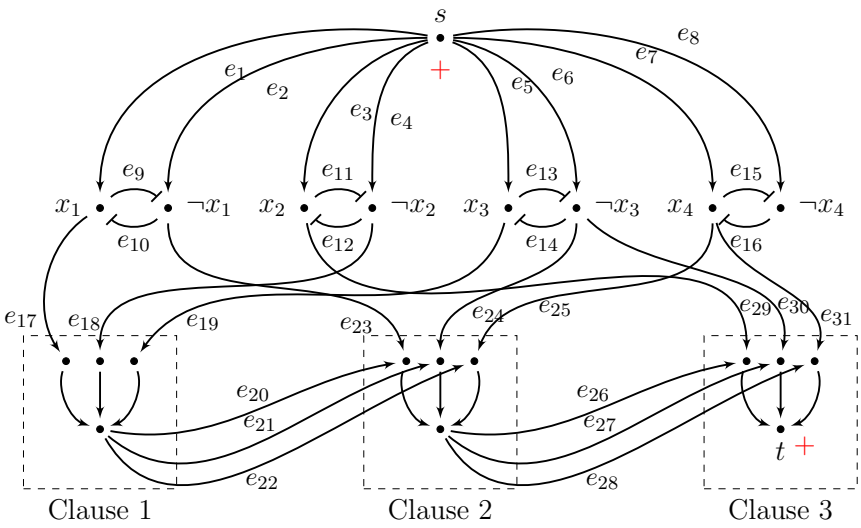


Figure 4.11: Construction for reduction from 3-SAT

$e_1 \implies e_9$	$e_2 \implies e_{10}$	$e_3 \implies e_{11}$	$e_4 \implies e_{12}$
$e_5 \implies e_{13}$	$e_6 \implies e_{14}$	$e_7 \implies e_{15}$	$e_8 \implies e_{16}$
$e_1 \implies e_{17}$	$e_2 \implies e_{23}$	$e_3 \implies e_{29}$	$e_4 \implies e_{18}$
$e_5 \implies e_{19}$	$e_6 \implies e_{24}$	$e_6 \implies e_{30}$	$e_7 \implies e_{31}$
$e_{20} \implies e_{23}$	$e_{21} \implies e_{24}$	$e_{22} \implies e_{25}$	$e_{23} \implies e_{20}$
$e_{24} \implies e_{21}$	$e_{25} \implies e_{22}$	$e_{26} \implies e_{29}$	$e_{27} \implies e_{30}$
$e_{28} \implies e_{31}$	$e_{29} \implies e_{26}$	$e_{30} \implies e_{27}$	$e_{31} \implies e_{28}$

Table 4.3: Positive edge implications

node and its negation node are never picked together. For example, the constraint $e_1 \implies e_9$. Presence of activating edge e_1 ensures that the inhibitory edge e_9 is active and hence inhibiting $\neg x_1$ is absent.

Few other positive edge constraints are of the form $e_1 \implies e_{17}$. This ensures that if variable x_1 is active then its corresponding l_i^c node is also active. In turn node l_w^c too becomes active. Finally, constraints of the form $e_{20} \implies e_{23}$ ensures that if the l_w^c node is active and the edge e_{20} edge is present, then the corresponding x_i node in the subsequent clause is also active.

From this construction, it can be said that an explanation subgraph from s to t exists if and only if the formula is satisfiable. In one direction, if there is an explanation from s to t, the additional node l_w^c at clause c for every clause must be active, and each variable is assigned a unique value. Further, to make this active, by the compatibility condition, one of the literals in that clause gadget B_i

must be active. In turn, to make that literal active, node corresponding to the literal should be active, and this gives the satisfying assignment. Conversely, if the formula is satisfiable, then the satisfying assignment defines an explanation subgraph. These conditions complete the proof of NP-hardness.

The same example formula as before is reconsidered $\varphi = (x_1 \vee \neg x_2 \vee x_3) \wedge (\neg x_3 \vee x_4 \vee \neg x_1) \wedge (x_2 \vee \neg x_3 \vee x_4)$, whose graph construction is shown in Fig. 4.11 with a source node s and target node t. The positive edge implications for the example discussed are shown in Table 4.3. □

Case *iv*: With positive node constraints and inhibitory edges

The fourth case permits presence of positive node constraints and inhibitory edges in the master network. The corresponding lemma and its proof is discussed in this section.

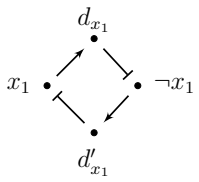


Figure 4.12: Gadget D_{x_i} for variable x_i

Before going into the lemma, we discuss a small gadget that is used in this proof only. We call this Gadget D_{x_i}, that occurs one for every variable in the CNF formula. It is shown in Figure 4.12.

This gadget consists of two activating edges, one from x_i to d_{x_i} and the other from $\neg x_i$ to d'_{x_i}. It also has two inhibiting edges from d_{x_i} to $\neg x_i$, and from d'_{x_i} to x_i. This gadget ensures that a variable and its negation are not active simultaneously.

Claim 4. *For variable x_i, gadget D_{x_i} ensures that nodes x_i and $\neg x_i$ are not active at the same time.*

Proof. The proof is straightforward from construction and compatibility conditions. □

Now we state the lemma when positive node constraints and inhibitory edges are present in the master network.

Lemma 4.3.4. *Checking the existence of an explanation subgraph in presence of positive node constraints and inhibitory edges is* NP*-complete.*

Proof. As before, it is easy to show that the problem is in NP since we can simply guess the solution and chenck it in polynomial time.

For the proof of NP-hardness, 3-SAT is reduced to the problem being discussed. Let φ be an instance of 3-SAT in CNF, with ℓ variables $x_1, \ldots, x_\ell$ and m clauses. The variables and clauses need to be encoded in the form of a graph. Then, an explanation subgraph of the graph exists if and only if the formula is satisfiable.

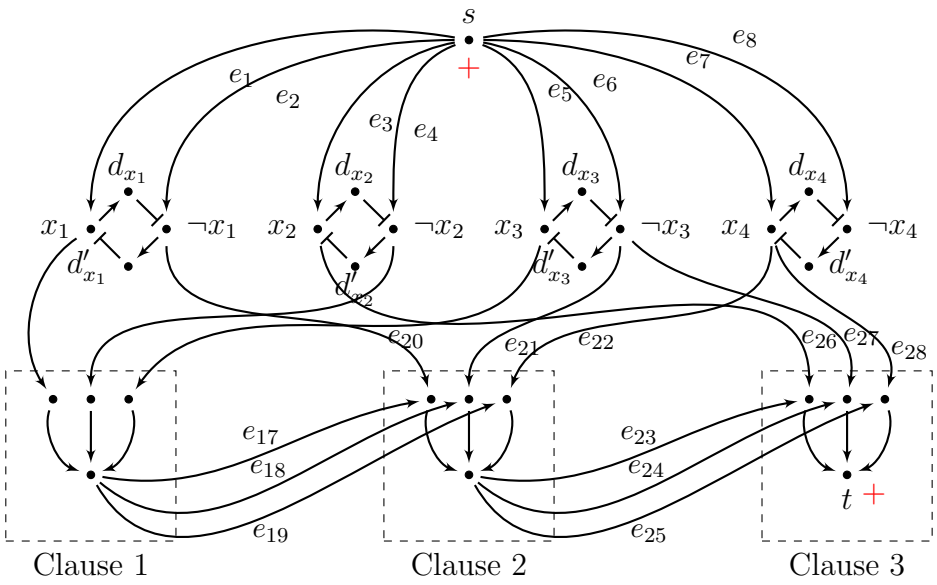


Figure 4.13: Construction for reduction from 3-SAT

For each variable x_i, at first, a gadget D_{x_i} is constructed and necessary edges are added. Activating edges from source s to x_i and $\neg x_i$ for all i are added.

Rest of the construction is same as the previous case involving gadgets B_c. Source node s and target node t are labeled $+$.

$$\begin{array}{llll} x_1 \implies d_{x_1} & \neg x_1 \implies d'_{x_1} \\ x_2 \implies d_{x_2} & \neg x_2 \implies d'_{x_2} \\ x_3 \implies d_{x_3} & \neg x_3 \implies d'_{x_3} \\ x_4 \implies d_{x_4} & \neg x_4 \implies d'_{x_4} \end{array}$$

Table 4.4: Positive node implications

From this construction, we can say that an explanation subgraph from s to t exists if and only if the formula is satisfiable. In one direction, if there is an explanation from s to t, the additional node l_c^c at clause c for every clause must be active, and each variable is assigned a unique value. Further, to make this active, by the compatibility condition, one of the literals in that clause gadget must be active. In turn, to make that literal active, node corresponding to the literal should be active, and this gives the satisfying assignment. Conversely, if the formula is satisfiable, then the satisfying assignment defines an explanation subgraph. These conditions complete the proof of NP-hardness.

In the old example formula is $\varphi = (x_1 \vee \neg x_2 \vee x_3) \wedge (\neg x_3 \vee x_4 \vee \neg x_1) \wedge (x_2 \vee \neg x_3 \vee x_4)$. Its graph construction is shown in Fig 4.13 with a source node s and target node t. The positive edge implications for the example discussed are shown in Table 4.4. □

Case *v*: With positive internal node labels and inhibitory edges

This case allows positive internal node labels along with inhibitory edges. The gadget used in the proof of the lemma is discussed here.

For each variable x_i, first, a gadget E_i of six nodes is constructed, which is depicted in Fig. 4.14, There are three nodes for variable x_i (which are called a_i, b_i, d_i) and three nodes for $\neg x_i$ (which are named na_i, nb_i, nd_i). Activating edges are added from source s to a_i and na_i for all i. Another four activating edges are added from a_i to b_i, b_i to d_i, na_i to nb_i and nb_i to d_i and four inhibiting edges are added from a_i to nb_i, na_i to b_i, b_i to nd_i and nb_i to nd_i. Finally, both nodes d_i and nd_i get label + and activating edges are added from both d_i and nd_i to target t (for each i).

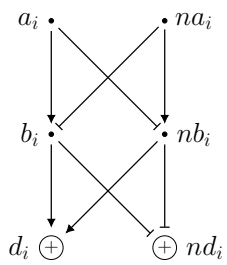


Figure 4.14: Gadget E_i

Claim 5. *This gadget E_i necessarily ensures that a variable x_i and its negation are not active at the same time.*

Proof. Suppose, both b_i and nb_i are activated. Now, d_i is labeled + which means it is activated either by b_i or by nb_i. If b_i is activated by a_i, then a_i inhibits nb_i. And if nb_i is activated by na_i, then na_i inhibits b_i. Thus both b_i and nb_i cannot be active at the same time. Proof by contradiction. It can be similary shown that both b_i and nb_i cannot be inhibited simultaneously. □

Here we state the lemma.

Lemma 4.3.5. *Checking the existence of an explanation subgraph in presence of positive internal node labels and inhibitory edges is* NP*-complete.*

Proof. As in the previous proofs, the problem is in NP since we can simply guess a solution and check it in polynomial time. To prove NP-hardness, the 3-SAT problem can be reduced to the current problem. Let φ be an instance of 3-SAT in CNF, with ℓ variables $x_1, \ldots, x_\ell$ and m clauses. The variables and clauses are encoded in the form of a node- and edge-labeled graph such that an explanation subgraph exists if and only if the formula is satisfiable.

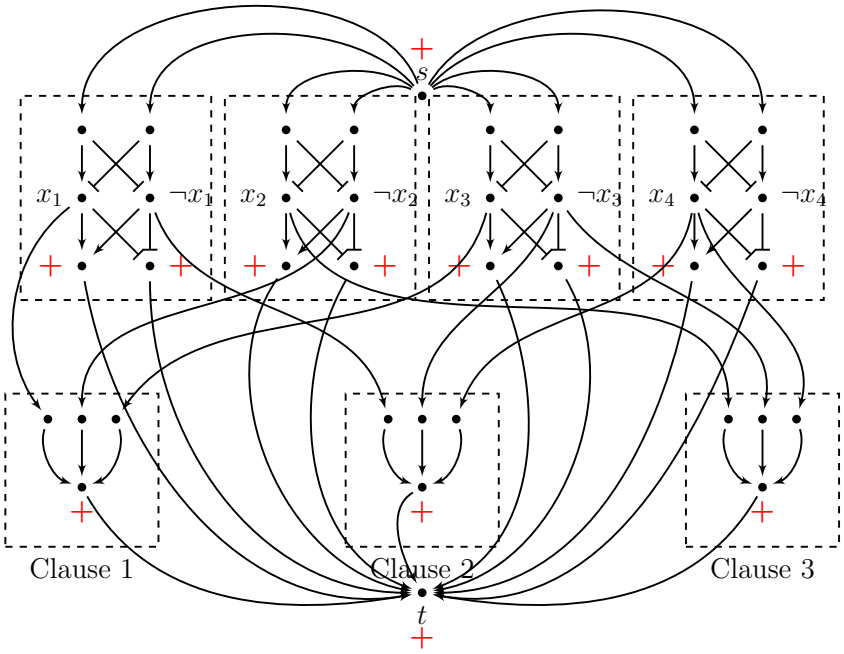


Figure 4.15: Construction for reduction from 3-SAT

For each clause, c, a gadget B_c is constructed with four nodes: one for each of the three literals that occurs in the clause.

Remaining of the construction is as shown in Figure 4.15. From this construction, it can be said that an explanation subgraph from s to t exists if and only if the CNF formula is satisfiable. In one direction, if there is an explanation from s to t, the additional node l_c^c at clause c for every clause must be active, and each variable is assigned a unique value. Further, to make this active, by the compatibility condition, one of the literals in that clause gadget must be active. In turn, to make that literal active, node corresponding to the literal should be active, and this gives the satisfying assignment. Conversely, if the formula is satisfiable, then the satisfying assignment defines an explanation subgraph. This step completes the proof of NP-hardness. As an example consider the formula $\varphi = (x_1 \vee \neg x_2 \vee x_3) \wedge (\neg x_3 \vee x_4 \vee \neg x_1) \wedge (x_2 \vee \neg x_3 \vee x_4)$ below, whose graph construction is shown in Fig. 4.15 with a source node s and target node t. □

Case *vi*: With negative internal node labels and inhibitory edges

We now discuss the case that allows internal nodes with negative labels and inhibitory edges in the master network. The gadget used for proof of this lemma is a small modification over gadget E_i. We call this gadget F_i.

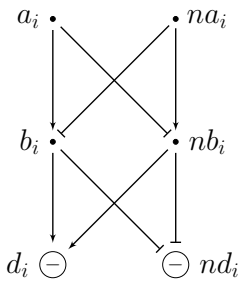


Figure 4.16: Gadget F_i

For each variable x_i, first, a gadget F_i of six nodes is constructed similar to the previous proof, as depicted in Figure 4.16. This gadget F_i necessarily ensures that a variable x_i and its negation are not active at the same time. In this case, d_i and nd_i have negative labels.

Claim 6. *The gadget F_i ensures that a variable b_i and its negation are not active at the same time. In this case, d_i and nd_i have negative labels.*

Proof. The proof is analogous to the proof for the previous gadget E_i. □

We state the main lemma for this case.

Lemma 4.3.6. *Checking the existence of an explanation subgraph in presence of negative internal node labels and inhibitory edges is* NP*-complete.*

Proof. The problem is in NP since we can simply guess a solution and check it in polynomial time.

The 3-SAT problem can be reduced to the problem being addressed here. Let φ be an instance of 3-SAT in CNF, with ℓ variables $x_1, \ldots, x_\ell$ and m clauses. The variables and clauses need to be encoded in the form of a graph. Then it can be said that an explanation subgraph of the graph exists if and only if the formula is satisfiable.

For each clause c, another gadget B_c is constructed with four nodes as described previously, and associated edges are added. Each clause of the original instance is replicated by edges from a variable or its negation as appropriate, which eventually converge at l_w^c emulating the disjunctions within each clause. In this case the nodes

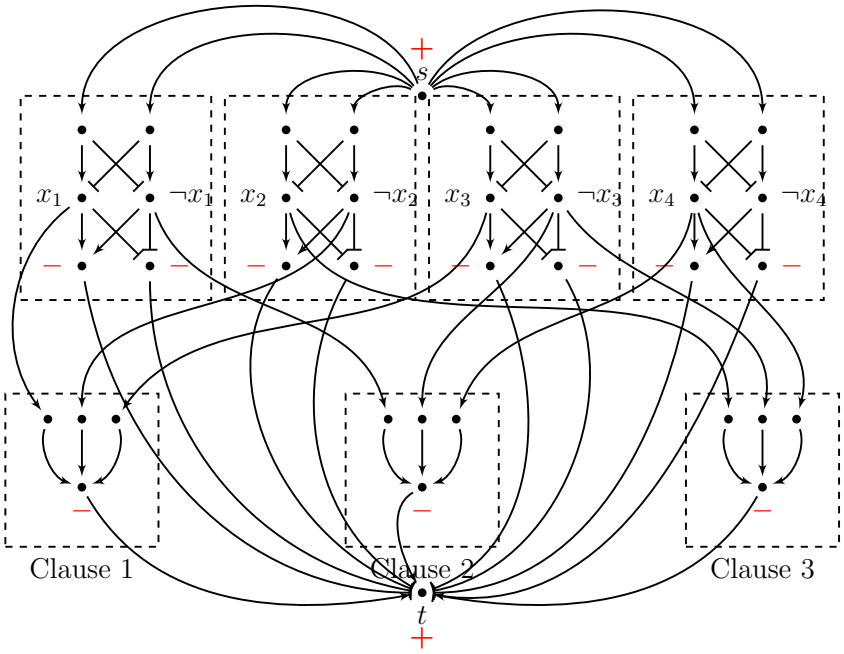


Figure 4.17: Construction for reduction from 3-SAT

l_w^c in each claused is labeled $-$. Finally from each l_w^c node an edge is added to the target node t. Recall that the source node s and target node t is labeled $+$.

From this construction, it can be said that an explanation subgraph from s to t exists if and only if the formula is satisfiable. In one direction, if there is an explanation from s to t, the additional node l_c^c at clause c for every clause must be active, and each variable is assigned a unique value. Further, to make this active, by the compatibility condition, one of the literals in that clause gadget must be active. In turn, to make that literal active, node corresponding to the literal should be active, and this gives the satisfying assignment. Conversely, if the formula is satisfiable, then the satisfying assignment defines an explanation subgraph. These conditions complete the proof of NP-hardness. As an example consider the earlier formula $\varphi = (x_1 \vee \neg x_2 \vee x_3) \wedge (\neg x_3 \vee x_4 \vee \neg x_1) \wedge (x_2 \vee \neg x_3 \vee x_4)$ below, whose graph construction is shown in Fig. 4.17 with a source node s and target node t. □

Case *vii*: With only inhibitory edges

Here we discuss the case where inhibitory edges are present and all other conditions mentioned so far are absent.

Lemma 4.3.7. *Checking the existence of an explanation subgraph in presence of only inhibitory edges and all other types of constraints being absent is polytime solvable.*

Proof. Since there are no internal labels, there is no need to respect the consistency constraints. In the absence of any constraints, the problem of finding a subgraph reduces to a graph traversal and reachability problem. This task can be achieved in polynomial time with simple graph traversal algorithms like breadth-first or depth-first search. Hence the proof. □

Case *viii*: With no negative edge or node constraints and no inhibition edges.

The final case is in the absence of negative node constraints, negative node constraints and inhibition edges. Here we state the lemma.

Lemma 4.3.8. *Checking the existence of an explanation subgraph in absence of negative edges or node constraints and absence of inhibition edges is polytime solvable.*

Proof. Since there are no internal negative labels or negative constraints, and there is no need to have inhibition edges, this presents the most straightforward case of the subgraph extraction problem. Only positive constraints and positive labels can, but need not be present. The problem of finding a subgraph thus reduces to a graph traversal and reachability problem, keeping track of only positive labels and positive constraints. This goal can be achieved in polynomial time with simple modifications on graph traversal algorithms like breadth-first or depth-first search. Hence the proof. □

We have seen a total of eight cases which cover all combinations of the different constraints. Here we summarize the results in a table. The interpretation of the column headers is as follows.

- **-ve EC**: negative edge constraints, i.e., if edge e_1 is present, edge e_2 must be absent
- **-ve NC**: negative node constraints, i.e., if node n_1 is active, node n_2 must be inhibited or absent
- **+ve EC**: positive edge constraints, i.e., if edge e_1 is present, edge e_2 must be present
- **+ve NC**: positive node constraints, i.e., if node n_1 is active, node n_2 must be active

- **Inh edges**: Inhibitory edges present
- **Labels (+)**: Up-regulated internal (non-source non-target) nodes present
- **Labels (-)**: Down-regulated internal (non-source non-target) nodes present
- **Complexity**: The complexity classes NP-complete (NPC), polytime (P).

	-ve EC	-ve NC	+ve EC	+ve NC	Inh	Labels (+)	Labels (-)	Complexity
i	Y	-	-	-	-	-	-	NPC
ii	-	Y	-	-	-	-	-	NPC
iii	-	-	Y	-	Y	-	-	NPC
iv	-	-	-	Y	Y	-	-	NPC
v	-	-	-	-	Y	Y	-	NPC
vi	-	-	-	-	Y	-	Y	NPC
vii	N	N	N	N	Y	N	N	P
viii	N	N	-	-	N	-	-	P

Table 4.5: Complexity results for all variants: the column headings are the variants, where EC, NC are edge and node implication constraints respectively, Inh are inhibition edges, Labels (+), Labels (-) refer to positive and negative internal labels respectively. Y means present, N is absent and - means either.

In Table 4.5 entries, a 'Y' denotes the presence of the type of entity mentioned in the column header, 'N' denotes the absence of it, while '-' means presence or absence does not matter.

4.4 Checking the functional significance of a gene is coNP-complete

Here is the discussion that shows that deciding the existence of an explanation subgraph is NP-complete, and that of checking the functional significance of a gene is coNP-complete.

For each of these combinations of constraints, the question is if hardness is preserved. In the presence of inhibitor edges and positive implication constraints, internal nodes do not need to be labeled + in G to obtain coNP-hardness. On the other hand, if there are no inhibition edges, and neither negative implication constraints nor internal labels are allowed, then Ptime algorithms can be obtained. Table 4.5 summarizes the comprehensive set of results. The proofs of the different variants will be presented accordingly.

If the relaxation window W is fixed, i.e., W is not part of the input to a decision procedure, the following results are obtained for every relaxation weight function R.

Theorem 4.4.1. *For every* $(n, e) \in \mathbb{N}^2$,

1. *Checking the existence of an* (n, e)*-relaxed explanation subgraph is* NP*-complete.*
2. *Checking functional significance in* $W = \langle n, e \rangle$ *is* coNP *complete.*

Proof. Recall that every relaxation weight function assigns a natural number to each node and edge. Therefore, the proof of the first part follows from that of Theorem 4.2.1 Lemma 4.3.5 with a simple modification: the gadget for each variable and clause is replicated $n+e+1$ times so that even if total node relaxation of n nodes and total edge relaxation of e edges are relaxed, finding an explanation subgraph would require setting each variable in a way that all clauses are satisfied.

For the second part, the construction in the proof of Theorem 4.2.1 is modified by adding a special node n_i, where i is the node whose significance is to be checked.

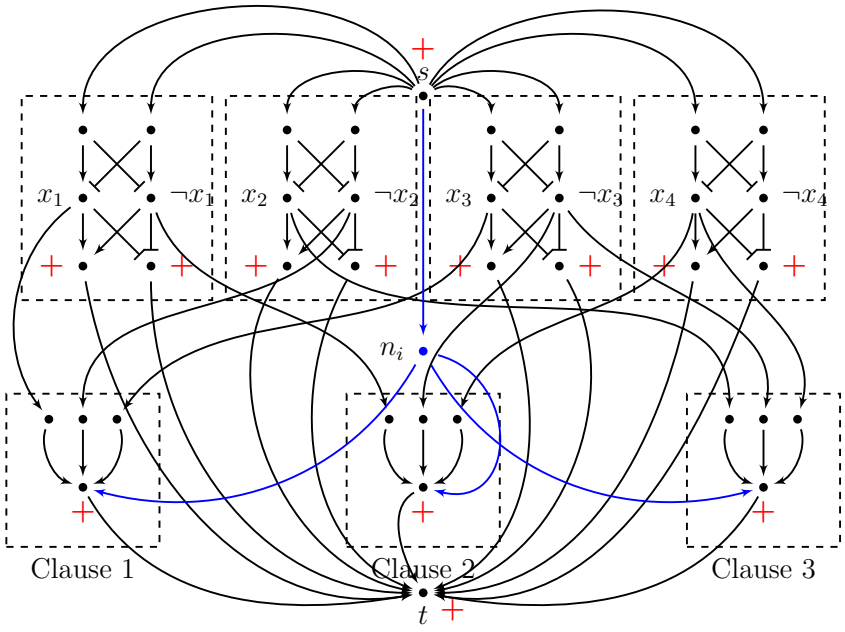


Figure 4.18: Construction for Proof of Thm 4.4.1

Edges are added from the source s to n_i and from n_i to each node l_c^c for each clause c. This construction is shown in Fig. 4.18. Node n_i and edges added on top of the previous construction are shown in blue. Let G' denotes the resulting graph.

Note that in G', there is a path from s to n_i to l_c^c (for each clause c) to t. Thus, with (0,0) relaxation, there is an explanation. However, on the removal of node n_i, there is an explanation with no relaxations iff φ is satisfiable. In other words, the solution curve shifts, i.e., i is functionally significant in G iff φ is unsatisfiable. □

We modify Table 4.5 to summarize these results.

	-ve EC	-ve NC	+ve EC	+ve NC	Inh	Labels (+)	Labels (-)	Complexity
i	Y	-	-	-	-	-	-	coNPC
ii	-	Y	-	-	-	-	-	coNPC
iii	-	-	Y	-	Y	-	-	coNPC
iv	-	-	-	Y	Y	-	-	coNPC
v	-	-	-	-	Y	Y	-	coNPC
vi	-	-	-	-	Y	-	Y	coNPC
vii	N	N	N	N	Y	N	N	P
viii	N	N	-	-	N	-	-	P

Table 4.6: Complexity results for all variants: the column headings are the variants, where EC, NC are edge and node implication constraints respectively, Inh are inhibition edges, Labels (+), Labels (-) refer to positive and negative internal labels respectively. Y means present, N is absent and - means either.

In Table 4.6 entries, a 'Y' denotes the presence of the type of entity mentioned in the column header, 'N' denotes the absence of it, while '-' means presence or absence does not matter.

4.5 Complexity when relaxation window is part of input

If the relaxation window is part of the input, functional significance checking turns out to be harder.

Theorem 4.5.1. *If the relaxation window is part of the input, functional significance checking is Δ_2^P-hard and is contained in Π_2^P.*

Proof. A reduction from the Δ_2^P-complete problem [111] is shown here. Given a satisfiable CNF formula φ and a linear ordering of $x_1 \prec \ldots x_n$ in φ, does the lexicographically largest satisfying assignment of φ have its least significant bit $x_1 = 1$?.

To reduce this to the functional significance checking problem, consider the same construction as in the proof of NP-hardness above, but with the following modification.

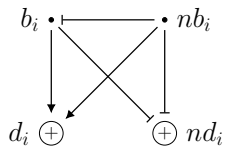


Figure 4.19: Gadget G_i

The gadget in Fig. 4.14 is modified as shown in Fig. 4.19 so that nodes a_i, na_i are removed and so are all edges coming in and out of them. The modified diagram can be seen in Fig. 4.20. An inhibiting edge is added from the source s to each nb_i, and an inhibiting edge is added from each nb_i to the corresponding node b_i.

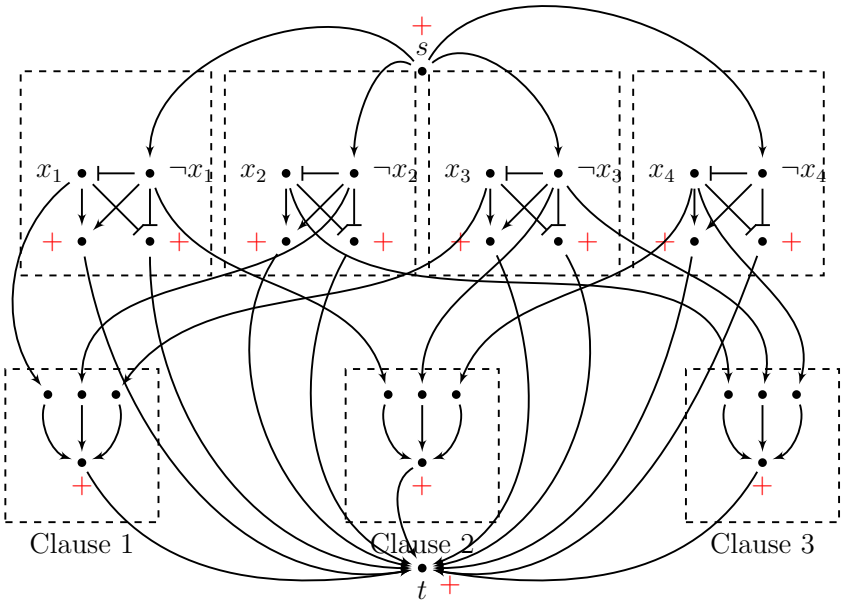


Figure 4.20: Construction for reduction from 3-SAT

Let G be the resulting graph. Let R be the relaxation weight function that assigns weight 2^{i-1} to the inhibiting edge from s to nb_i and 2^n to all other edges. R also assigns weight 2^n to all nodes. Finally, the question is asked if b_1 is functionally significant in (G, s, t, W, R), where $W = \langle [0, 0], [0, 2^n - 1] \rangle$. Note that the size of (G, s, t, W, R) is polynomial in $|\varphi|$. Additionally, the choice of W disallows relaxation of any node and also of any edge other than those from s to some nb_i. The topologically largest satisfying assignment of φ corresponds to an explanation graph with the smallest edge relaxation noise. If this explanation includes b_1, then removing b from G disallows this explanation. This proves Δ_2^P-hardness of functional significance checking.

Containment in Π_2^P is easy to see. We encode the problem as: for all (n', e')-relaxed solutions without the actor, there is an (n, e)-solution with the actor, where (n', e') strictly dominates (n, e) and both are within relaxation bounds. Since n, e, n', e' are integers within given relaxation bounds, the quantifier free part has a polynomial sized propositional encoding. □

The problem of *counting explanation subgraphs* corresponds to #SAT, which is widely believed to be beyond the polynomial hierarchy (by Toda's theorem [112]). Thus, unless long-standing complexity-theory conjectures are falsified, checking functional significance (in Π_2^P) is easier than counting explanations.

Chapter 5

Methods

Given a network of signal transduction and/or gene regulatory pathways, finding potentially "important" actors and interactions in the context of a specific biological experiment is significant in many ways. From a biological perspective, once a few such actors and interactions are identified, if possible, those can be disabled/de-activated using suitable inhibitors. The outcome of such an experiment will show a change if the experiment is indeed affected by this intervention. If so, this will provide evidence in favor of the importance of the actor/interaction in the biological process under study. Such evidence is invaluable in understanding parts of the pathway(s) that are responsible for the biological process, and also for therapeutic interventions, whenever appropriate.

From a computational perspective, identifying key actors and interactions allows for checking if the explanation subgraphs are "sensitive", i.e., change significantly in a graph-theoretic or constraint-theoretic sense, to the removal of identified actors and interactions from the network. If so, one can infer that the actors and interactions likely serve as vital links in the chain of regulatory mechanisms that connect the stimulus to the observed phenotype. This knowledge can also be used as a computational filter to narrow down the set of potential actors and interactions whose roles deserve investigation via wet-lab experiments.

The SAT solver often generates a large number of solutions to the explanation subgraph finding problem. Then it becomes impractical to manually examine each explanation subgraph and identify common key actors and interactions. It would, therefore, be desirable to identify key actors without necessarily generating all explanation subgraphs. This work proposes a three-step approach as a solution. Recall that the explanation subgraph problem requires as inputs: a labeled graph G representing the entire collection of pathways (in KEGG database, for example), binarized differential gene expression levels arising from a biological experiment, a source node s, a target node t, and bounds on the node and edge relaxations. With these inputs, here are the steps to follow.

Step 1: In the first step, a parameter P (> 0) is used, and a subgraph G_P of G is identified such that every path of length bounded by P from s to t in G is also present in G_P. Moreover, every node and edge in G_P must participate in a path from s to t, of length bounded by P. The parameter P is obtained as input from domain experts and represents an upper bound on the length of the shortest path from s to t in which a key actor or interaction is expected to appear.

Step 2: In the second step, nodes/edges in G_P are ranked to serve as a first level filter for identifying actors and interactions with an essential role in the regulatory mechanism from s to t as encoded by G_P. For reasons of computational efficiency, in this step only the directions of the edges are considered, and not their functionality (like inhibiting, activating, and others).

Step 3: In the third step, the top nodes/edges identified in Step 2 are selected. A constraint solver is used to determine if the exclusion of these nodes/edges significantly affects *the entire set* of explanation subgraphs from s to t in G_P, respecting the user-provided relaxation bounds. This step takes into account the functionality and connectivity of all edges and hence is more computationally intensive. Instead of enumerating all solutions, for this step a Pareto-optimality based approach is used that drastically cuts down the computational resource requirements while still providing strong guarantees.

This chapter elaborates on steps 1 and 2 outlined above. The next chapter will discuss step 3 in detail.

5.1 Merging pathways

As described before, 163 *Homo sapiens* pathways were selected from the KEGG database and were merged into one *master network*. If two nodes have a shared id, they were merged into one node. If two edges between the same pair of nodes had identical annotations/labels on them, they were merged into one edge. If the annotations were different, both edges were retained as parallel edges between the said pair of nodes. In essence, this step takes the union of nodes and edges. Note that no node or edge information was either introduced or removed in this step. Edges between common nodes across pathways open up the opportunity to detect crosstalk. It might also introduce spurious connections across mixed contexts, but those get filtered out in the constraints solving step.

KEGG pathways include many experimentally determined edges that are manually curated. However, they do not represent all biologically relevant interactions.

All edges of all types representing every biological situation are not possible to determine. It becomes even more challenging with dynamics that are context dependent. These variations in curation have to be accounted for a priori for a context independent study. This work tries to make the best use out of this limitation. It proposes to determine the Pareto-optimality curves that may provide solutions to a given biological question without explicitly finding any solution subgraph.

5.2 Step 1: Pruning the graph

Any actor or interaction that plays an essential role in a biological process is likely to appear prominently in some short (but not necessarily shortest) path from s to t. Conversely, if the only paths from s to t with the actor present are long, its influence is likely to be moderated by several other actors along the (long) chain of regulatory mechanisms from s to t. Hence it is necessary to restrict the attention to s-t paths of length bounded by the parameter P, whose value is chosen in consultation with domain experts. An incorrect choice of P can affect the results if a key actor participates only in paths of length exceeding P. If so, the value of P can be increased to accommodate such key actors as well.

Example Fig. 5.1 shows an example of pruning of path of length greater than bound. Suppose Fig. 5.1a is the original graph with the source node as s and the target node as t. Suppose path bound P is 3.

Fig. 5.1b shows that though the shortest distance of t from s is less than bound 3, there might be paths that are of length greater than 3 within this graph. The path shown in red is one such example. As a result of the pruning discussed in this work, the node e that is present only on a path that is of length greater than the bound 3 will get eliminated, and the long path will get removed. The resultant graph, as shown in the final Fig. 5.1c, will retain only the paths that are up to the length of the bound 3.

Given a labeled graph $G = (V, E, \lambda, \mu)$, vertices s and t in V, and a path bound P, Algorithm 1 produces a subgraph $G_P = (V_P, E_P, \lambda_P, \mu_P)$ of G such that the following conditions are satisfied:

(i) every path from s to t of length bounded by P in G is present in G_P

(ii) every node in V_P and every edge in E_P participates in at least one s-t path of length bounded by P

(iii) λ_P and μ_P are the restrictions of λ and μ to V_P and E_P, respectively

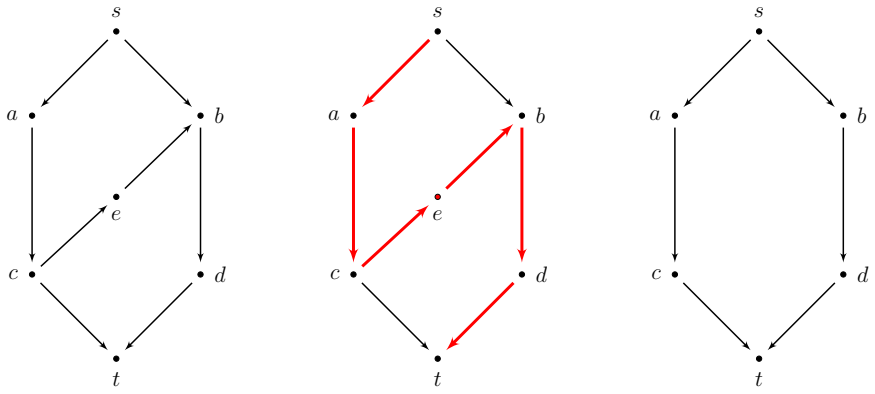


Figure 5.1: (a) shows an example graph with source s and target node t with a reachability bound 3. (b) shows that node e lies only on a path that is of length $>$ given bound 3, that is shown in red. (c) Gives the graph pruned by Algo 1 excluding node e and associated edges. All $s-t$ paths in this graph are of length $\leq$ bound 3. Nodes, like e, that appear only on paths of length $>$ are excluded.

The subgraph G is called the *pruned reachable graph for path bound* P.

Algorithm 1 uses two one-dimensional matrices called DistS and DistT. For each vertex v in V, it first computes the length of the shortest path from s to v following forward edges and stores it in $\mathsf{DistS}[v]$. Vertices with the shortest distance from s exceeding P are not of interest, and hence, $\mathsf{DistS}[v]$ is not updated for those. Similarly, the algorithm computes the length of the shortest path from t to v following backward edges and stores this distance in $\mathsf{DistT}[v]$. As an optimization, this computation is restricted to vertices v that have $\mathsf{DistS}[v] \leq P$, since it is sufficient to retain only those vertices v in V_P that have $(\mathsf{DistS}[v] + \mathsf{DistT}[v]) \leq P$. The set E_P is exactly the set of edges (u, v) that have both u and v in V_P, and in addition $\mathsf{DistS}[u] + 1 + \mathsf{DistT}[v] \leq P$. Note that the last condition for (u, v) being in E_P implies that G_P is not necessarily the subgraph of G induced by V_P.

Lemma 5.2.1. *The graph $G_P = (V_P, E_P, \lambda_P, \mu_P)$ resulting from application of Algorithm 1 to $G = (V, E, \lambda, \mu)$ with source node s, target node t and path bound P, has the following properties:*

P1: Every path $(s, v_1, \ldots v_{k-1}, t)$ from s to t in G of length $k \leq P$ is also present in G_P.

P2: Every vertex $v \in V_P$ lies on an s-t path of length $\leq P$ in G.

P3: Every edge $(u, v) \in E_P$ lies on an s-t path of length $\leq P$ in G.

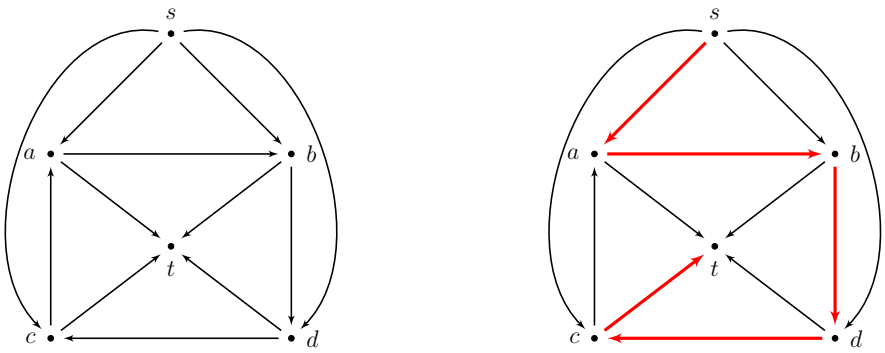
Algorithm 1: Pruning the reachable graph for path bound P

Input: Graph $G = (V, E, \lambda, \nu)$, nodes s and t in V, int P
Output: $G_P = (V_P, E_P, \lambda_P, \nu_P)$

For all $v \in V$, set $\mathsf{DistS}[v] := \infty$ and $\mathsf{DistT}[v] := \infty$;
$\mathsf{DistS}[s] := 0$, $\mathsf{DistT}[t] := 0$;
// Assign DistS[·] using BFS with forward edges
$d := 0$;
InsertInQueue(FwdQueue, s);
repeat
  // Assign shortest distance from s
  $u :=$ DeleteFromQueue(FwdQueue);
  if $\mathsf{DistS}[u] < P$ **then**
    for *each* v *s.t.* $(u, v) \in E$ *and* $\mathsf{DistS}[v] = \infty$ **do**
      $\mathsf{DistS}[v] := \mathsf{DistS}[u] + 1$;
      InsertInQueue(FwdQueue, v)
until FwdQueue *not empty*;
// Assign DistT[·] using BFS with backward edges
$d := 0$;
if $\mathsf{DistS}[t] \leq P$ **then**
  InsertInQueue(BwdQueue, t);
  repeat
    // Assign shortest distance to t
    $v :=$ DeleteFromQueue(BwdQueue);
    if $\mathsf{DistT}[v] < P$ **then**
      for *each* u *s.t.* $(u, v) \in E$, $\mathsf{DistT}[u] = \infty$ *and* $\mathsf{DistS}[u] \neq \infty$ **do**
        $\mathsf{DistT}[u] := \mathsf{DistT}[u] + 1$;
        InsertInQueue(BwdQueue, u)
  until BwdQueue *not empty*;
// Compute V_P, E_P, λ_P and μ_P
$V_P := \{v \mid v \in V, \mathsf{DistS}[v] + \mathsf{DistT}[v] \leq P\}$;
$E_P := \{(u, v) \mid u, v \in V_P, \ \mathsf{DistS}[u] + 1 + \mathsf{DistT}[v] \leq P\}$;
$\lambda_P := \lambda$ restricted to V_P;
$\mu_P := \mu$ restricted to E_P;
return $G_P = (V_P, E_P, \lambda_P, \mu_P)$

Proof. The proof follows from the definitions of V_P, E_P and from the observation that every vertex v on an s-t path of length $\leq P$ necessarily has $\mathsf{DistS}[v] + \mathsf{DistT}[v] \leq P$. Similarly, every edge (u, v) on an s-t path of length $\leq P$ necessarily

has $\mathsf{DistS}[u] + 1 + \mathsf{DistT}[v] \leq P$. □



(a) (b)

Figure 5.2: (a) shows an example graph with source s and target node t where every node and edge falls on some s-t path of length ≤ 3. (b) Gives the same graph and shows an s-t path of length > 3 in red.

A few points about Algorithm 1 are worth noting.

- The time complexity of Algorithm 1 is clearly in $\mathcal{O}(|V| + |E|)$.
- Despite Lemma 5.2.1, it is possible to find s-t paths of length exceeding P in G_P. An example is given in Fig. 5.2a, where the path bound $P = 3$. While each node and edge in this graph participates in at least one s-t path of length ≤ 3, there are s-t paths of length > 3, viz. $s \to a \to b \to d \to c \to t$, shown in red in Fig. 5.2b. Since the focus is only on paths of length $\leq P$, one cannot simply consider all s-t paths in G_P for purposes of computing node and/or edge ranks. This problem is addressed in Sections 5.3 and 5.4.

5.3 Step 2: Ranking of nodes

Several centrality measures for nodes/edges in networks were discussed earlier as part of the literature survey. In the context of gene regulatory networks or protein-protein interaction networks, these can be broadly categorized as degree centrality measures used to identify hubs, betweenness centrality measures used to identify bottlenecks, and eigenvector centrality measures used to identify the importance of a node and its transitive neighbors in a graph.

Of the different measures, it is common practice in biological network study to use betweenness centrality to identify essential nodes or hubs. The betweenness computation in [64] is used to rank the nodes. Higher the betweenness score higher

is the importance of that node. It is to be noted that this centrality based ranking is just a first level filter. Analyzing the top-ranked nodes along with feedback from domain experts regarding which nodes are relevant in the context of the system under study, a few candidates are chosen for performing the next step of the process.

These few top-ranked nodes become the candidates for functional significance checking. Each of these chosen nodes is considered for step 3. The third step of the approach is described in Section 5.5.

5.4 SAT encoding of the problem

A labeled graph $G = (V, E, \lambda, \mu)$ representing the merge of all gene regulatory information available in a public database has thousands of nodes and edges. There can be very long simple paths (spanning hundreds or thousands of nodes) between a pair of source and target nodes in such a graph. Such long paths typically represent unrealistic and unsustainably long chains of gene regulatory mechanisms. Therefore, such paths have been pruned out before searching for an explanation subgraph. Given a source node s, a target node t, and a path length bound Δ, a subgraph $\hat{G} = (\hat{V}, \hat{E}, \hat{\lambda}, \hat{\mu})$ of G is extracted, that contains every simple path of length $\leq \Delta$ from s to t in G. Once $\hat{V}$ is defined, $\hat{E}$, $\hat{\lambda}$ and $\hat{\mu}$ are obtained by restricting E, λ and μ respectively to $\hat{V}$. To obtain $\hat{V}$ algorithmically, a forward breadth-first search (BFS) is performed from s in G, and the reachable nodes are marked with their shortest distance from s up to a maximum of Δ. Next, a backward BFS from t in G is performed, and the reachable nodes are similarly marked with their shortest distance to t up to a maximum of Δ. Finally, only those nodes are chosen for which the sum of the shortest distance from s and shortest distance to t is $\leq \Delta$. It is not hard to see that while $\hat{G}$ contains every simple s-t path of length $\leq \Delta$ in G, it may also contain some longer s-t paths. In practice, the parameter Δ is chosen based on domain expert inputs, such that all potentially important s-t paths are included. Henceforth, whenever a reference is made to a labeled graph G, it would mean a reference to the pruned graph $\hat{G}$ for the value of Δ.

The problem of deciding whether an (n, e)-relaxed explanation subgraph exists was shown to be NP-complete in Section 4.4. Now an efficient SAT encoding of the problem is presented. This encoding allows us to leverage spectacular advances made in SAT solving in the past decade to find explanation subgraphs, and eventually to check the functional significance of implicated nodes.

Given a labeled graph $G = (V, E, \lambda, \mu)$, nodes $s, t \in V$, and integers n and e such that $0 \leq n \leq |V|$ and $0 \leq e \leq |E|$, a propositional formula $\varphi_{G,s,t,(n,e),R}$ is constructed that is satisfiable iff there is an (n, e)-relaxed explanation subgraph

of (G, s, t). The formula $\varphi_{G,s,t,(n,e),R}$ is obtained as the conjunction of six sub-formulas:

(i) φ_{conn} encoding topological connectivity between nodes in the explanation subgraph (this uses the fact that all paths are of length $\leq \Delta$)

(ii) φ_{data} encoding the labeling of nodes obtained from microarray data

(iii) φ_{act} encoding the activity condition in Defn 3.2.1

(iv) φ_{comp} encoding the compatibility condition in Defn 3.2.1

(v) φ_{rel} encoding that total node relaxation is $\leq n$ and total edge relaxation is $\leq e$

(vi) φ_{imp} encoding that every node is reachable from s by a path of length at most Δ

These sub-formulas use a set of variables as described below. For each node $v \in V$, three boolean variables, p_v, a_v and r_v, encode whether v is present, active and relaxed respectively, in the explanation subgraph. Similarly, for each edge $e \in E$, three boolean variables, p_e, r_e and f_e encode whether e is present, relaxed and contributes to the activity condition in Defn 3.2.1 respectively, in the explanation subgraph. Finally, for each $v \in V$, $\log \Delta + 1$ propositional variables $d_{v,0}, \ldots d_{v,\log \Delta}$ encode a measure of "distance" from the source s to v in the explanation subgraph. This gives a total of $(4 + \log \Delta) \cdot |V| + 3|E|$ boolean variables.

Encoding of φ_{conn}

The formula φ_{conn} is constructed from two sub-formulas listed below.

- $\varphi^1_{conn} \equiv p_s \wedge p_t \wedge \bigwedge_{(u,v)\in E} \left(p_{(u,v)} \rightarrow p_u \wedge p_v\right)$. This asserts that s and t are present in the explanation subgraph. Furthermore, if an edge (u, v) is present, then so are the nodes u and v.

- For notational convenience, let $(d_{v,*} = \mathbf{k})$ denote the assertion that $\langle d_{v,\log \Delta} \ldots d_{v,0}\rangle$ represents the number k in binary, where $0 \leq k \leq \Delta$. Furthermore, let $(d_{v,*} < d_{u,*})$ denote the assertion that the unsigned integer represented by $\langle d_{v,\log \Delta} \ldots d_{v,0}\rangle$ is less than that represented by $\langle d_{u,\log \Delta} \ldots d_{u,0}\rangle$. Note that the formulas corresponding to both assertions have size in $\mathcal{O}(\log^2 \Delta)$. Now φ^2_{conn} is defined as $(d_{s,*} = \mathbf{0}) \wedge \bigwedge_{v\in V\setminus\{s\}} \left(p_v \rightarrow \bigvee_{(u,v)\in E} \left(p_{(u,v)} \wedge (d_{u,*} < d_{v,*})\right)\right)$. This asserts that the "distance" metric from s to s is 0. Moreover, for every other vertex v in the explanation subgraph, the "distance" metric from s is larger than that of at least one predecessor u in the subgraph.

Proposition 5.4.1. *$\varphi_{conn} \equiv \varphi^1_{conn} \wedge \varphi^2_{conn}$ is satisfiable iff the explanation subgraph has a s-t path of length $\leq \Delta$.*

Encoding of φ_{data}

The formula φ_{data} is simply $\left(\bigwedge_{v:\lambda(v)=+} a_v\right) \wedge \left(\bigwedge_{v:\lambda(v)=-} \neg a_v\right)$.

Encoding of φ_{act}

In order to formulate φ_{act}, the boolean flag $f_{(u,v)}$ associated with each $(u,v) \in E$ is used. This flag indicates whether every path from s ending with (u,v) in the explanation subgraph passes through a node that was originally active (as per given data).

φ_{act} is constructed from the following subformulas.

- $\varphi^1_{act} \equiv \bigwedge_{(v,t)\in E} \left(p_{(v,t)} \rightarrow f_{(v,t)}\right)$. This asserts that all paths from s to t in the explanation graph must pass through an originally active node.

- $\varphi^2_{act} \equiv \bigwedge_{(s,v)\in E: v\neq t} \neg f_{(s,v)}$. In other words, no path of length 1 from s can pass through an originally active node different from s and t.

- $\varphi^3_{act} \equiv \bigwedge_{v\in V:\lambda(v)\neq+} \left(\bigvee_{(u,v)\in E} p_{(u,v)} \wedge \neg f_{(u,v)}\right) \rightarrow \bigwedge_{(v,w)\in E} \neg f_{(v,w)}$. Suppose a node v is not originally active and it already has some incoming path in the explanation subgraph that does not pass through any originally active node. The sub-formula asserts that in such cases, for every outgoing edge (v,w), there is at least one path from s ending in (v,w) that does not pass through any originally active node.

- $\varphi^4_{act} \equiv \bigwedge_{v\in V:\lambda(v)=+} \bigwedge_{(v,u)\in E} f_{(v,u)}$. In other words, all paths that visit an active node (as per given data) immediately satisfy the activity condition.

Lemma 5.4.2. *Let $\varphi_{act} \equiv \bigwedge_{i=1}^{4} \varphi^i_{act}$. Then φ_{act} is satisfiable iff every path from s to t (other than an s-t edge) in the explanation subgraph passes through at least one node v such that $\lambda(v) = +$.*

Encoding of φ_{comp}

As per Defn 3.2.1, if an edge has the label activation ($\mathcal{A}$) then either both its source and target are active or both are inactive. If the edge has label inhibition ($\mathcal{I}$) then the source and target have opposite signs, that is one is active and the

other is inactive. This is encoded as

$$\varphi_{comp} \equiv \bigwedge_{(u,v)\in E} (\mu(e) = \mathcal{A} \longrightarrow ((a_u \wedge a_v) \vee (\neg a_u \wedge \neg a_v))) \\ \wedge (\mu(e) = \mathcal{I} \longrightarrow ((a_u \wedge \neg a_v) \vee (\neg a_u \wedge a_v)))$$

Encoding of φ_{rel}

Let $\left(\sum_{v\in V} r_v.R(v)\right) \leq n$ denote a propositional formula that asserts that the total node relaxation is at most n. $\left(\sum_{(u,v)\in E} r_{(u,v)}.R((u,v))\right) \leq e$ is used to denote a similar formula for variables r_e. Note that sizes of the above formulas are polynomial in the size of the input problem. The formula φ_{rel} is obtained as $\varphi^1_{rel} \wedge \varphi^2_{rel}$, where

$$\begin{array}{rcl} \varphi^1_{rel} & \equiv & \bigwedge_{v\in V:\lambda(v)=+} (r_v \leftrightarrow (\neg p_v \vee \neg a_v)) \wedge \bigwedge_{v\in V:\lambda(v)=-} (r_v \leftrightarrow (p_v \wedge a_v)) \wedge \\ & & \left(\sum_{v\in V} r_v.R(v)\right) \leq n \\ \varphi^2_{rel} & \equiv & \bigwedge_{(u,v)\in E:\mu((u,v))\in\{\mathcal{E},\mathcal{A}\}} \left(r_{(u,v)} \leftrightarrow (p_u \wedge p_v \wedge (a_u \not\leftrightarrow a_v))\right) \wedge \\ & & \bigwedge_{(u,v)\in E:\mu((u,v))\in\{\mathcal{I},\mathcal{R}\}} \left(r_{(u,v)} \leftrightarrow (p_u \wedge p_v \wedge (a_u \leftrightarrow a_v))\right) \wedge \\ & & \left(\sum_{(u,v)\in E} r_{(u,v)}.R((u,v))\right) \leq e \end{array}$$

Encoding of φ_{imp}

The implication constraints specify that every node other than the source and target must have at least one incoming and at least one outgoing edge. The source must have at least one outgoing edge, and the target must have at least one incoming edge. This is written as

$$\begin{array}{rcl} \varphi_{imp} \equiv & & \bigwedge_{v\in V\setminus\{s,t\}} p_v \rightarrow (\bigvee_{(x,v)\in E} p_{(x,v)} \wedge \bigvee_{(v,y)\in E} p_{(v,y)}) \\ & & \wedge p_s \rightarrow \bigvee_{(s,y)\in E} p_{(s,y)} \wedge\ p_t \rightarrow \bigvee_{(x,t)\in E} p_{(x,t)} \end{array}$$

Finally all six types of constraints are conjuncted to obtain $\varphi_{G,s,t,(n,e),R}$.

Theorem 5.4.3. *Let $\varphi_{G,s,t,(n,e)} \equiv \varphi_{conn} \wedge \varphi_{data} \wedge \varphi_{act} \wedge \varphi_{comp} \wedge \varphi_{rel} \wedge \varphi_{imp}$. Then $\varphi_{G,s,t,(n,e),R}$ is satisfiable iff there is an (n,e)-relaxed explanation subgraph of (G,s,t) under R.*

Proof sketch: Any (n,e)-relaxed explanation subgraph of (G,s,t) defines values of p_v, a_v, p_e and a_e for all $v \in V$ and $e \in E$. These, in turn, define values of r_v, r_e and f_e for all vertices and edges. Finally, the values of $d_{v,\log\Delta}, \ldots d_{v,0}$ can be assigned based on the shortest distance of v from s in the explanation subgraph. The resulting values of all boolean variables can easily be shown to satisfy $\varphi_{G,s,t,(n,e)}$. Conversely, a solution of $\varphi_{G,s,t,(n,e),R}$ gives values of all boolean variables. Using p_v, a_v, p_e and a_e for all $v \in V$ and $e \in E$, we obtain a subgraph of G. By virtue of

the construction of $\varphi_{G,s,t,(n,e),R}$, this subgraph satisfies all properties of an (n, e)-relaxed explanation subgraph. □

5.5 Step 3: Pareto optimality and functional significance

5.5.1 Functional significance of a node

Recall that the eventual goal of this thesis is to check the *functional significance* of a node that plays such a central role that its removal makes it impossible to explain the observations without admitting higher node or higher edge relaxations. In such a case, removing v from G would require using a relaxation pair (n, e) that strictly dominates $(n^\star, e^\star)$.

Note that since every explanation subgraph without v is also an explanation subgraph when v is present in G, it must be that $n^\star \leq n$ and $e^\star \leq e$. If, on the other hand, v does not play any role in explaining the observations, in its absence, other actors can compensate for its role. In this case, an explanation subgraph may be obtainable with $(n^\star, e^\star)$ relaxations even after removing v from G. This is the motivation to use the domination of Pareto-optimal points in the presence and absence of v as a means to detect its functional significance.

To see the importance of a particular node of interest, say i in (G, s, t, W) the Pareto-optimal curve could be obtained in the presence of node i. This Pareto-optimal curve is called the *solution curve* of G, denoted $SolC(G, s, t, W)$. Now, for a particular pair (n, e) of values on the solution curve, if the number of (n, e)-explanation subgraphs is small, then one could inspect those to see if a particular node of interest is present in all (most of them). However, it turns out that in most experiments, the number of such explanations is more than 1000, which makes it impossible to inspect them manually. Instead, in order to see the *importance* of a node, the shift of the solution curve on the removal of the node i is examined. Formally, given two solution curves C_1 and C_2, it is said that C_2 is *worse than* C_1 if both the following colditions hold.

(i) The curves do not intersect

(ii) For each (n_1, e_1) on C_1 and (n_2, e_2) on C_2 we have, if $n_2 = n_1$ then $e_2 > e_1$, and if $e_2 = e_1$ then $n_2 > n_1$.

5.5.2 Efficient computation of PO-curves

Two extreme ends of the Pareto-optimal curve represent solutions with either a very high node relaxation or a very high edge relaxation. The nature of our

problem is such that these solutions will not carry much weight.

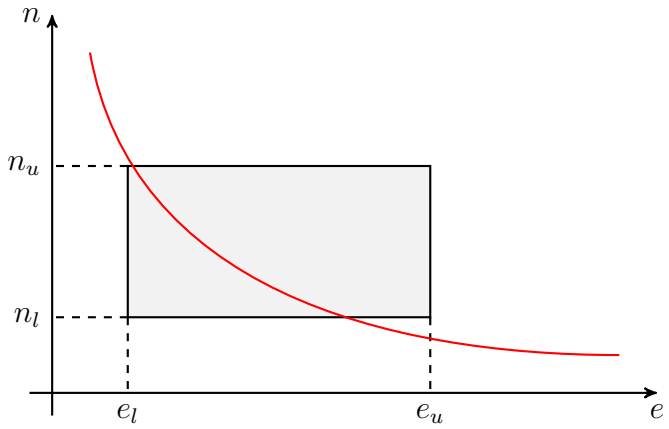


Figure 5.3: Clipping PO curve at user provided bounds

Towards this effect, we ask the user for upper and lower bounds on the node and edge relaxation values that may be allowed. This restriction transforms the search region into a rectangular "window" as shown in Fig. 3.4. The values e_l, e_u, n_l, and n_u are the user given values for lower bound on edge relax, upper bound on edge relax, lower bound on node relax and upper bound on node relax respectively.

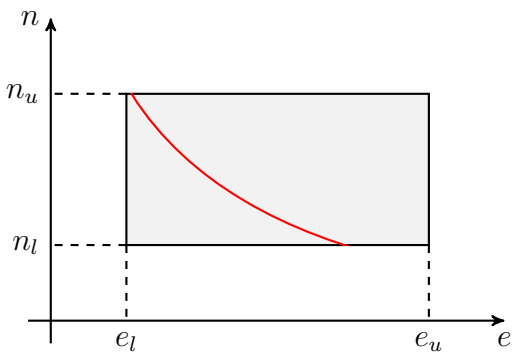


Figure 5.4: PO curve restricted within user defined window

Fig. 5.3 shows the user-defined window on a specimen PO curve. Only the part of the curve that lies within the grey rectangle representing the window is to be retained. Fig. 5.4 shows the relevant part to be obtained from the specimen PO curve.

The next section delves into the approach to obtain such a truncated PO curve from the compiled set of constraints. Note that, at each point on the curve, the solver might return several possible solution subgraphs.

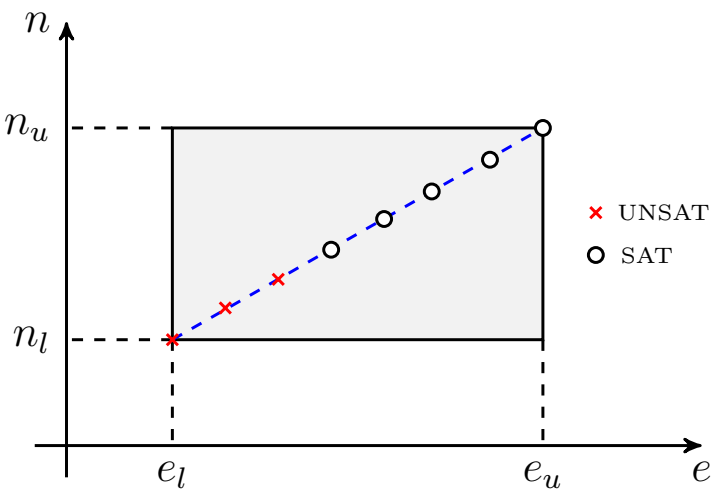


Figure 5.5: Search for SAT point along diagonal

SAT computation along diagonal

To arrive at the first PO point quickly, we perform a binary search along the diagonal of the window and find the point along the diagonal, where we get a sat solution for the first time. This point may or may not be a Pareto-optimal point by itself. This scenario is depicted in Fig. 5.5. The red cross points denote unsat, and the black circles denote points that have a satisfiable solution.

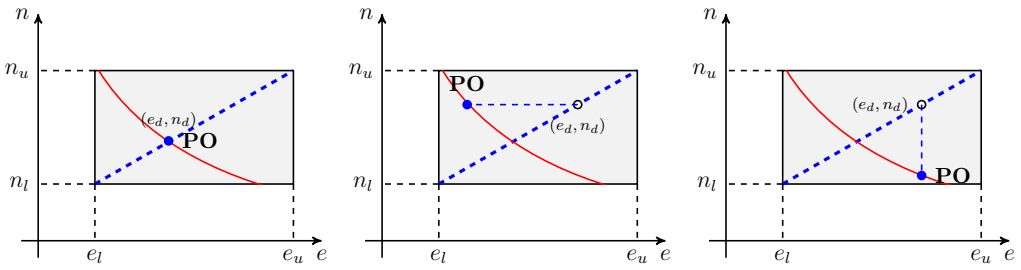
Theorem 5.5.1. *If there is no solution at a specific point (n, e) on the diagonal of the user-defined box, then there is no solution at any point (n', e') strictly dominated along the diagonal.*

Proof. Let us assume that, there is a solution with relaxation bound (n', e') such that, $n' \geq n$ and $e' \geq e$. The solution at point (n', e') continues to be a solution for every relaxation bound, even point (n, e). This is a contradiction. Hence the assumption is wrong. Hence the proof. □

Corollary 5.5.1.1. *If there is no solution along the diagonal of the user-defined box, there is no solution in the box.*

Proof. The proof follows from the proof of Theorem 5.5.1. Consider the top-rightmost corner of the user-defined box. Let this point be called (n_c, e_c) If there is no solution at this point, it would imply that there is no solution at any point strictly dominated by it. Since all points in the window is strictly dominated by the point (n_c, e_c), it follows that there can be no solution in the box. □

Thus, if there is a Pareto-optimal point within the window, and if a search is performed along the diagonal a point will be encountered on the diagonal at the same value of edge or node relaxation, that is SAT. Discovery of this point vastly reduces the search space and helps in keeping the run-time low. The time complexity of the process is discussed in a later section.



(a) Point on diagonal is PO (b) PO at same node relax (c) PO at same edge relax

Figure 5.6: Possible positions of the PO point relative to the diagonal

From points on diagonal to PO points

As claimed earlier, if there exists a Pareto-optimal point in the window, it will have a satisfiable point along the diagonal. Here, this claim is investigated further by case analysis.

Suppose, a solution is found (black circle) along the diagonal (blue dashed line) as shown in Fig. 5.6.

> **Case I:** As shown in Fig. 5.6a, the point obtained on the diagonal might itself be a Pareto-optimal point (lies on the red line representing the Pareto-optimal front). This case is easy. The next step of the approach can be taken up immediately.
>
> **Case II:** A sat point is obtained on the diagonal. It is not a Pareto-optimal point but is dominated by a point in the same horizontal line (blue dot) which is Pareto-optimal. That is, there exists an $e < ed$, such that, (n_d, e) dominates (n_d, e_d). This PO point can also be interpreted as the projection of all sat points with the same n_d. In other words, to find the non-dominant

point, the smallest e needs to be found such that (n_d, e) is sat. This case is shown in Fig. 5.6b.

Case III: The case, shown in Fig. 5.6c co-occurs with Case II and is analogous to it. Here the point (n_d, e_d) is dominated by point (n, e_d), where $n < n_d$. The point (n, e_d) can be looked at as a projection of all sat points whose edge relaxation value equals to e_d. Here the search is for the smallest n such that (n, e_d) is sat.

Of the three cases, Case I already gives a non-dominated point. For Cases II and III, the strategy described next is employed.

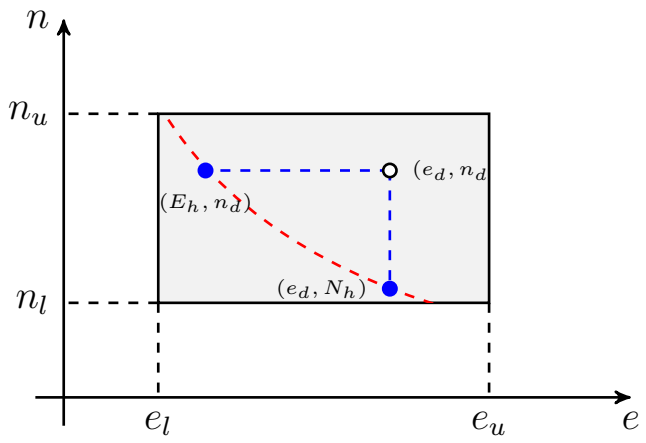


Figure 5.7: Finding non-dominated points

Non-dominated point with smallest node relax bound Recall that Cases II and III occur together. Fig. 5.7 shows these two cases. To find the non-dominated PO point with the smallest n, while e_d is kept constant, and a search is performed for the smallest n within the window such that (n, e_d) is sat. A binary search is performed within the bounds n_d and user-defined node relaxation lower bound n_l. The point from this binary search where the answer is sat for the first time along this vertical line, (N_h, e_d), is one of the Pareto-optimal points. This solution is called the *lower solution*.

Non-dominated point with smallest edge relax bound For the non-dominated PO point with smallest e, n_d is analogously kept constant, and a search is conducted for the smallest e within the window such that (n_d, e) is sat. The binary search, in this case, is performed along a horizontal line within the bounds e_l, which is the user-defined edge relaxation lower bound, and e_d. Analogous to the previous

case, the solution of the binary search, (n_d, E_h), is the other Pareto-optimal point. This solution point is termed the *upper solution*.

If the *lower solution* and the *upper solution* return the same point, then only one solution PO point (n_d, e_d) is obtained. This is the Case I.

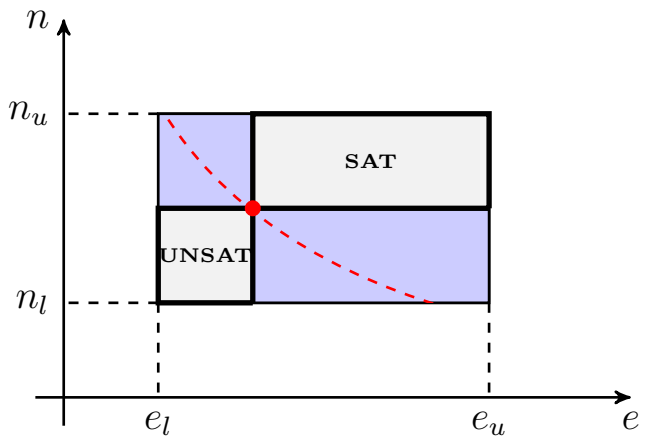


Figure 5.8: Pruning out SAT and UNSAT regions of the solution space

Recursive method for PO-curve computation

The Pareto-optimal curve consists of points in a line along a front that encloses a convex area. Only a small fraction of the points in the feasible region constitute the PO-curve. The PO-points are obtained by making satisfiability calls for a small over-approximation of the area that nests the actual PO-curve.

Pruning of SAT and UNSAT regions Once the algorithm finds one Pareto-optimal point, some areas on the solution space can be excluded from the subsequent search. Fig. 5.8 shows two boxes outlined in bold. If the red point is obtained as the PO-point, all points in the lower left box will surely be UNSAT. Also, the upper right box will certainly have all SAT solution points. The remaining points of the PO-front can only lie in the two regions colored in blue.

At each PO-point obtained these known SAT and UNSAT regions are pruned, and the next satisfiability query is focussed on only the part of the solution space, some of which points will be Pareto-optimal. By this process, the search area is cut down at a logarithmic rate.

In Fig. 5.9, a grid is shown. Since this is a discrete domain problem, only integer node and edge relaxation bounds are used. Once the Pareto-optimal point

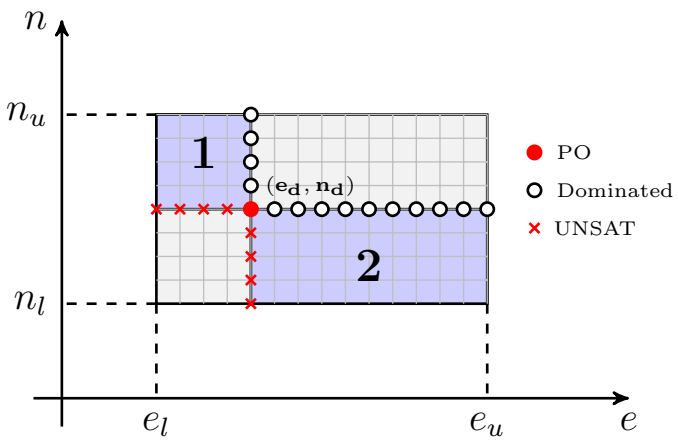


Figure 5.9: UNSAT and dominated points obtained from a PO point

(shown as a red dot) is obtained, the status of all points at node relax value n_d are known. At the same edge relax value, all points with node relax bound $< n_d$ are unsat, and all points with node relax bound $> n_d$ are dominated by the PO point (n_d, e_d). Similarly, at the same node relax value, all edge relax points with value $< e_d$ are unsat, and all points with edge relax value $> e_d$ are dominated by (n_d, e_d).

Two windows for subsequent calls The entire user-defined window is considered at the start. As each PO-point is retrieved, the focus moves to the blue boxes marked with the numbers 1 and 2, as shown in Fig. 5.9. After the first call, based on the coordinates of the PO-point obtained, the subsequent window(s) are identified. The same procedure is repeated on these automatically identified windows. The procedure exits when the search space either shrinks to a point or contains no more SAT solutions.

Recursive calls The successive recursive calls will be explained with the help of Fig. 5.10. Suppose the first PO point obtained is at (n_d, e_d). As shown in the previous figure, Fig. 5.9, two recursive calls to the Pareto-optimality inference method are made with the two light blue windows. Suppose in the window marked 1, the next PO point is obtained as (n'_d, e'_d). After pruning out the SAT and UNSAT boxes, two recursive calls are made to the method with the two dark blue boxes as the search windows. Similarly for window 2, say the PO point is at (n''_d, e''_d). The window marked 2 is then reduced to the two dark blue boxes within window 2. The algorithm proceeds thus until each search window either shrinks to a point or has no more sat solutions remaining.

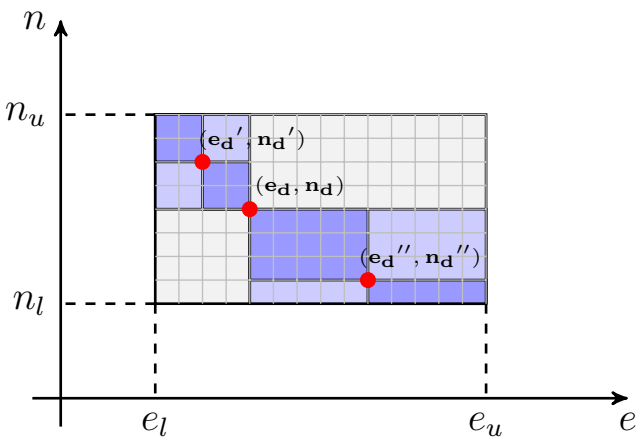


Figure 5.10: Recursive calls

End point calculation of subsequent windows Finally, a more in-depth discussion is presented on how the endpoints of the search window for the subsequent recursive calls are determined automatically by the method.

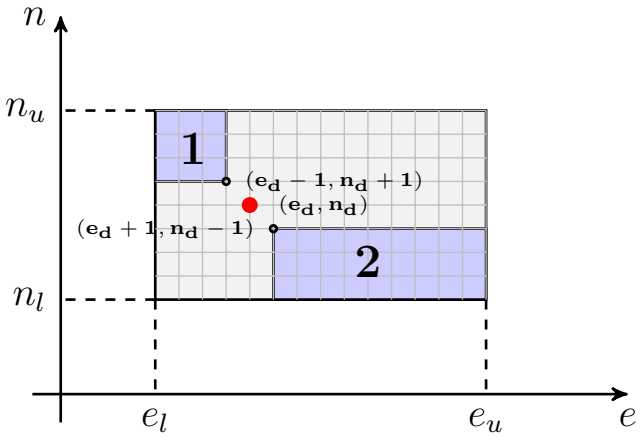


Figure 5.11: Recursive call window end points

In Fig. 5.11, suppose we have obtained the Pareto-optimal point (n_d, e_d). As shown earlier in Fig. 5.9, the points in the search window can be divided into the following categories:

(i) $\forall(e, n)$ where $e < e_d$ and $n < n_d$, there are no sat solutions.

(ii) $\forall(e, n)$ where $e = e_d$ and $n < n_d$ also, there are no sat points.

(iii) $\forall(e, n)$ where $e < e_d$ and $n = n_d$, analogously, there are no solutions.

(iv) $\forall(e, n)$ where $e > e_d$ and $n > n_d$, (e, n) is a dominated point.

(v) $\forall(e, n)$ where $e > e_d$ and $n = n_d$, (e, n) is a dominated point.

(vi) $\forall(e, n)$ where $e = e_d$ and $n > n_d$, (e, n) is a dominated point.

Thus the regions that harbor the other Pareto-optimal points and need to be explored further are the two rectangle windows defined by the points:

(i) Window 1: $(e_d - 1, n_d + 1)$ and (e_l, n_u)

(ii) Window 2: $(e_u, n_l))$ and $(e_d + 1, n_d - 1)$

The search method is called recursively on these two rectangle search spaces. All the PO points obtained in the various calls are put together to construct the final PO curve inside the user-defined window.

This recursive calling approach was developed independently as part of this thesis. Later the work mentioned in [113] was discovered, where the authors have used a similar technique to call their function on subproblems recursively. The context and main algorithms are vastly different from the approach described in this work.

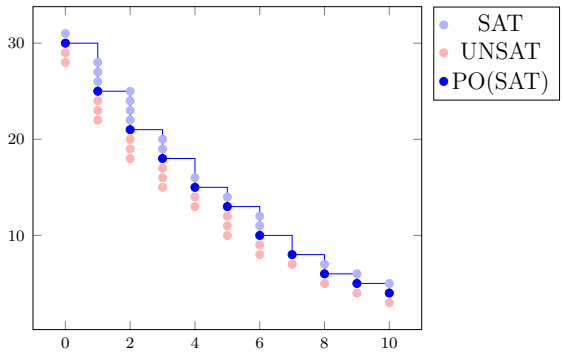


Figure 5.12: Step-like structure of PO curve

A plot was obtained from the number nodes relaxed, and the corresponding number of edges relaxed for every point on the PO curve. It was observed that if number of nodes relaxed was increased, keeping the number of edges relaxed the same, or the number of edges relaxed was increased, keeping the number of nodes relaxed the same, there is a step-like structure as shown in Fig. 5.12. The dark blue points are PO points. The light blue points are dominated points at which the problem is SAT. The light red points are UNSAT.

Algorithm 2: F(e_u, n_l, e_l, n_u)

Input : Two points (e_u, n_l) and (e_l, n_u) defining the search box
Output: All PO points within this box

```
1  curr_n ← (n_l + n_u)/2
2  curr_e ← (e_l + e_u)/2
3  if no solution exists at (curr_n, curr_e) then
       /* All points along diagonal lower than this have no
          solution                                          */
4      if (curr_n == n_u && curr_e == e_u) then
5          res = (0, -1, -1, -1, -1)
6      else
7          curr_n ← curr_n + (n_u − curr_n)/2
8          curr_e ← curr_e + (e_u − curr_e)/2
           /* Check for solutions                           */
9      end
10 end
11 if ((n_u < n_l) || (e_u < e_l)) then
12     return;
13 end
14 res ← (count, n_1, e_1, n_2, e_2)
15 if (res.count == 0) then
       /* No more PO points in window                       */
16     return;
17 end
18 if (res.count == 1) then
       /* One PO point in window                            */
19     Add (res.n_1, res.e_1) to set of PO points
20     F (res.e_1, res.n_1+1, e_l, n_u)
21     F (e_u, n_l, res.e_1, res.n_1-1)
22 end
23 if (res.count == 2) then
       /* Two PO points in window                           */
24     Add (res.n_1, res.e_1) and (res.n_2, res.e_2) to set of PO points
25     F (res.e_1, res.n_1+1, e_l, n_u)
26     F (e_u, n_l, res.e_2, res.n_2-1)
27 end
```

Time complexity First a binary search is used along the (n_l, e_l) to (n_u, e_u) diagonal of W to find the smallest (under $\sqsubseteq$) pair (n_d, e_d) for which $\varphi_{G,s,t,(n_d,e_d),R}$

is satisfiable. Note that (n_d, e_d) may not be a Pareto-optimal point. Then a binary search is used on (n, e) pairs in $\langle [n_d, e_l], [n_d, e_d] \rangle$ and $\langle [n_l, e_d], [n_d, e_d] \rangle$ to find the projections of (n_d, e_d) on the Pareto-optimal curve. Once a Pareto-optimal point (n_p, e_p) is obtained, the problem can be recursively decomposed into those of generating Pareto-optimal curves in the relaxation windows $\langle [n_p, n_u], [e_l, e_p] \rangle$ and $\langle [n_l, n_p], [e_p, e_u] \rangle$. This requires a total of $\mathcal{O}(k\, log_2 k)$ invocations of a SAT solver, where $k = \texttt{max}(n, e)$, and gives the Pareto curves, from which the functional significance can be determined.

5.6 Interpretation of PO curves

5.6.1 Objectives

Given the master network constructed from the pathways and a set of microarray experimental data, it so turns out that, if the data is inconsistent with the networks, then something needs to be changed. Either the fidelity of the data is not good enough, or the curation of the network that is supposed to document known biological processes is not useful or relevant enough, or maybe both.

These two aspects feed into each other. If there is disagreement with many of the expression data values, then perhaps there is no need to make any change in the network. Similarly, if a lot of the edges are changed, then it is unnecessary to change too much of the data. Intuitively, more relaxation on one aspect requires less relaxation on the other. Thus the aim is to find minimal points in respect to both node and edge relaxation.

When talking about minimal change for two mutually dependent and opposing parameters, there is no global minimum in terms of both. Such situations are when the notion of Pareto-optimality comes in. The points where there are solutions with minimal node and edge relaxation are not ordered. Only the dominated points are ordered, but they are not minimal using the search criteria. The Pareto-optimal curve gives the minimal changes that need to be made in the collection of expression data and network edges.

5.6.2 Single pareto-optimal curve

The microarray experiment that led to the observations has certain gene regulatory/signal transduction pathways involved in it. If that ideal subnetwork, i.e., the fully or partially known golden truth, is subsumed in the original network, and all the observations were correct, then the constraint solver will undoubtedly find out the ideal subnetwork as a solution with no modification required.

Now, suppose the actual pathways are present within the reachable network,

given a source and target, but the wet-lab experiment gives rise to data which is not entirely consistent with the network. Then some of the data points would need to be changed to get a consistent solution. On the other hand, suppose the pathways or network edges from KEGG are not precisely capturing the phenomenon that occurs. Then some edges in the network would need to be changed. Hence three situations may arise:

(i) A few data points in the dataset need to change in order to capture the precise data that the ideal network would generate

(ii) The ideal network is not represented entirely by the KEGG edges, and hence a few edges have to be modified

(iii) Both the above have to be done simultaneously

If the most plausible pathway that matches with the experimental observation exists, then the search method using a constraint solver will furnish one way of explaining the data. This one possibility will correspond to at least one biologically relevant solution in at least one point on the Pareto-optimal curve. This PO point gives precisely the numbers of edges and nodes to be relaxed for the ideal explanation.

What the real solution represents might be at just one or a few points on the Pareto-optimal curve. In the worst case, there might be only one solution at only one of the Pareto-optimal points. All other points might represent solutions that require either more edges or more nodes to be relaxed.

Thus, there are three levels of existential quantification:

(i) $\exists$ a PO point with e edge and n node relaxations

(ii) $\exists$ set $N = \{node_1, node_2, \ldots, node_n\}$ of n relaxed nodes, and set $E = \{edge_1, edge_2, \ldots, edge_e\}$ of e relaxed edges, corresponding to this PO point

(iii) $\exists$ a solution subgraph H corresponding to the set E of edges and set N of nodes

Just by looking at the PO curve, the PO point with the ideal solution cannot be identified. Also, it is not known which edges and nodes are relaxed for a PO point. Even if it is known which edges and nodes got relaxed, there is no unique solution.

The relaxation of some high-confidence edges and nodes might need to be avoided. The implementation presented in this work has the facility to avoid the

relaxation of specific edges or nodes. If the relevant information is provided, then solutions and Pareto-optimal curves without relaxing those edges and nodes can be obtained.

5.6.3 Exclusion experiments

The three levels of existential quantification indicate that there are a lot of unknown factors in the system, and the current situation is far away from identifying the network that represents the context-dependent biological phenomenon. The method given here is notable in the following instances.

(a) Given a source and a target, all possible ways in which the source and target are connected in the network can be identified

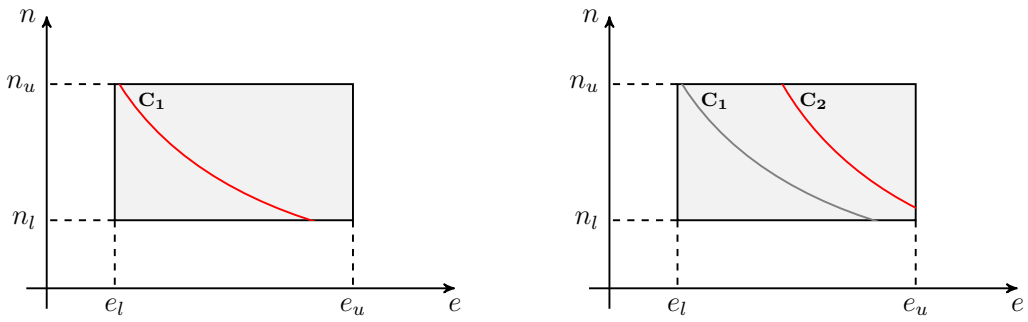
(b) The key players responsible for information flow within the bounds specified can be distinguished

In order to identify such a player, what needs to be performed is what is called a node exclusion experiment. Here the constraint solver is asked to find solutions or explanatory graphs by intentionally denying a specific node to appear in the solution. The node and its related edges are present both in the KEGG and user-defined network, but the constraint solver is asked to avoid this node and its related edges and still find a rewired network or path. This experiment comes close to mimicking the biological circumstances when knocking out or inhibiting a node is expected to affect the original phenomena. Because of the redundancy in biology that ensures robustness, more than one node may be necessary for the observed phenomena, and their removal or inhibition may affect the outcome differentially and to varying degrees.

Upon node exclusion, the behavior of the PO curves would change. Depending on the nature of change after the exclusion of each node, its importance can be ascertained.

Identifying key actors

The goal of this thesis is to use the Pareto-optimal curve to identify key actors (nodes). Suppose, in the ideal pathway, there was a key actor. Also, this key actor had been so central that on excluding this, there is no signal transduction from the source to the target, and the original phenomenon is not observed. In that case, if the key player is removed from the pathway, that is, the constraint solver is asked for an explanation without choosing this node, it will not be able to provide an explanation. The unavailability of even a single solution means every solution needs the presence of this node. Once that node is left out, there is no way to



(a) PO curve with key actor (b) PO curve without key actor

Figure 5.13: Shifting of pareto-optimal curve

explain the data. The ideal network might have had changes in the data and the network edges from what was given initially, but on knocking out the key player, there is no longer a solution with the same amount of node and edge changes. By retaining the node, there would have been at least one solution.

Recall that the method measures only the number of nodes relaxed and the number of edges relaxed, and not the actual nodes that are relaxed or the actual edges that are relaxed. Suppose, without excluding the critical node, a solution subgraph with some specific nodes and edges relaxed was obtained. After excluding the critical node, solutions were no longer obtained at the same point on the Pareto-optimal curve. Then the following statements hold.

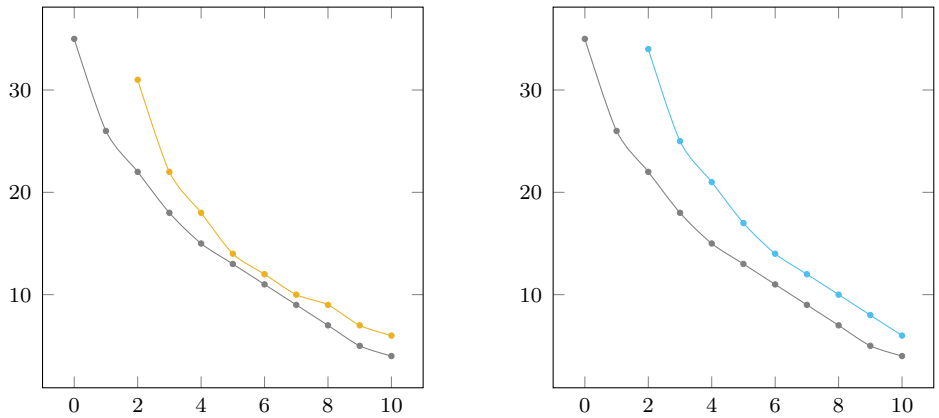
(i) There is no way of explaining the data on the reachable network with the same edges and same nodes relaxed

(ii) There is no way of explaining the data on the reachable network with the same number of edges and the same number of nodes relaxed

The second point is a more stringent condition. It says one has to change; in reality, one has to increase, the number of edge or node (or both) relaxations to obtain solutions after knocking out the key player. With no node relaxation combination and no edge relaxation combination of the same sizes as before, can a solution be obtained without this node? Some severe repercussion has happened on removing this node. Hence the claim is that this node is playing an essential role in the process that is being captured.

Given two solution curves C_1 and C_2, C_2 is said to be *worse than* C_1 in W if the curves do not intersect and for each (n_1, e_1) on C_1 and (n_2, e_2) on C_2 the following hold. If $n_2 = n_1$ then either $e_2 \notin [e^l, e^u]$ or $e_2 > e_1$, and if $e_2 = e_1$ then either $n_2 \notin [n^l, n^u]$ or $n_2 > n_1$.

On the removal of a node, if the whole PO curve shifts, that is, the earlier and the new curve do not intersect at any point within the user-defined window, it is claimed that the node was playing a very crucial role. Any of the PO points could have contained the ideal solution. It might not be known which PO point contains the ideal solution. One or more PO points might have hosted alternate hypotheses solutions. The exclusion of this node has shifted all the PO points, including the one that contained the ideal solution. Hence it does not matter where the PO point corresponding to the ideal solution was on the curve.



(a) PRKACA (b) AKT3

Figure 5.14: Node exclusion experiments giving shifted PO curves

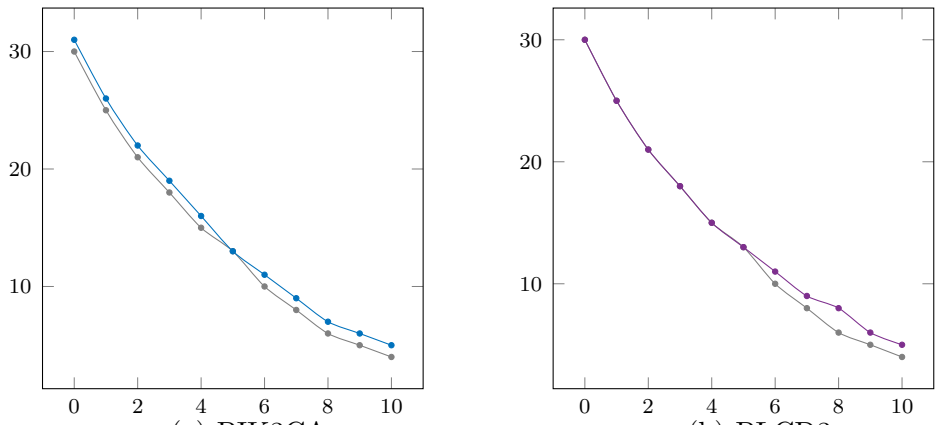
If, after the node exclusion experiment, the PO curve shifts away from the original one, an absolute claim can be made that the exclusion node (or key actor) was playing an important role in the biological phenomenon captured. A higher claim can be made that, the role of this actor was critical not only in the ideal solution but also in alternative hypotheses with few more nodes and/or edges relaxed. The possible solutions that allowed for a little more leeway either on the node or on the edge relaxations also had this node as a crucial candidate.

Partially shifted PO curve

For cases where the PO curve without the critical player does not separate from the one that is in the presence of the key player, a weaker claim can be made.

As seen in Fig. 5.15a, the two Pareto-optimal curves, with and without PIK3CA, intersect at one point. At every other point, the curve has shifted. The common point might contain the explanation that took place in the experiment. If that is the case, then PIK3CA is not such a central actor.

The fact that one of the points has not shifted could mean any one of the following things.



(a) PIK3CA (b) PLCD3

Figure 5.15: Node exclusion experiments giving shifted PO curves

(i) The ideal solution was at one of the points that shifted and PIK3CA played an important role.

(ii) The ideal solution is present at the intersecting point, and PIK3CA is not a valuable player. Keeping it or knocking it out does not make any difference. The ideal solution still has explanatory paths from source to target.

(iii) The ideal solution is there at a shared point, and PIK3CA plays an important role. If PIK3CA is excluded, then the same number, but a different set of nodes or edges, kick in as a compensatory mechanism. In this alternate solution, PIK3CA does not play an important role. Since only the number of nodes and edges relaxed are tracked, the two solutions cannot be distinguished.

The two curves with and without node PLCD3 overlap more than the two curves involving node PIK3CA. An argument can be presented on similar lines for this or any case where there is at least one common point between the two curves within the user-defined window.

Magnitude of shift

Recall that all solutions that are present without the key actor, that is the solutions above the colored curves in Fig. 5.14 and Fig. 5.15, are also present when considering solutions in the presence of the key actor, that is, the solutions above the grey curve in the two figures mentioned.

The gap between two completely disjoint PO curves indicates that, when the key player is removed, there is no alternate solution until the colored curve is

reached (relaxed as many nodes and edges as indicated by the colored curve). Thus, for a smaller gap, the alternate pathways kick in with a few more nodes and edges relaxed. For a bigger gap, more nodes and edges have to be relaxed for any compensatory pathway to get activated.

From the experiments in Fig. 5.14 and Fig. 5.15, it seems both PRKACA and AKT3 are individually playing an important role. However, the role that PRKACA is playing can be compensated by relaxing a few more nodes and edges. Whereas to compensate for the role of AKT3 relatively more nodes and edges will have to be relaxed in its absence.

Limitations

Suppose the explanation network was such that the signal was getting transducted along multiple parallel paths to give rise to the phenotypic outcome. Then, if one player is removed, it will not be possible to see the effect of others either. Also, as seen in Section 5.6.3, if the shifted curves do not separate entirely, no strong claim can be made, since, in such cases, several different outcomes are possible.

As part of future work, the relaxation metric can be refined to track the specific nodes and edges that are relaxed in a solution. With that information, it will be possible to distinguish between sets of solutions at each PO point.

5.6.4 Double exclusion experiments

The behavior is plotted when two nodes are excluded simultaneously.

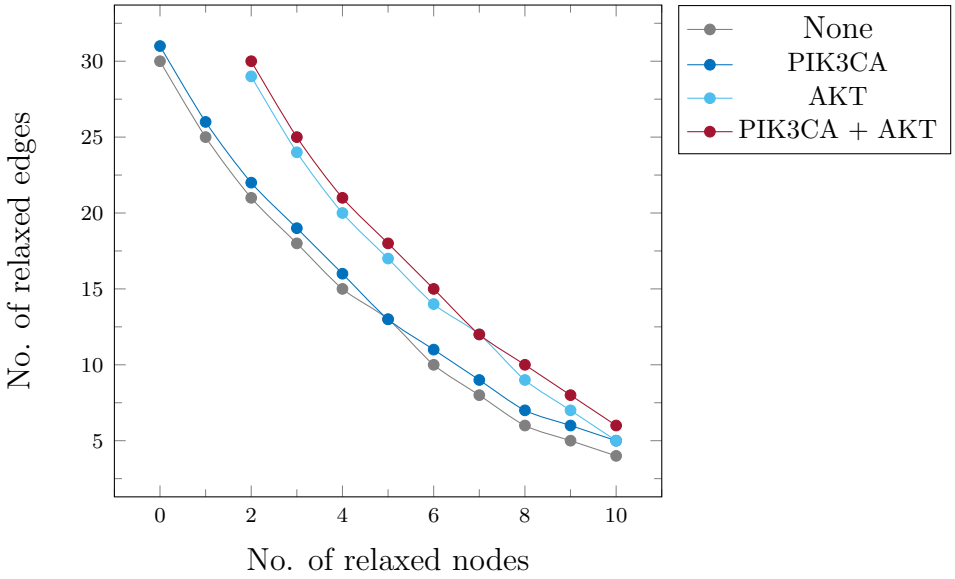


Figure 5.16: Relaxation bounds excluding AKT and PIK3CA

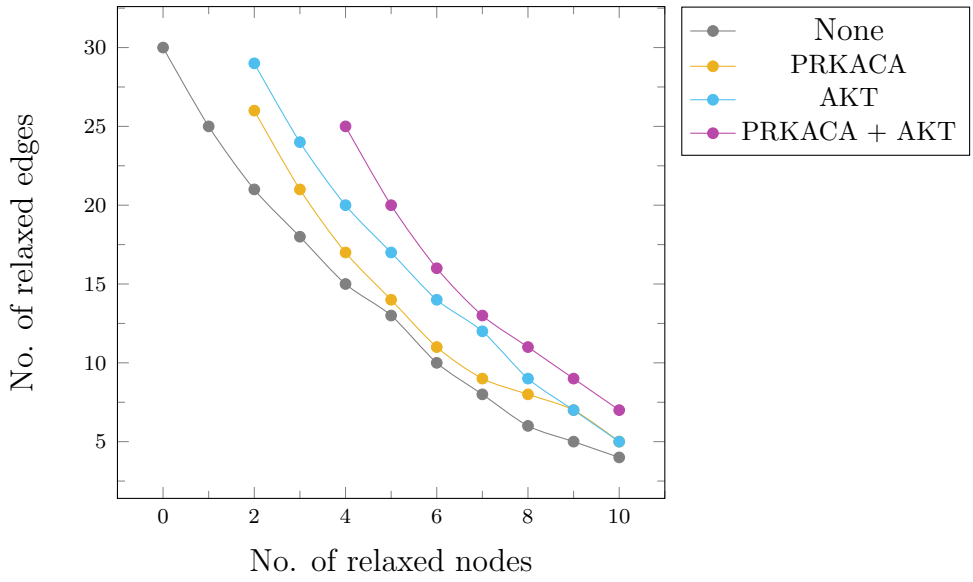


Figure 5.17: Relaxation bounds excluding AKT and PRKACA

Fig. 5.16 shows the effects of disallowing AKT and PIK3CA separately and together. It shows that if PIK3CA is excluded, there is a shift in the curve; that is, some solutions are invalidated. If AKT3 is excluded alone, several solutions are

invalidated. Hence when both these nodes are excluded simultaneously, even more solutions are invalidated. This observation is consistent with the single exclusion curves, and the results are not unexpected. Fig. 5.17 shows similar effects for AKT and PRKACA.

5.7 Implementation

The procedure has been implemented as a tool written in C++ language. The product of the approach is a pipeline of commands carefully sequenced to obtain the required outcome one step at a time. Given a set of pathways, a microarray dataset, a source, and a target node, it will compute the pairs of node and edge relaxation values that can give solution subgraphs and produce the baseline Pareto-optimal curve. If a node is provided whose functional significance needs to be checked, the tool will exclude this node from the analysis and generated the respective exclusion PO-curve. The curves can then be compared to infer the significance of the given node.

The tool has around 35 KLOC. The details of the commands and a short description of how to run those commands as part of the tool is given in Appendix B.The experiments shown in the following section were performed on an Intel(R)-Core(TM)-i7-3770 CPU with 8 cores, clock speed 3.4 GHz and total of 32 GB RAM. The code used C++ API of Z3 version 4.7.1 on Ubuntu 18.04.

Chapter 6

Experiments and Results

6.1 Background for experiments

Cancer is a disease characterized by unrestricted cell growth and impaired control of timely destruction and elimination of old or diseased cells. This results in abnormal formation of masses of cells that we have come to know as tumors. A tumor with malignancy is the most common characterization of cancer.

Ideally, the cell itself tries to degrade the damaged or diseased proteins. One mechanism of degrading a damaged protein is through the ubiquitin-proteasome system. When a protein is to be degraded, it is marked by the attachment of a polyubiquitin chain to it. This protein and ubiquitin chain combination is then attracted to the 26S proteasome. A proteasome is a structure responsible for degrading certain proteins into smaller peptides. The proteasome plays an important role in regulation of proteins that control cell-cycle progression and apoptosis. Hence, this structure is often targetted for anti-cancer therapy [114].

PSMD9 and PSMD10 are helper proteins (called chaperones) that assist in the assembly and maintainance of the proteasome [115]. NFκB is a transcription factor that is involved in a large number of cellular processes and many cells depend on the NFκB activity for their survival [116]. NFκB is responsible for activating proteins that are involved with the immune responses in the cell and in control of cell growth [117]. Hence unwanted behavior of NFκB could result in uncontrolled cell growth and may even play a role in proliferating cancer [118].

Biologists have recently observed that proteins PSMD9 and PSMD10, that mediate proteasome assembly, also differentially affect NFκB activity through their interaction with the hnRNPA1 RNA protein. While PSMD10 is known to inhibit NFκB function, they have discovered a novel role of PSMD9 in direct NFκB activation [119].

Collaboration It has been a privilege to work with expert molecular cell biologists from the Advanced Centre for Treatment, Research and Education in Cancer (ACTREC), Tata Memorial Centre, Navi Mumbai, as part of this research. This collaboration did not only provide the indispensable real-life context where to apply the computational techniques developed in this thesis but also provided valuable opportunity to learn and clear out many doubts from their deep understanding of the domain. One example of an intriguing biological question is what our collaborators are faced with pertaining to proteins PSMD9 and PSMD10 in cancer.

In this thesis, computational experiments have been performed in consultation with our cancer biologist collaborators. These experts had studied the effect of over-expression of gene PSMD9 on HEK 293 human embryonic kidney cell lines. Among many roles played by gene PSMD9, a vital function is regulation of expression of IκBa, an inhibitor of gene NFκB. NFκB is interesting because it is a transcription factor induced upon chemo and radiation therapy in cancer treatment. The microarray dataset from this experiment is the profile on which the computational studies mentioned as results in this thesis have been performed.

6.1.1 Database, data profile, source and target nodes

The dataset used in the computational experiments is a microarray profile obtained by over-expressing PSMD9 and observing expression values of an array of nodes affected by this perturbation. The microarray dataset had 1654 up-regulated and 1199 down-regulated nodes. If a gene occurred more than once in the expression data, we took the average of the fold-change and ignored the few genes that showed as both up and down regulated.

As the database, we used 163 human pathways downloaded from KEGG. These are all the human pathways in KEGG that had information in computer readable format. These 163 KEGG pathways were merged to obtain the master network. Source node, target node and the differentially expressed nodes were not merged with any other node.

For simplicity, the relaxation weights of nodes and edges are assigned to 1. Hence total relaxation weight of nodes is simply the number of nodes relaxed, and the total relaxation weight of edges is the number of edges relaxed.

Our collaboratos had performed extensive wet-lab experiments to understand the role of PSMD9 in migration (movement of cells from one place to another). These experiments were performed on a matrix of a protein called fibronectin. One of the receptors for fibronectin is the protein ITGB1. Hence, after discussion

with the biologists, node ITGB1 was chosen as a source in the computational experiments. On the other hand, migration is directly influenced by actin (ACTB) status in the cell. Several regulators of actin are direct targets of STAT3 transcription factor. Thus the biological experts guided us to choose STAT3 and ACTB as targets.

6.2 ITGB1-STAT3 experiments and results

The first set of experiments were performed with node ITGB1 as source and node STAT3 as target. Merging the 163 KEGG pathways gave a master network of 2498 nodes and 10497 edges.

A reachability traversal was performed on the master network to obtain the reachable network as per Algorithm 1. The bound of reachability is the number of hops or levels, within which nodes and edges are retained in the reachable graph. Depending on the reachability bound, the size of the reachable graph varied significantly. The sizes of the reachable network for different bounds are given in Table 6.1.

Bound	No. of nodes	No. of edges	Up reg	Down reg
2	3	2	1	0
3	15	35	5	0
4	53	231	14	3
5	191	1195	47	25
6	326	2167	69	35
7	483	3044	96	52
8	568	3560	113	59
9	636	3852	119	65
10	659	3959	121	70
11	676	3993	123	73
12	677	3996	123	73
13	677	3997	123	73
14	677	3997	123	73
15	677	3997	123	73

Table 6.1: Sizes of reachable graph and overlap for ITGB1-STAT3 case

While a small bound will not cover many differentially expressed nodes, a large bound will include very long paths that are unlikely to be the explanation pathways. It is important to determine the optimal reachability bound. After

studying the graph sizes, it was seen that with bounds higher than 7, the number of differentially expressed nodes did not vary too much. After a consultation with the experts, reachability bound of 7 seemed reasonable. The reachable graph with a bound of 7 was used for the experiments.

Time required for generating solutions

As discussed earlier, several value pairs for edge and node relaxation were explored. Within the user defined window there are a large number of such points representing various possible pairs of values. Fig. 6.1 shows the times taken to solve individual queries with different relaxation values.

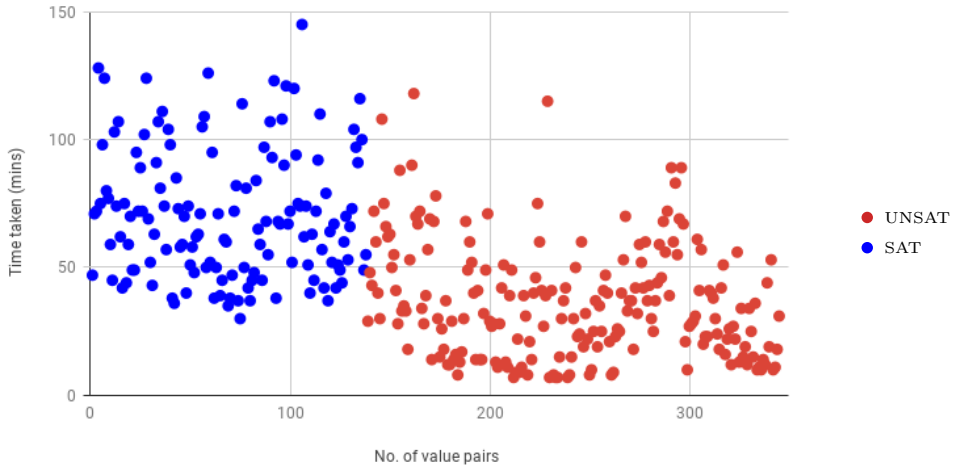


Figure 6.1: Scatterplot of time taken

It was observed that for most of the relaxation value pairs, the time taken to answer a single satisfiability query was between 30 - 180 mins. It was also observed that in general, the unsat queries returned faster than the sat ones which had solution subgraphs.

Even for a reasonable sized user defined window, the number of points that hosts at least one solution can be a large fraction of the window. For example, a window of size 20×20 has 400 possible points. If answering one satisfiability query for each one of them takes one hour on an average, the total time taken is 400 hours.

Number of solutions

Though timewise expensive, but this is still a possibility, if there were only one solution at every relaxation value pair. It was investigated to find how many solutions were present at each satisfiable point.

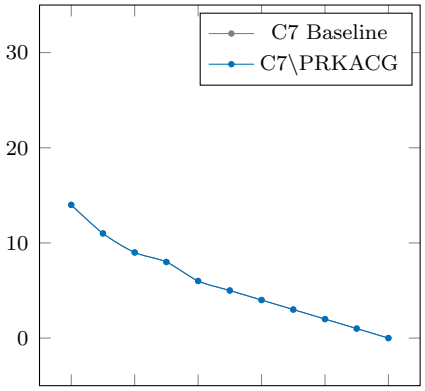
For the ITGB1-STAT3 experiments, it was found the smallest possible solutions had 62 nodes and 104 edges in the consistent subgraph. All possible solutions of this size were counted. There were 6188 solutions that differed in at least one node or edge. For larger subgraphs, the number is only expected to increase. Counting of solutions was also performed at some of the other value pairs that gave larger sized subgraphs. In some cases, number of solution subgraphs exceeded 40,000.

All possible node-edge relaxation value pairs and all possible solutions at each of those value pairs, generate a very large space of points to explore. Each solution takes non-trivial amount of time to obtain. Also, individual solutions that differ little from each other, does not provide a lot of insight. Hence the goal was to find significance of a node without explicitly enumerating all solutions at each satifiable point. Towards this effect the shifting of pareto-optimality curves was used to identify key players.

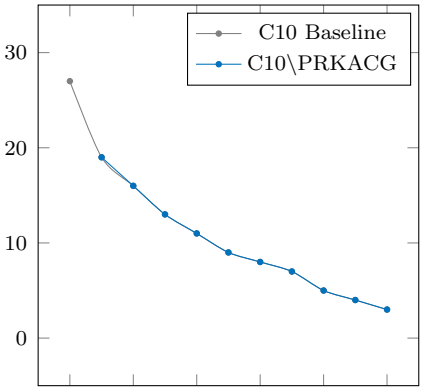
Pareto-optimal curves

In the plots shown below, the x-axes represent number of nodes relaxed, while the y-axes represent number of edges relaxed. Limit of 10 nodes relaxation was used, since relaxing too many nodes leaves out too much of the context provided by the microarray data. It beats the purpose of finding explanatory subgraphs that are consistent with the data.

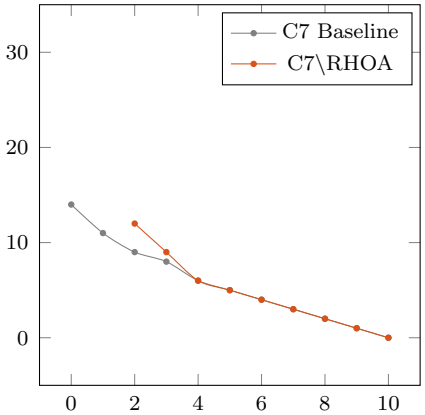
Clone 7 (C7) and Clone 10 (C10) are replicates of the original microarray experiment. We looked at the nodes with highest betweenness centrality and discussed with the biologists. The nodes of interest, for which we wanted to the see the shift in the Pareto-curves, were PRKACG, RHOA, PRKACA, TFG, PIK3CA, MAPK1, AKT3 and PLCD3. The PO-curves for baseline and exclusion of each of these nodes are shown in Fig. 6.2. Grey line denotes the baseline behavior and the colored one with the respective node excluded.



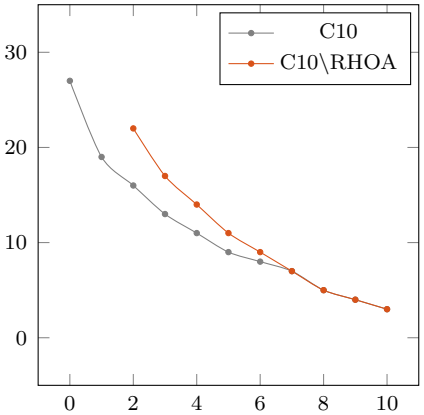
(a) C7 excluding PRKACG



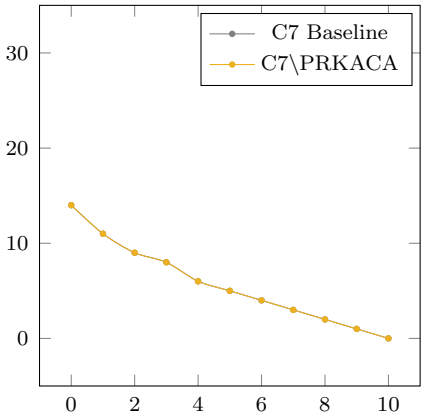
(b) C10 excluding PRKACG



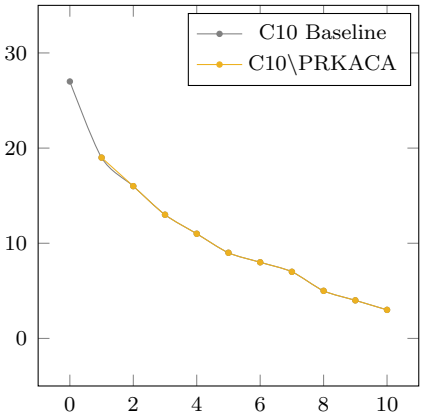
(c) C7 excluding RHOA



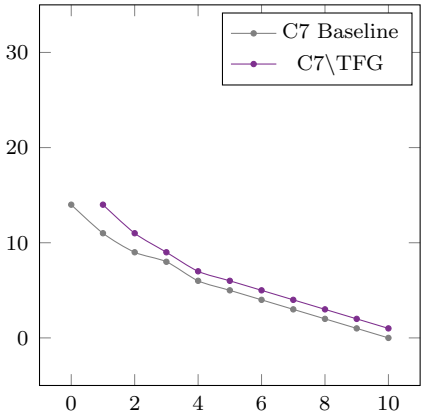
(d) C10 excluding RHOA



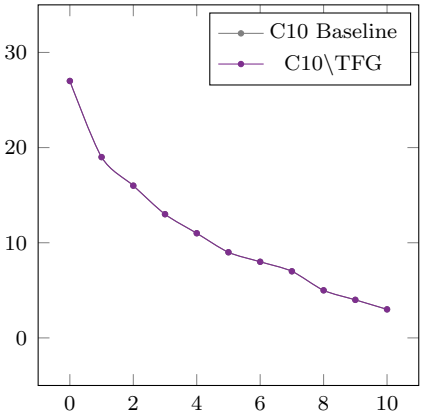
(e) C7 excluding PRKACA



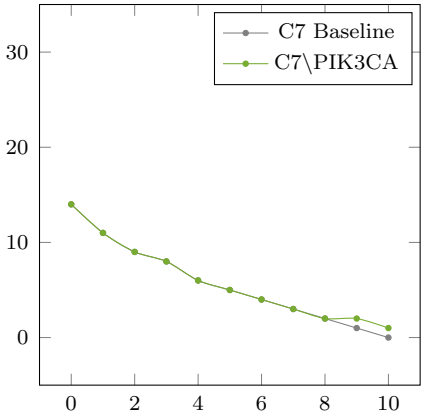
(f) C10 excluding PRKACA



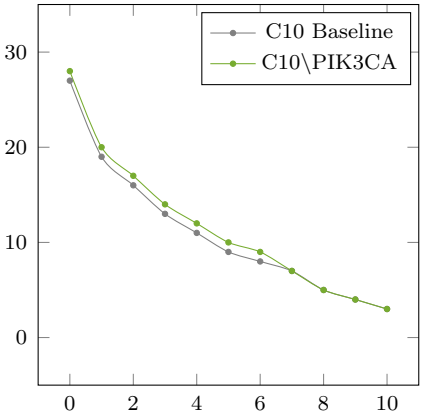
(g) C7 excluding TFG



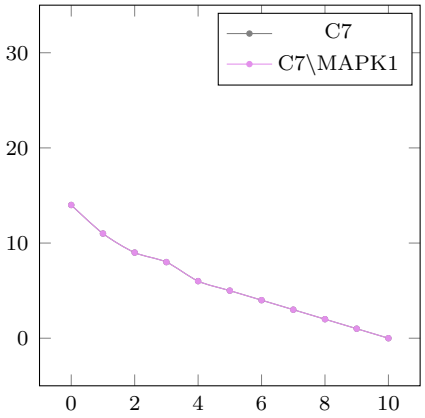
(h) C10 excluding TFG



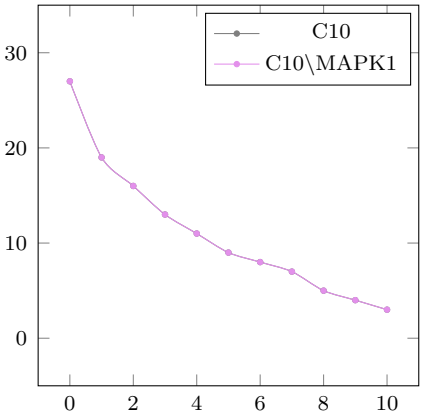
(i) C7 excluding PIK3CA



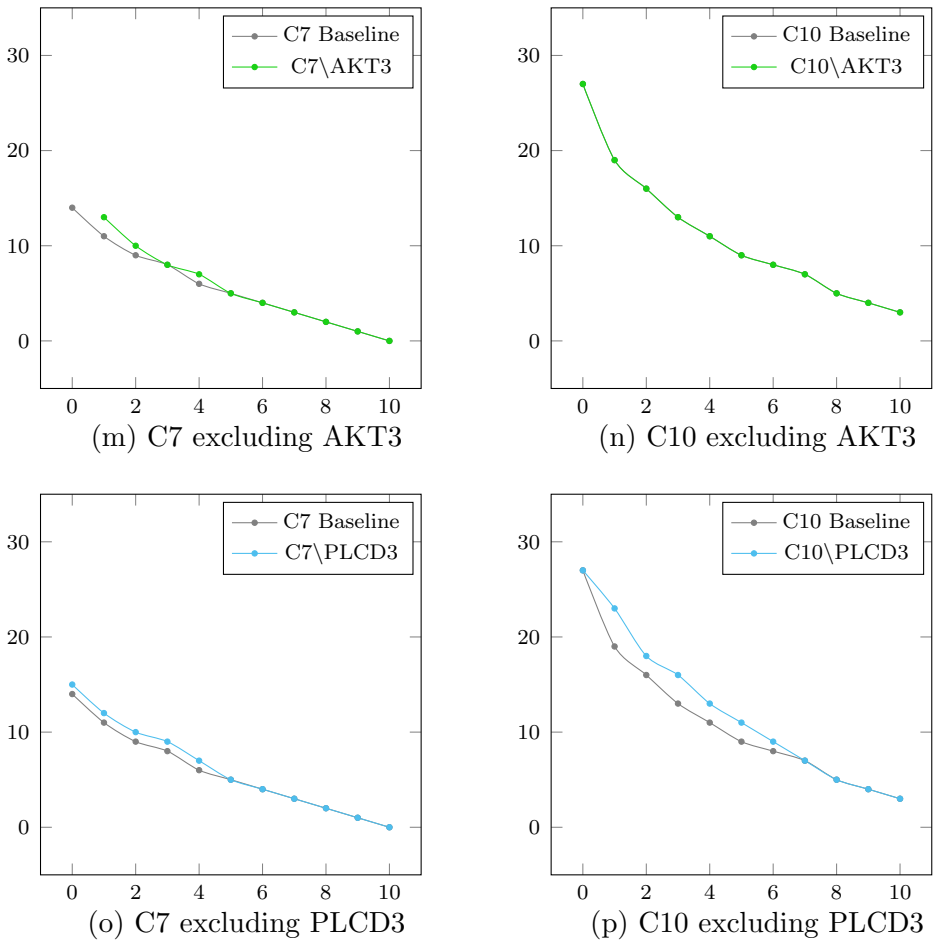
(j) C10 excluding PIK3CA



(k) C7 excluding MAPK1



(l) C10 excluding MAPK1



(m) C7 excluding AKT3 (n) C10 excluding AKT3

(o) C7 excluding PLCD3 (p) C10 excluding PLCD3

Figure 6.2: Individual plots

Genes RHOA, PIK3CA and PLCD3 seem to be important players for a range within the user defined window. TFG indicates to be playing an important role in the clone 7, but no such claim can be made in case of clone 10. None of the nodes stand out as an extremely important player. As discussed before, certain part of the curve can be brought into focus depending on the user defined window and claims can be made depending on the shift of the curve within that window. Overall, this dataset did not seem too sensitive for the ITGB1-STAT3 case.

Double exclusion

The effect was observed on excluding two nodes at the same time, trying to mimic a double knockout experiment. The behavior of excluding PIK3CA and AKT simultaneously as shown in Fig. 6.3

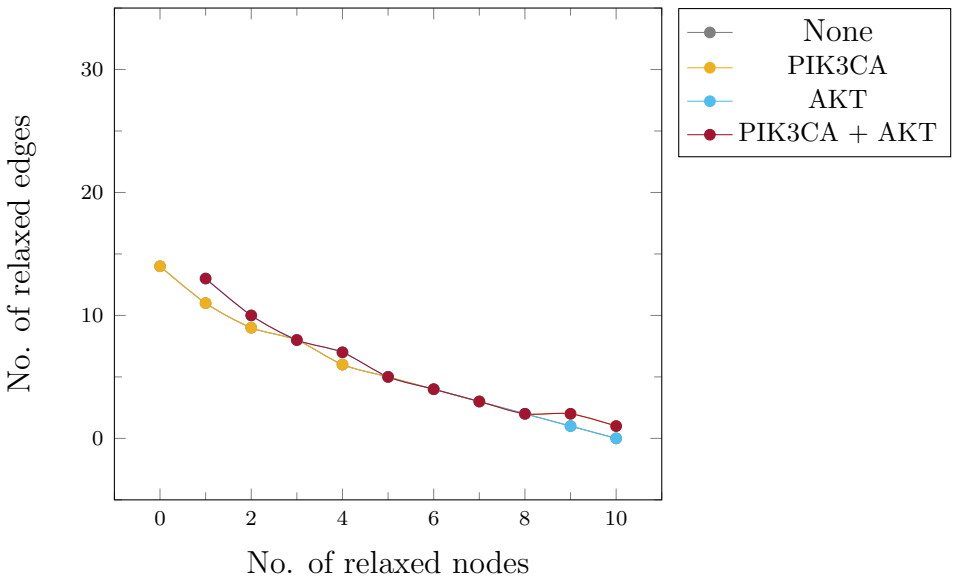


Figure 6.3: Relaxation bounds excluding AKT and PIK3CA in Vec4(7)

Since the individual nodes were not key players, it was not surprising to observe that the double exclusion also did not show considerable shift.

6.3 ITGB1-ACTB results

The set of experiments discussed next is very similar to the previous section, except that here target was the node ACTB. After the initial set, the biologists wanted to see the behavior of ACTB as the target. They also wanted to combine the Clone 7 and Clone 10 data from now onwards. A union of the two datasets was taken. The average value was used for common entries. This dataset will be referred to as P9 in this work. The initial bound selection and reachability traversals were performed in the same way as before.

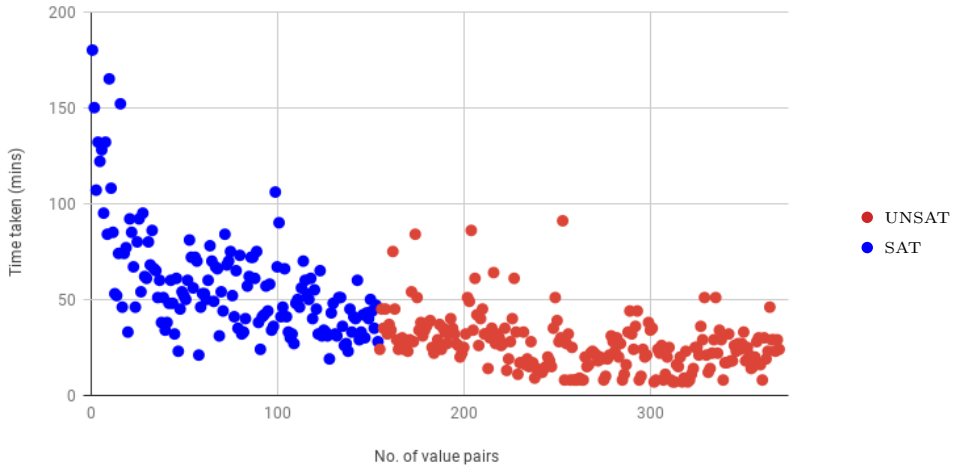
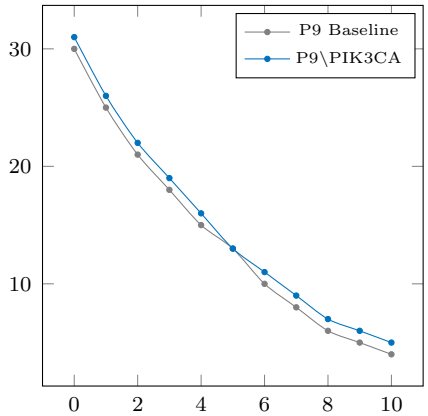


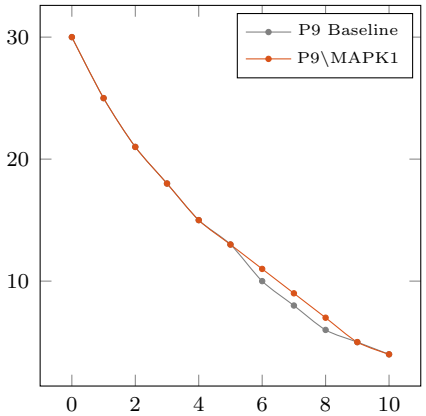
Figure 6.4: Scatterplot of time taken

Fig. 6.4 shows the time taken by the individual SAT calls. Like before, this case also shows that each call took a non-trivial amount of time, and hence solving for every solution at every sat point within the user-defined window is unrealistic.

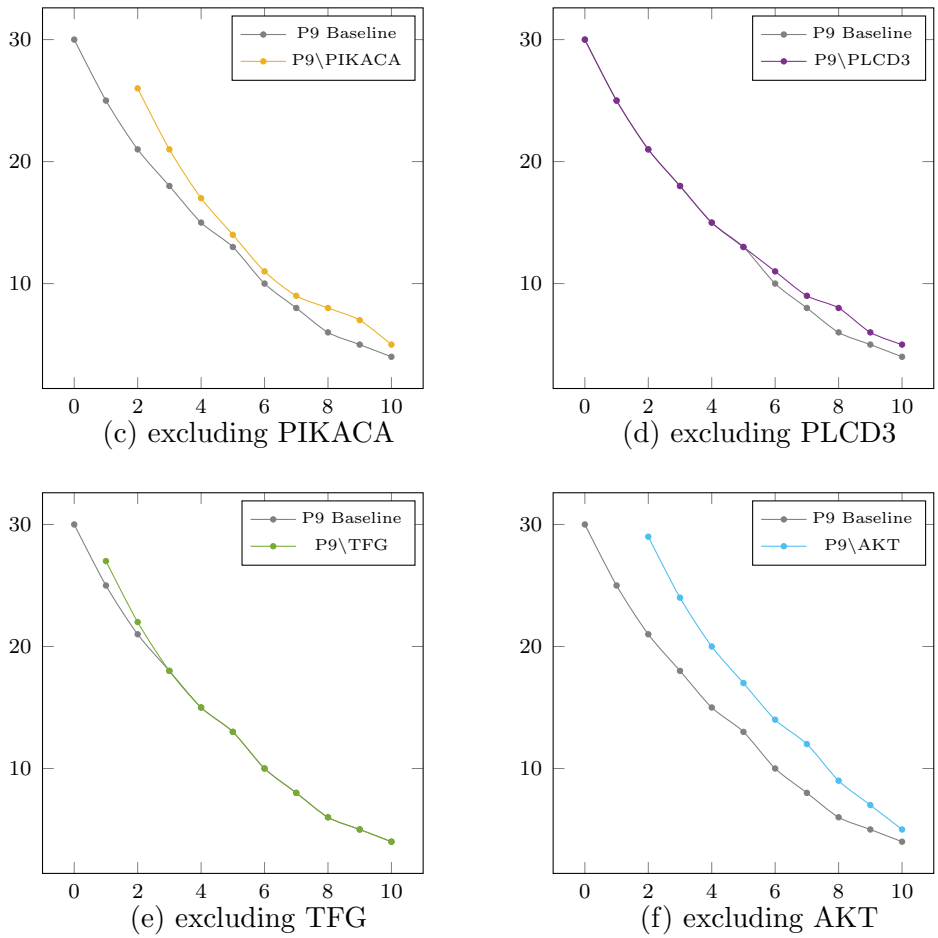
The top candidates in this case were PIK3CA, MAPK1, PIKACA, PLCD3, TFG and AKT. In Fig. 6.5 the comparison of the baseline PO-curve to the exclusion experiments PO-curve are shown for each of these six nodes. The x-axes represent number of nodes relaxed, while the y-axes represent number of edges relaxed.



(a) excluding PIK3CA



(b) excluding MAPK1



(c) excluding PIKACA (d) excluding PLCD3

(e) excluding TFG (f) excluding AKT

Figure 6.5: Individual plots

In the above figure, the curves for PIKACA and AKT show complete separation. Hence these two nodes were predicted to be significant in this setup.

Double exclusion

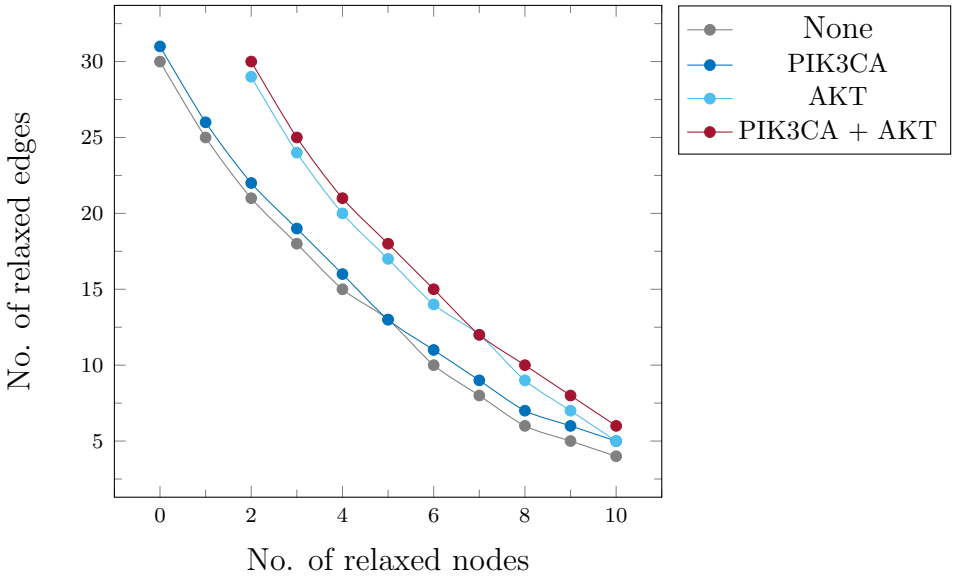


Figure 6.6: Relaxation bounds excluding AKT and PIK3CA

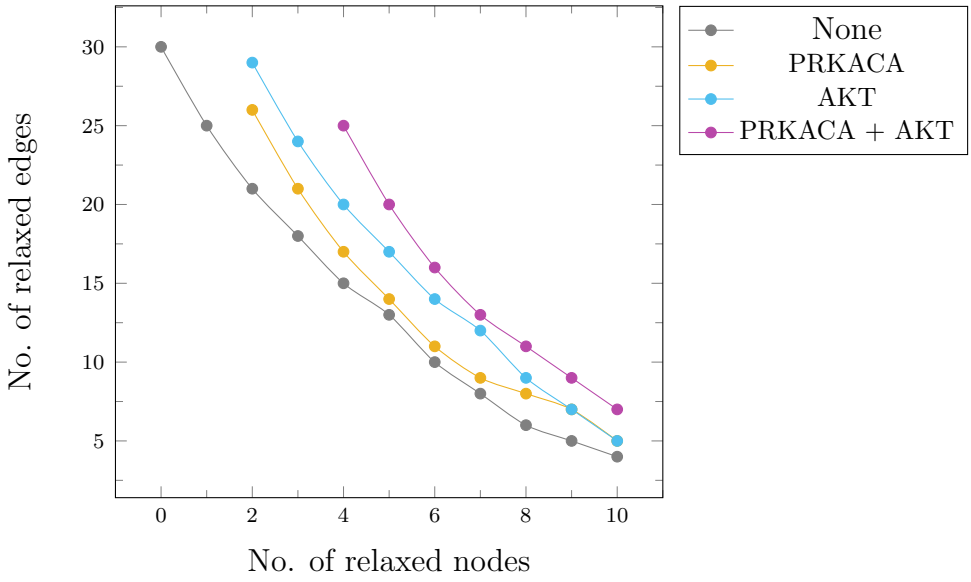


Figure 6.7: Relaxation bounds excluding AKT and PRKACA

As in the previous case, here too two double exclusion experiments were performed. These curves are shown in Fig. 6.6 and Fig. 6.7. The PRKACA+AKT exclusion curve showed complete shift. This result was in accordance with biological domain knowledge that these nodes are closely interdependent.

The ITGB1-STAT3 and ITGB1-ACTB cases gave enough validation from existing knowledge and hence confidence developed about the approach. Other experiments were perforemd that are dicussed in the coming sections.

6.4 TNFα to IKBa experiments

Next the performance of the P9 dataset was explored in a different setup. Our collaborators had previously demonstrated that PSMD9 overexpressing cells undergo TNF-induced IKBa degradation and NFκb activation. Hence they chose to study TNF as the common source for both IKBa and A20 targets. It must be noted that both IKBa and A20 engage in a feedback loop to inhibit NFκB activity. It was decided after consultation that experiments with TNFα as the source node, and IKBa and A20 as the targets, would be of interest.

Merged and reachable graphs

Bound	No. of nodes	No. of edges	Up reg	Down reg
2	3	4	1	0
3	5	9	1	0
4	32	95	4	1
5	72	353	12	5
6	153	881	28	14
7	297	1858	55	26
8	495	2917	100	53
9	591	3573	116	61
10	642	3812	120	67
11	661	3890	121	72
12	675	3918	123	73
13	675	3920	123	73
14	675	3920	123	73
15	675	3920	123	73

Table 6.2: Different reachability bounds and graph sizes

The different sizes of the reachable networks are shown in Table 6.2. Once again the reachability bound was selected to be 7.

Time required for generating solutions

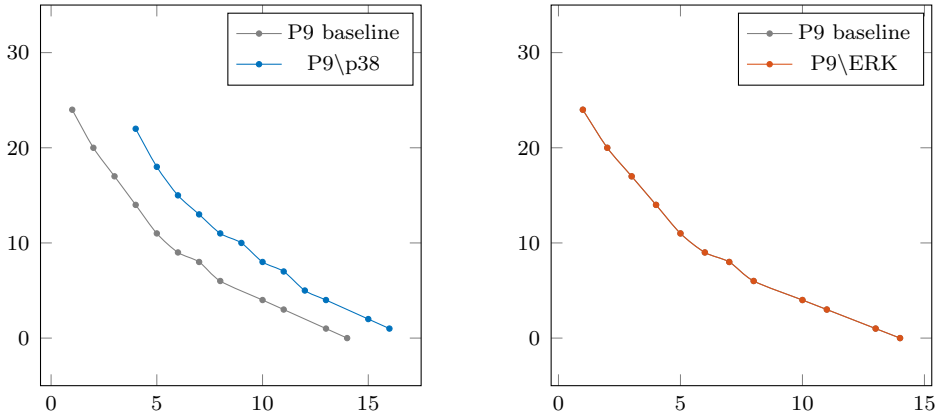
Source	Target	Excluded	# SAT calls	Total time
TNFα	IKBa	none	62	5 hrs
TNFα	IKBa	p38	72	5 hrs
TNFα	IKBa	ERK	62	2.6 hrs
TNFα	IKBa	PIK3CA	71	1.5 hrs
TNFα	IKBa	AKT	42	11 hrs

Table 6.3: Time taken in different exclusion experiments

In this and later experiments the algorithm for automated generation of the entire PO-curve as outlined in Section 5.5.2 was used. Henceforth the incremental solver of Z3 was used and thus the scatter-plots could not be reported in the same format as before. Starting from this set of experiments, the number of sat calls made and the time taken to generate the entire PO-curve are reported in Table 6.3. The user defined window size was from coordinates (0,30) to (0,30).

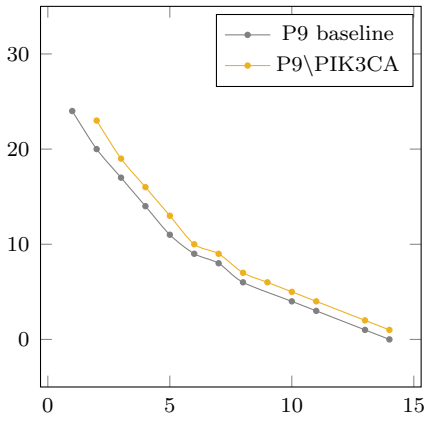
Pareto-optimal curves

The aim here was to see the baseline Pareto-optimal curve and compare that to the Pareto-optimal curves after exclusion of the nodes p38, ERK, PIK3CA and AKT. Fig. 6.8 shows how the Pareto-optimal curves changed with and witout these nodes being in the picture. The x-axes represent number of nodes relaxed, while the y-axes represent number of edges relaxed.

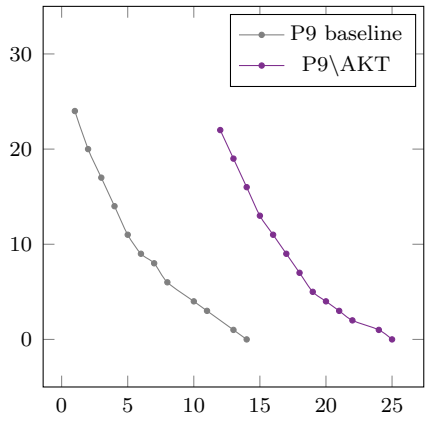


(a) P9 excluding p38

(b) P9 excluding ERK



(c) P9 excluding PIK3CA



(d) P9 excluding AKT

Figure 6.8: Individual plots

It was found that for all the exclusion experiments other than ERK, there was a complete shift in the Pareto-optimal curve. As mentioned earlier, it shows that the nodes p38, PIK3CA and AKT play a significant role. ERK, on the other hand, either does not play such an important role, or its effect is compensated by other players. With a curve that does not shift, much about the role of ERK cannot be predicted.

6.5 TNFα to IKBa expression vs activation experiments

From the previous set of results, it was decided to make two modifications for the further experiments. Firstly, to leave nodes ERK, P38 and JNK unmerged. Hence

it was needed to leave out 5 pathways that were merging those nodes.

Secondly, ERK not showing up as a definite player in the previous experiment was unexpected in the TNFα to IKBa setting. Hence at this point it was decided to re-examine the model of the system. It was pointed out by the collaborators, that it was necessary to distinguish between expression and activation of the target node IKBa. The behavior of the system and the key players are supposed to be sensitive towards the final fate - activation or expression - of target IKBa. Hence it was decided to make a distinction between activation and expression of target IKBa.

At the final step before reaching target IKBa no activation or inhibition edges were allowed when studying the expression of IKBa. Similarly when exploring the activation of IKBa, no expression or repression edges were allowed to be incident on IKBa. The Pareto-optimal curves were explored individually for these two cases.

Merged and reachable graphs

The KEGG pathways were merged for this experiment and few edges were manually added from NFKB signaling pathway to ensure connectivity. A reachability traversal was then performed with bound 7 between the above mentioned source and target nodes.

	No. of nodes	**No. of edges**
Merged graph	2469	10301
Reachable graph	296	1896

Table 6.4: Graph sizes in merged and reachable graphs

The reachable graph is the same for both activation and expression based PO curve generation. The overlap of differentially expressed data with the reachable graph was 53 up-regulated and 29 down-regulated entities.

Exclusion experiments

In this set of experiments, the goal was to identify:

(i) What are the key players (nodes) in activating target IKBa. It was expected to identify these players by exclusion experiments in the activation of IKBa scenario.

(ii) When IKBa is expressed, what are the key players (nodes) that bring about this expression. Exclusion experiments in the expression scenario should point to the important players.

The baseline PO curves for the two cases - expression and activation - were performed with the respective criteria explained later. Along with the baseline curves the exclusion curves were obtained for some nodes of interest. These were ERK, PIK3CD, PIK3CA, CREB3, PRKCA, TFG, JNK, RHOA, MAPK10, NFKB1, PLCD3, P38, AKT3 and NFKB1.

Expression vs activation of IKBa

Expression edges

Following are the expression/repression edges in Table 6.5, incident on IKBa(NFKBIA), that is the target. It was presumed that at least one of these edges are responsible for expressing IKBa in the phenotype.

Source	Edge type	Target
NFKB1	expression	NFKBIA
NFKB1+NFKB1	expression	NFKBIA

Table 6.5: Expression, repression or semantically similar edges incident on target IKBa(NFKBIA)

Activation edges

The following edges, in Table 6.6, have an activation/inhibition related semantics that are incoming to target IKBa. Again, the expectation was that, at least one of these would be needed to activate IKBa.

Source	Edge type	Target
CHUK	activation	NFKBIA
CHUK	phosphorylation	NFKBIA
CHUK	activation, phosphorylation	NFKBIA
CHUK+CHUK	activation, phosphorylation	NFKBIA
AKT3	activation, phosphorylation, indirect	NFKBIA
TAB1+MAP3K7	activation, phosphorylation, indirect	NFKBIA
CSNK2A1	activation, phosphorylation	NFKBIA
CASP8	activation, indirect	NFKBIA
EIF2AK1	activation, phosphorylation	NFKBIA
EIF2AK1	activation, phosphorylation, indirect	NFKBIA

Table 6.6: Activation, inhibition or semantically similar edges incident on target IKBa(NFKBIA)

Expression of target IKBa(NFKBIA)

To emulate expression of the target node, the following restrictions were put on the constraints.

(i) No edge from Table 6.5 were allowed to be relaxed. That is, if present, they would require to act in their original form.

(ii) No edge from Table 6.6 were allowed to be present in the solution.

These two conditions together ensured that, at the last step of the solution subgraph, only expression edges were incident on IKBa. Connectivity ensured at least one such edge would be present. Hence such solution graphs can be seen as mimicking the expression of IKBa.

Activation of target IKBa(NFKBIA)

Similarly, to construct the environment where IKBa is activated at the final step, the following restrictions were enforced.

(i) No edge from Table 6.5 were allowed to be present in the solution.

(ii) No edge from Table 6.6 were allowed to be relaxed. If present, they would need to be act in their original form.

These two conditions together ensured that, at the last step of the solution subgraph, only activation edges were incident on IKBA. Again, to ensure connectivity, at least one such activation (or other semantically equivalent) edge from Table 6.6 would be present. Hence such solution graphs can be seen as capturing the activation of IKBa.

Note that, in both these cases, the restriction was only on the last edges before IKBa is reached. Rest of the network may retain any type of edge and the solver is free to show them as relaxed where the need be.

Time required for generating solutions

As mentioned before, the total number of sat calls made to the solver were counted and the time taken to generate the full PO curve were calculated for the baseline in each of the expression and activation case, and also for the selected nodes exclusion cases in both the expression and activation scenarios.

	Expression		**Activation**	
Excluded node	**# SAT calls**	**Time taken**	**# SAT calls**	**Time taken**
None	63	9 hrs	64	15.6 hrs
ERK	63	15 hrs	64	25.8 hrs
PIK3CD	68	10.4 hrs	64	22.7 hrs
PIK3CA	68	14 hrs	64	18.5 hrs
CREB3	57	9.4 hrs	57	21.4 hrs
PRKCA	34	9.8 hrs	64	32 hrs
TFG	47	10 hrs	65	25 hrs
JNK	22	7.2 hrs	64	34 hrs
RHOA	24	9 hrs	58	21.7 hrs
MAPK10	24	11.8 hrs	64	31 hrs
PLCD3	63	35 hrs	60	46.8 hrs
P38	63	15 hrs	64	37 hrs
AKT	68	18.4 hrs	54	44 hrs

Table 6.7: Number of SAT calls and time taken for PO curve generation

As shown in Table 6.7, it was observed that in the expression case, an average of 51 sat calls were made and total time taken to generate the full PO curve was 13.3 hours. In the activation case 62 sat calls were made on an average and the total time taken to construct the full PO curve was about 28.8 hours.

Pareto-optimal curves

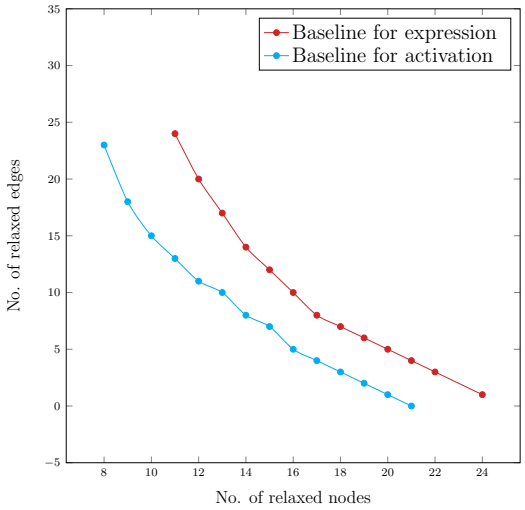
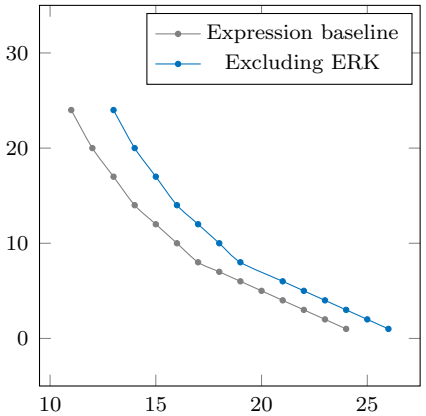


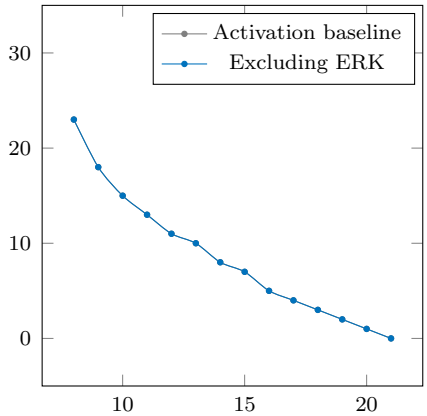
Figure 6.9: Baseline curves

The pareto-optimal (PO) curves were plotted for the expression and activation of IKBa cases. The baseline curves were found to differ. Fig. 6.9 shows the baseline curves in the two cases.

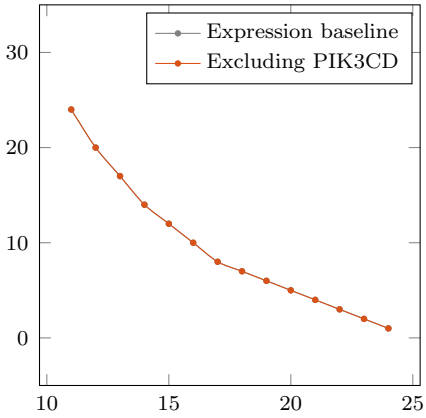
The PO curves for the exclusion of the selected nodes in the expression and activation setup are shown in Fig. 6.10. The x-axes represent number of nodes relaxed, while the y-axes represent number of edges relaxed.



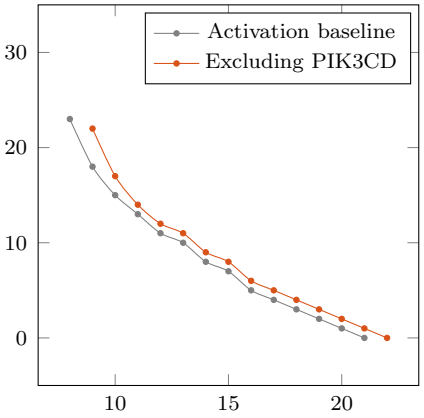
(a) Expression excluding ERK



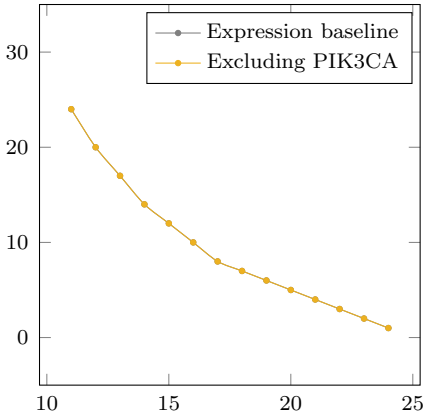
(b) Activation excluding ERK



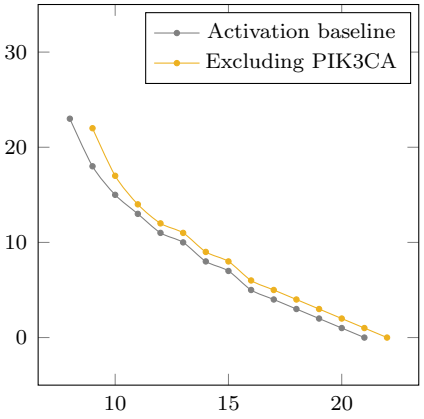
(c) Expression excluding PIK3CD



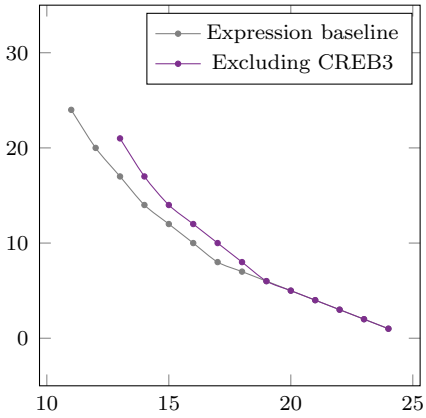
(d) Activation excluding PIK3CD



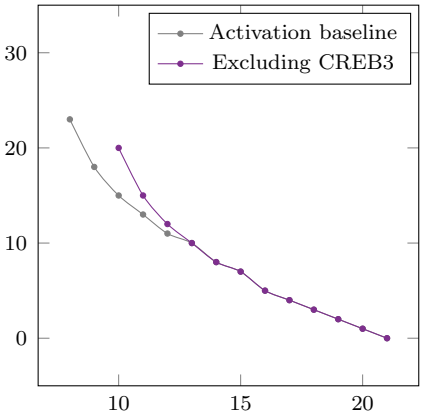
(e) Expression excluding PIK3CA



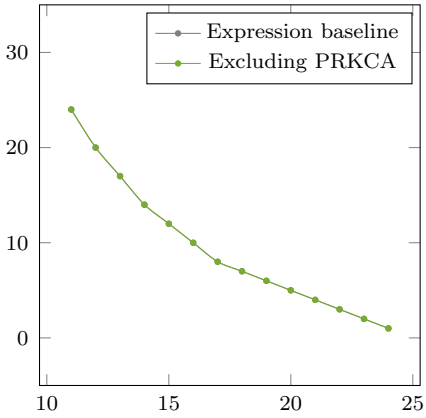
(f) Activation excluding PIK3CA



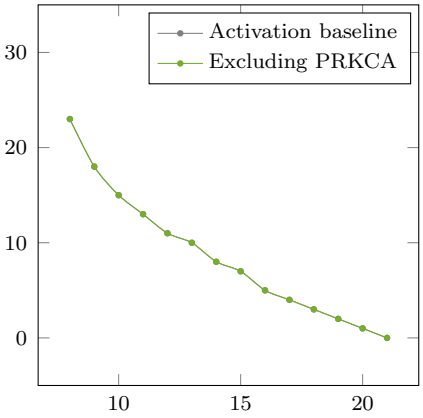
(g) Expression excluding CREB3



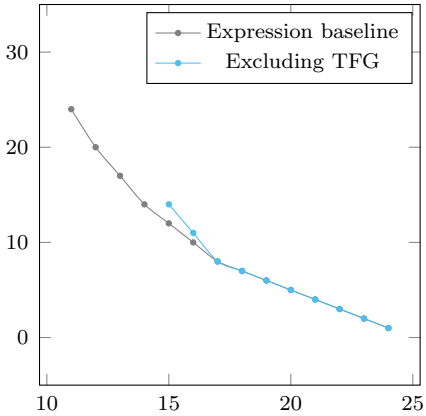
(h) Activation excluding CREB3



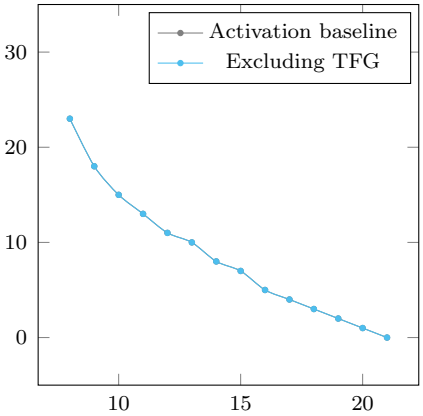
(i) Expression excluding PRKCA



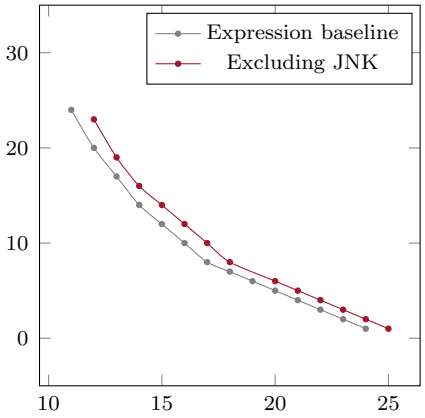
(j) Activation excluding PRKCA



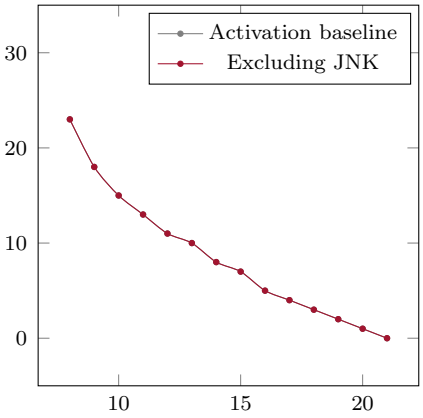
(k) Expression excluding TFG



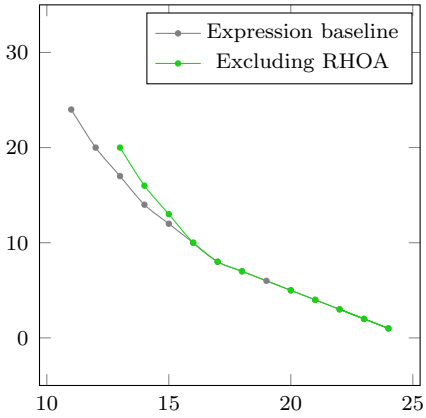
(l) Activation excluding TFG



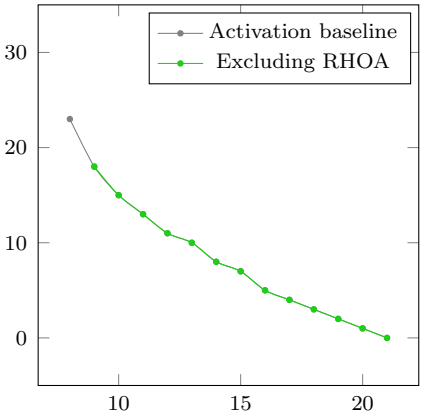
(m) Expression excluding JNK



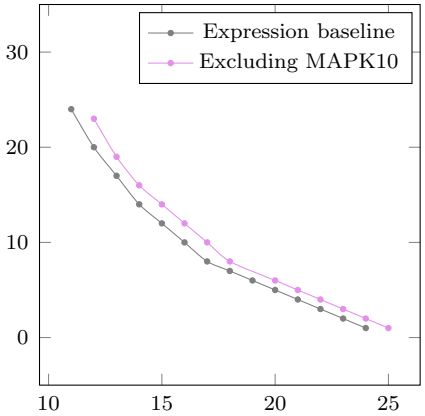
(n) Activation excluding JNK



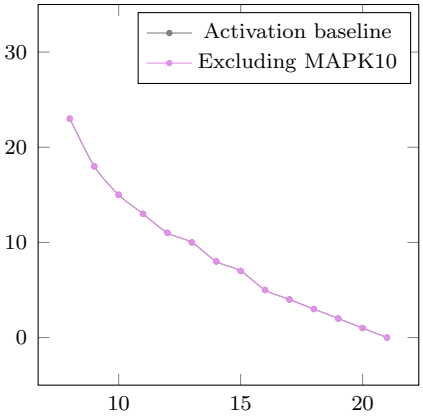
(o) Expression excluding RHOA



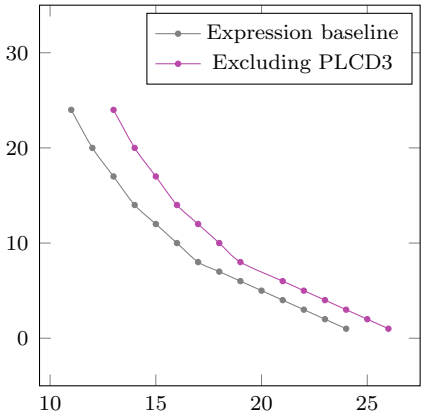
(p) Activation excluding RHOA



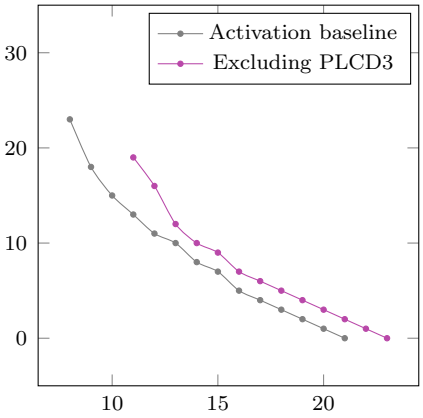
(q) Expression excluding MAPK10



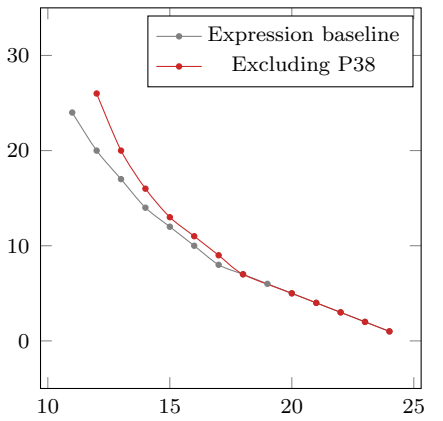
(r) Activation excluding MAPK10



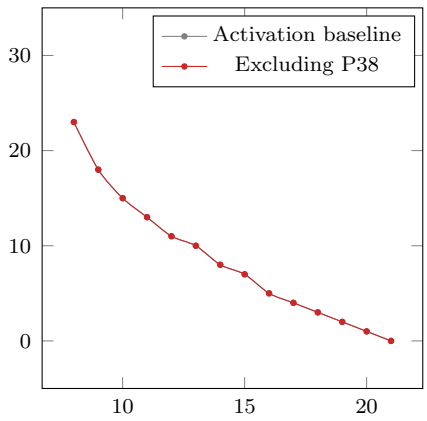
(s) Expression excluding PLCD3



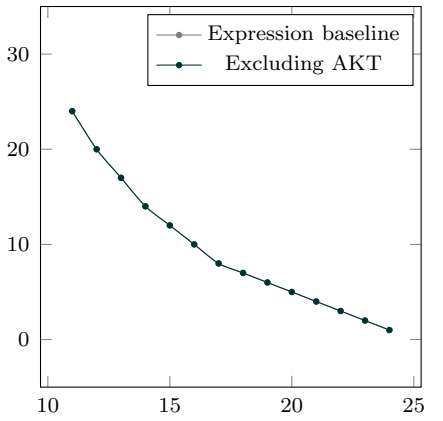
(t) Activation excluding PLCD3



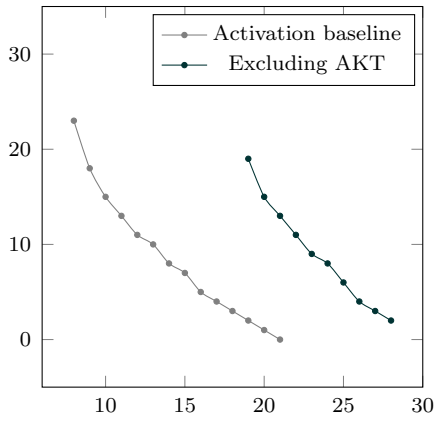
(u) Expression excluding P38



(v) Activation excluding P38



(w) Expression excluding AKT



(x) Activation excluding AKT

Figure 6.10: Exclusion PO curves

Of all the PO curves the most significant were ERK and AKT exclusions. As can be seen in Fig. 6.10a and Fig. 6.10b, ERK exclusion shows a complete separation from the baseline curve for the expression case. This suggests that ERK plays a definite role in the expression of target IKBa. Any such claim cannot be made for the activation case.

For AKT, Fig. 6.10w and Fig. 6.10x give exactly the opposite behavior. AKT surely plays a role in (in)activation of the target IKBa, while much cannot be said about its role in expression of IKBa.

The role of ERK in expression and role of AKT in (in)activation of IKBa was in accordance with the initial intuition of the biologists. They have been able to experimentally validate this significance in wet-lab as well. The biological validation is discussed later in Section 6.7.

NFKB1 exclusion

In the expression scenario, it appears that no solution is possible on the exclusion of NFKB1. Thus NFKB1 is absolutely necessary for paths reaching IKBa from TNFα. It is very much expected from biology that NFKB1 is absolutely necessary for expression of TNFα. Thus not getting any solutions on its exclusion gives more confidence on the method.

6.6 TNFα to A20 experiments

Next are the results of the exploration of this dataset with respect to another source target pair. Here also the analysis was expected to reveal important behavior in terms of significant nodes. Here results with TNFα as source and A20 target are shown.

Merged and reachable graphs

The selected 159 KEGG pathways were merged and a few edges from NFKB signaling pathway were manually added to ensure connectivity. Then a reachability with bound 7 between the above mentioned source and target nodes was performed. The reachable graph, as shown in Table 6.8, covered 45 up-regulated and 23 down-regulated nodes from pooled clone 7 and clone 10 data.

	No. of nodes	**No. of edges**
Merged graph	2469	10301
Reachable graph	255	1532

Table 6.8: Sizes of merged and reachable graphs

Expression vs activation of A20

Expression edges

Following are the expression/repression edges incident on the target node IKBa as shown in Table 6.9. It was presumed that at least one of these edges would be responsible for expressing A20 in the phenotype.

Source	Edge type	Target
NFKB1	expression	TNFAIP3
NFKB1+NFKB1	expression	TNFAIP3

Table 6.9: Expression, repression or semantically similar edges incident on target A20(TNFAIP3)

Activation edges

It was observed that the incoming edges to A20 were only of the expression type. There are no edges with activation or similar semantics in the 159 merged KEGG graphs.

Expression of target IKBA(NFKBIA)

Condition was:

(i) No edge from Table 6.9 were allowed to be relaxed. That is, if present, they would require to act in their original form.

Activation of target IKBA(NFKBIA)

Condition was:

(i) No edge from Table 6.9 were allowed to be present in the solution.

Hence for A20 as target there are only the expression of A20 case. Note that, the restriction was only on the last edges before A20 is reached. The rest of the network may retain any type of edge and the solver is free to show them as relaxed where the need be.

Time required for generating solutions

As before, here also the total number of sat calls made to the solver were counted and the time that was taken to generate the full PO curve was calculated for the baseline and with the selected nodes excluded cases.

Excluded node	# SAT calls	Time taken
None	56	21 mins
ERK	57	41 mins
AKT	52	22 mins
P38	54	19 mins

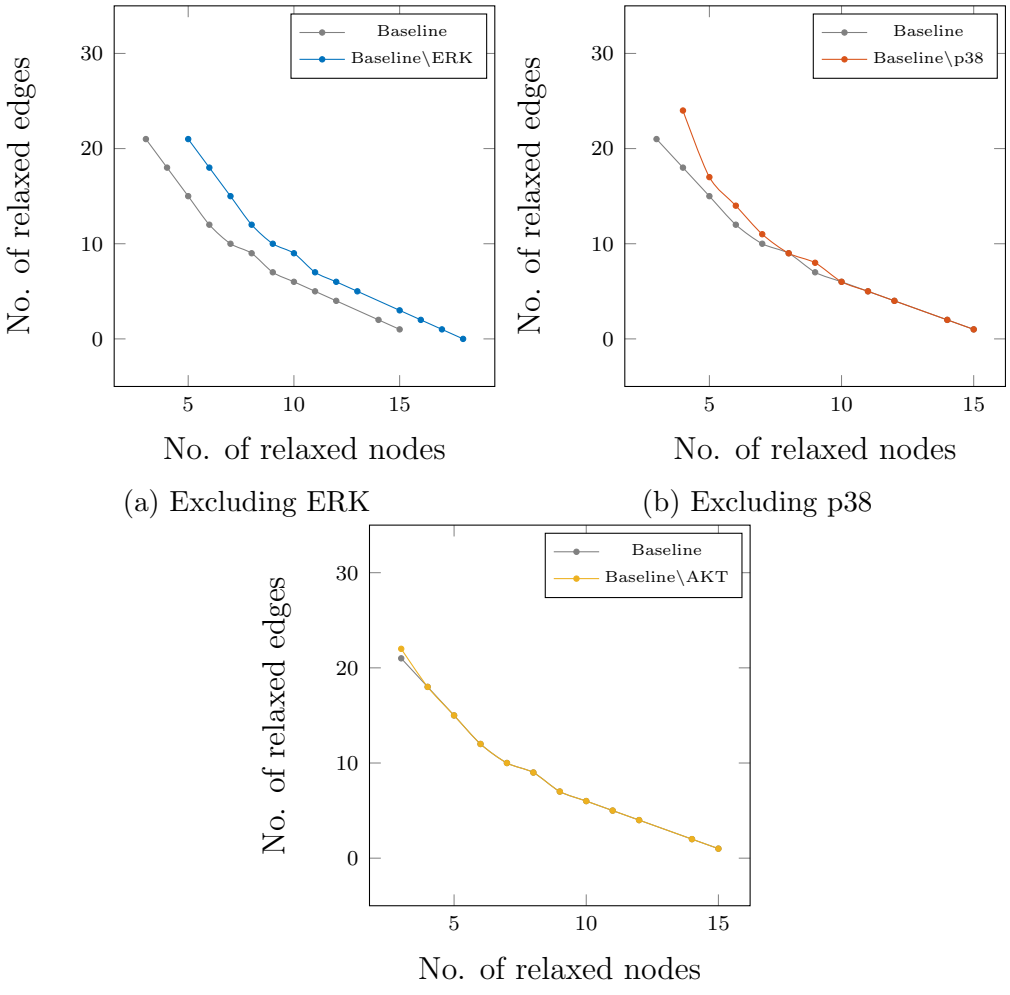
Table 6.10: Number of SAT calls and time taken for PO curve generation

As shown in Table 6.10, it was observed that in the expression case, an average of 55 sat calls were made, and the total time taken to generate the full PO curve was 25.75 mins.

It was argued earlier that it is infeasible to obtain potentially important actors by enumerating all solutions at all the points within the user-defined window. Here it is seen that the proposed method of calculating the Pareto-optimal curve is scalable enough.

Pareto-optimal curves

PO curves could be plotted for only the expression scenario.



(a) Excluding ERK

(b) Excluding p38

(c) Excluding AKT

Figure 6.11: Individual plots

The final incident edges on A20 are of type expression - these would have to be removed for the activation case. Since these two edges shown in Table 6.9 form the only connection, under the activation scenario, on discarding these edges, the target would not be connected to the source.

The Pareto-optimal (PO) curves were plotted for the expression case only, and hence, only the expression baseline curve was obtained.

With A20 as the target, it was of interest to see the Pareto-optimal curves after the exclusion of the nodes p38, ERK, AKT, and NFKB1. Fig. 6.11 shows how the Pareto-optimal curves changed with and without these node being present.

It was found that of all the exclusion experiments, only ERK showed a complete shift. As mentioned earlier, it shows that ERK plays a significant role.

NFKB1 exclusion

With source and target nodes to be TNFα and A20, respectively, it appears that no solution is possible on the exclusion of NFKB1. Thus NFKB1 is necessary for paths reaching A20 from TNFα. Here too, it is highly biologically intuitive that NFKB1 is necessary for expression of A20. Thus not getting any solutions on its exclusion gives more confidence on the method.

6.7 Biological validation

Among nodes explored for functionality, wet-lab investigations have been completed by our collaborators for ERK and AKT kinases, which showed PSMD9-induced phosphorylation. Since the negative feedback loop involving both IKBa and A20 control NFKB activation and target gene expression, experiments were also conducted after separating the composite graphs into activation and expression graphs. Phosphorylation impacts (in)activation status of transcription factors and hence must be integrated into gene expression studies. Indeed, excluding ERK from composite graphs did not induce Pareto shift whereas separating them into activation and expression graphs did. ERK exclusion, but not AKT exclusion induced a Pareto shift indicating its requirement in both IKBa and A20 expression. Only AKT induced IKBa (in)activation Pareto shift.

Our biologist colleagues tested ERKs significance in IKBa and A20 gene expression using qPCR. A two-fold decrease in A20 mRNA was observed in PSMD9 overexpression cells upon ERK inhibition [p=0.03] whereas AKT inhibition did not impact IKBa or A20 mRNA levels, a trend consistent even upon TNFα stimulation (t=3h). The lack of impact on IKBa mRNA levels is likely due to yet unexplored PSMD9-specific effects. The NFKB-dependence for IKBa or A20 expression was evident from the lack of solutions upon its exclusion. The routinely used PD98059

and LY294002 signaling inhibitors achieved ERK (∼100%) and AKT (∼90%) phosphorylation inhibition, respectively, at recommended IC_{50} values. They may have off-target effects. Importantly, these inhibition-dependent mRNA level changes are PSMD9-specific, consistent with computational predictions.

Chapter 7

Conclusion

In this work, we present a novel problem formulation to capture the functional significance of a node in an interaction pathway between a stimulus and a target observation, in a highly noisy environment with minimal experimental data and using publicly available pathway databases. Our definition comes closest to computational simulation of a knockout experiment that is classically done to establish the functional significance of a node in wet-lab experiments. We validate this by some wet-lab experiments and domain knowledge.

To validate the results, we used a combination of domain knowledge, existing literature, and wet-lab experiments. Indeed, the most expensive but most reliable of these are the wet-lab experiments. The wet-lab experiment consisted of a knockout experiment where a particular reagent is used to inactivate the candidate actor and then check if adding the stimulus in this condition resulted in a change of the target observation. This approach was used to establish the functional significance of AKT and hence validate our approach above. For the others, except ERK, we used domain knowledge and literature to validate the functional significance of these. For ERK, we also performed another wet-lab experiment which showed its significance in the experiment. Note that our computational experiments above, however, do not show any shift. As discussed earlier, this may be expected, as the condition of the Pareto-curve shift is a sufficient but not necessary condition for functional significance.

This thesis leaves scope for much exciting work to be done in the future. We mention a few here.

1. **Optimal ordering of the experiments.** Each focused experiment is expected to give more information than what was available before performing that experiment. It will be a challenging CS problem to try to suggest the optimal ordering of the experiments. The restrictions of experiments are that

(i) Only one node can be knocked out in every experiment. If node A is knocked out in the first experiment and then node it is decided to knock out node B, then knocking out node B would be a whole new experiment, where A will be present.

(ii) Knocking out more than one node simultaneously. This experiment is complicated to perform, and the results may not be clear to interpret. Also, not all pairs of nodes can be knocked out together - some nodes can be knocked out only with some specific nodes. Usually, a maximum of two nodes might be knocked out if they are compatible.

(iii) Even if different nodes are knocked out, the final observation should be on the same phenotype (target node).

(iv) These restrictions make the CS problem of finding an optimal ordering even more challenging.

2. **Incorporate time-series data.** The microarray data represents a snapshot of the system at one particular time point. It can only show correlation or coregulation among genes, but cannot say anything about a causal relationship between any two genes. Hence there is no directionality, only a bunch of genes that get up-(or down)regulated as a result of a stimulus, such as a perturbation in, suppose, the PSMD9 gene. However, a sense of causality arises when the same system is observed over a while, at least at a few time-points. A gene showing a perturbation at an earlier time point might be causing changes in the expression of a gene that is reflected at a later time. For this suitable constraints have to be written that will successfully capture the behavior of the genes over the entire time period.

3. **Tracking relaxed nodes/edges at each Pareto-optimal point.** In the current work, the solutions at a Pareto-optimal point has been considered equivalent. The only concern has been the presence or absence of solutions at some value of (n, e). The next step would be to distinguish the different sets of nodes and edges relaxed and not just the number. This distinction will help provide more refined and targetted solutions.

4. **Refinement and focussed experiments.** Results from the previous point would guide the search for the ideal solution closer to the parts of the networks that are most likely to hold explanations of observed biological behavior. The next step would be to refine this further. From the thousands of nodes and edges, biologists can be asked for more targeted experiments to be performed on these few nodes/edges. Among the targeted experiments would be analyses that provide quantitative information about the relevant part of the network, eg., rate of reaction of a specific interaction (edge), rate

of change of concentration of specific proteins at specific times, and others. This kind of specific quantitative data would provide an opportunity to apply some more rigorous analysis techniques.

Appendices

Appendix A

List of 163 KEGG pathways

Table A.1: Table of 163 KEGG pathways used in analysis

Pathway id	Pathway name
hsa03320	PPAR signaling pathway
hsa03460	Fanconi anemia pathway
hsa04010	MAPK signaling pathway
hsa04012	ErbB signaling pathway
hsa04014	Ras signaling pathway
hsa04015	Rap1 signaling pathway
hsa04020	Calcium signaling pathway
hsa04022	cGMP-PKG signaling pathway
hsa04024	cAMP signaling pathway
hsa04060	Cytokine-cytokine receptor interaction
hsa04062	Chemokine signaling pathway
hsa04064	NF-kappa B signaling pathway
hsa04066	HIF-1 signaling pathway
hsa04068	FoxO signaling pathway
hsa04080	Neuroactive ligand-receptor interaction
hsa04110	Cell cycle
hsa04114	Oocyte meiosis
hsa04115	p53 signaling pathway
hsa04122	Sulfur relay system
hsa04130	SNARE interactions in vesicular transport
hsa04140	Autophagy
hsa04141	Protein processing in endoplasmic reticulum
hsa04150	mTOR signaling pathway
hsa04151	PI3K-Akt signaling pathway

Table A.1: Table of 163 KEGG pathways used in analysis

hsa04152	AMPK signaling pathway
hsa04210	Apoptosis
hsa04261	NOD-like receptor signaling pathway
hsa04270	Vascular smooth muscle contraction
hsa04310	Wnt signaling pathway
hsa04330	Notch signaling pathway
hsa04340	Hedgehog signaling pathway
hsa04350	TGF-beta signaling pathway
hsa04360	Axon guidance
hsa04370	VEGF signaling pathway
hsa04380	Osteoclast differentiation
hsa04390	Hippo signaling pathway
hsa04510	Focal adhesion
hsa04512	ECM-receptor interaction
hsa04514	Cell adhesion molecules (CAMs)
hsa04520	Adherens junction
hsa04530	Tight junction
hsa04540	Gap junction
hsa04550	Signaling pathways regulating pluripotency of stem cells
hsa04610	Complement and coagulation cascades
hsa04611	Platelet activation
hsa04612	Antigen processing and presentation
hsa04620	Toll-like receptor signaling pathway
hsa04621	NOD-like receptor signaling pathway
hsa04622	RIG-I-like receptor signaling pathway
hsa04623	Cytosolic DNA-sensing pathway
hsa04630	Jak-STAT signaling pathway
hsa04650	Natural killer cell mediated cytotoxicity
hsa04660	T cell receptor signaling pathway
hsa04662	B cell receptor signaling pathway
hsa04664	Fc epsilon RI signaling pathway
hsa04666	Fc gamma R-mediated phagocytosis
hsa04668	TNF signaling pathway
hsa04670	Leukocyte transendothelial migration
hsa04672	Intestinal immune network for IgA production
hsa04710	Circadian rhythm
hsa04713	Circadian entrainment
hsa04720	Long-term potentiation
hsa04722	Neurotrophin signaling pathway

Table A.1: Table of 163 KEGG pathways used in analysis

hsa04723	Retrograde endocannabinoid signaling
hsa04724	Glutamatergic synapse
hsa04725	Cholinergic synapse
hsa04726	Serotonergic synapse
hsa04727	GABAergic synapse
hsa04728	Dopaminergic synapse
hsa04730	Long-term depression
hsa04740	Olfactory transduction
hsa04742	Taste transduction
hsa04744	Phototransduction
hsa04750	Inflammatory mediator regulation of TRP channels
hsa04810	Regulation of actin cytoskeleton
hsa04910	Insulin signaling pathway
hsa04911	Insulin secretion
hsa04912	GnRH signaling pathway
hsa04913	Ovarian steroidogenesis
hsa04914	Progesterone-mediated oocyte maturation
hsa04915	Estrogen signaling pathway
hsa04916	Melanogenesis
hsa04917	Prolactin signaling pathway
hsa04918	Thyroid hormone synthesis
hsa04919	Thyroid hormone signaling pathway
hsa04920	Adipocytokine signaling pathway
hsa04921	Oxytocin signaling pathway
hsa04930	Type II diabetes mellitus
hsa04932	Non-alcoholic fatty liver disease (NAFLD)
hsa04940	Type I diabetes mellitus
hsa04950	Maturity onset diabetes of the young
hsa04960	Aldosterone-regulated sodium reabsorption
hsa04961	Endocrine and other factor-regulated calcium reabsorption
hsa04962	Vasopressin-regulated water reabsorption
hsa04970	Salivary secretion
hsa04971	Gastric acid secretion
hsa04972	Pancreatic secretion
hsa04973	Carbohydrate digestion and absorption
hsa04974	Protein digestion and absorption
hsa04976	Bile secretion
hsa04978	Mineral absorption
hsa05010	Alzheimer's disease

Table A.1: Table of 163 KEGG pathways used in analysis

hsa05012	Parkinson's disease
hsa05014	Amyotrophic lateral sclerosis (ALS)
hsa05016	Huntington's disease
hsa05020	Prion diseases
hsa05030	Cocaine addiction
hsa05031	Amphetamine addiction
hsa05032	Morphine addiction
hsa05033	Nicotine addiction
hsa05034	Alcoholism
hsa05100	Bacterial invasion of epithelial cells
hsa05110	Vibrio cholerae infection
hsa05120	Epithelial cell signaling in Helicobacter pylori infection
hsa05130	Pathogenic Escherichia coli infection
hsa05131	Shigellosis
hsa05132	Salmonella infection
hsa05133	Pertussis
hsa05134	Legionellosis
hsa05140	Leishmaniasis
hsa05142	Chagas disease (American trypanosomiasis)
hsa05143	African trypanosomiasis
hsa05144	Malaria
hsa05145	Toxoplasmosis
hsa05146	Amoebiasis
hsa05150	Staphylococcus aureus infection
hsa05152	Tuberculosis
hsa05160	Hepatitis C
hsa05161	Hepatitis B
hsa05162	Measles
hsa05164	Influenza A
hsa05166	Human T-cell leukemia virus 1 infection
hsa05168	Herpes simplex infection
hsa05169	Epstein-Barr virus infection
hsa05200	Pathways in cancer
hsa05202	Transcriptional misregulation in cancer
hsa05203	Viral carcinogenesis
hsa05204	Chemical carcinogenesis
hsa05205	Proteoglycans in cancer
hsa05206	MicroRNAs in cancer
hsa05210	Colorectal cancer

Table A.1: Table of 163 KEGG pathways used in analysis

hsa05211	Renal cell carcinoma
hsa05212	Pancreatic cancer
hsa05213	Endometrial cancer
hsa05214	Glioma
hsa05215	Prostate cancer
hsa05216	Thyroid cancer
hsa05217	Basal cell carcinoma
hsa05218	Melanoma
hsa05219	Bladder cancer
hsa05220	Chronic myeloid leukemia
hsa05221	Acute myeloid leukemia
hsa05222	Small cell lung cancer
hsa05223	Non-small cell lung cancer
hsa05310	Asthma
hsa05320	Autoimmune thyroid disease
hsa05321	Inflammatory bowel disease (IBD)
hsa05322	Systemic lupus erythematosus
hsa05330	Allograft rejection
hsa05332	Graft-versus-host disease
hsa05412	Arrhythmogenic right ventricular cardiomyopathy (ARVC)
hsa05414	Dilated cardiomyopathy (DCM)
hsa05416	Viral myocarditis

Appendix B

Tool documentation

B.1 Operations and commands

Following are the commands available in the tool and their usage. Calling this commands in the order of the described process yeilds the desired output.

B.1.1 Read graph from xml file

`rgx`

Description

Reads a file written in KGML(XML) format and generates a graph in our internal representation. If successful, returns the newly generated graph id, otherwise returns -1.

Arguments

xml-file	path to KGML (XML) file that stores the graph
tool-written option	'Y' or 'y' if the xml file was earlier written by the tool; any other character otherwise
split node threshold	number of ids in a node such that the node will not be split if it has $\geq$ that many ids (unless specifically asked to split)

Input

```
> rgx <xml-file>
> Tool-written? (y/n): <option>
> Split node threshold: <threshold>
```

Output

```
> New graph id: <graph-id>
```

Example

```
> rgx test1.xml
> Tool-written? (y/n) : n
> Split node threshold: 1
> New graph id: 1
```

B.1.2 Merge graphs

`merge`

Description

Merges a number of graphs into one graph which is the union of the nodes and edges. If the nodes have overlapping sets of ids, then, unless specified, the ids will be merged into one node. For edges, if two edges between the same pair of nodes have different sets of labels on them, then the edges are retained separately as parallel edges between those two nodes. If the annotation labels are exactly same then only one edge is retained. If successful, returnes id of the newly created merged graph, otherwise returns -1.

Arguments

list of graph ids	list of already generated graph-ids to be merged terminated by '-1'
map of nodes	files containing maps among nodes from different databases terminated by '-1'

Input

```
> merge
> Enter graphs (end by -1): <graph1-id> <graph2-id> -1
> Enter map file names (end by -1): <map1> <map2> -1
```

Output

```
> New graph id: <merged-graph-id>
```

Example

```
> merge
> Enter graphs (end by -1): 1 2 -1
> Enter map file names (end by -1): -1
> New graph id: 3
```

B.1.3 Merge graphs from file

`mff`

Description

Similar to the command 'merge', this command merges a list of graphs that are provided as an xml file. It first constructs the individual graphs from the xml files and then merges them into one final graph. Behavior of merge is same as the previous command. It splits a node if it has more than specified number of ids. otherwise merges the ids into one node. If successful, returnes id of the newly created merged graph, otherwise returns -1.

Arguments

list-file	list of paths of xml files to be merged
file-format	either 'xml' or 'sbml' depending on format of input graphs
split node threshold	Number of ids above which the node will not be split unless otherwise specified
previously written option	'Y' or 'y' if the input xml files were previously written by the tool, otherwise any other character
map of nodes	files containing maps among nodes from different databases terminated by '-1'

Input

```
> mff <list-file>
> Enter file format (xml/sbml): <file-format>
> Split node threshold: <threshold>
> Were these graph earlier written by this tool?,
 'y' or 'Y' for yes, otherwise any other key: <option>
> Enter map file names (end by -1): <map1> <map2> -1
```

Output

```
Read graph: 1 from file <file_1>
Read graph: 2 from file <file_2>
.
.
.
Read graph: n from file <file_n>

merging graphs: 1 and 2
.
.
.

> New graph id: <merged-graph-id>
```

Example

```
> mff list.txt
> Enter file format (xml/sbml): xml
> Split node threshold: 1
```

```
> Were these graph earlier written by this tool?,
 'y' or 'Y' for yes, otherwise any other key: n
> Enter map file names (end by -1): -1

Read graph: 1 from file test1.xml
Read graph: 2 from file test2.xml
Read graph: 3 from file test3.xml
Read graph: 4 from file test4.xml
Read graph: 5 from file test5.xml

merging graphs: 1 and 2
merging graphs: 6 and 3
merging graphs: 7 and 4
merging graphs: 8 and 5

> New graph id: 9
```

B.1.4 Write graph to xml file

---

wgx

Description

Writes a graph in the internal representation onto an external file in the xml format. This command is mainly used to write some important graph like the merged graph, or the reachable graph onto a file that can be read again without repeating the constructing operation. Writes the xml file and prints the number of nodes and egdes on the terminal.

Arguments

graph id	id of the graph to be written to xml file
file path	path where the xml file would be written
id string	string to store any information about the graph

Input

```
> wgx <graph-id>
> Enter the file path to store the graph in xml format: <file-path>
> Enter the pathway name: <id-string>
```

Output

```
<num-of-nodes> nodes written to xml file
<num-of-edges> edges written to xml file
```

Example

```
> wgx 9
> Enter the file path to store the graph in xml format: test_merged.xml
> Enter the pathway name: test
281 nodes written to xml file
507 edges written to xml file
```

B.1.5 Get size of a graph

`size`

Description

Prints the number of nodes and the number of edges of the graph-id provided. Among nodes, this command also gives details like number of genes, complexes, compounds and others. It also prints the number of isolated nodes, that is, the nodes that do not have any edge associated with it.

Arguments

graph-id id of the graph whose size we want to see

Input

```
> size <graph-id>
```

Output

```
> Number of nodes: <num-of-nodes> <details-of-nodes>
> Number of edges: <num-of-edges>
```

Example

```
> size 9
Number of nodes: 281 (251 genes, 16 complexes, 11 compounds, 3 others)
    0 isolated
Number of edges: 507
```

B.1.6 Forward-backward reachability

`fb_rch`

Description

This command performs the forward from source traversal upto the given bound, excluding the specified nodes, followed by backward reachability traversal from target. It also eliminates any path of length > bound within the graph restricted between given source and target nodes. It follows the algorithm outline in Algorithm 1. If successful, returnes id of the newly created reachable graph, otherwise returns -1.

Arguments

graph id	id of graph on which reachability is to be performed
source nodes	list of hsa-ids of the source nodes in the reachability analysis
target nodes	list of hsa-ids of the target nodes in the reachability analysis
excluded nodes	nodes to be avoided in the reachability traversal
bound	number of hops from source nodes to be explored

Input

```
> fb_rch <graph-id>
> Source node ids list ending with -1: <source-node-1> <source-node-2> ... -1
> Target node ids list ending with -1: <target-node-1> <target-node-2> ... -1
> Excluded node ids list ending with -1: <excluded-node> -1
> Bound: <bound>
```

Output

```
Forward reachable graph id: <forward-reachable-graph-id>
Reachability Graph: <reachable-graph-id>
```

Example

```
> fb_rch 9
> Source node ids list ending with -1: hsa1956 -1
> Target node ids list ending with -1: hsa5599 -1
> Excluded node ids list ending with -1: hsa2885 -1
> Bound: 10
Forward reachable graph id: 10
Reachability Graph: 11
```

B.1.7 Get up and down regulated nodes from fold change file

`cudf`

Description

This command takes the fold change file data and create up-reg and down-reg files from fold change file using the provided thresholds. If successful, the command will generate the two files with lists of up-reg nodes in one and down-reg nodes in the other.

Arguments

fold change file	tab separated two column file where each row contains the hsa-id of one node and its log-fold change value
up reg cutoff	a single float value that is to be treated as the cutoff for selecting overexpressed nodes from this experimental data
down reg cutoff	another single float value that is to be treated as the cutoff for selecting under-expressed nodes from this experimental data
up reg filename	The single column file that the command generates with only the up-regulated node-ids
down reg filename	The single column file that the command generates with only the down-regulated node-ids

Input

```
> cudf
> Enter fold change file: <fold-change-file>
> Enter up-reg threshold (float): <up-reg-cutoff>
> Enter down-reg threshold (float): <down-reg-cutoff>
> Enter file to store up-reg entries: <up-reg-filename>
> Enter file to store down-reg entries: <down-reg-filename>
```

Output

```
Up-reg: <num-of-up-reg-entries>
Down-reg: <num-of-down-reg-entries>
```

Example

```
> cudf
> Enter fold change file: test_fold_change_file
> Enter up-reg threshold (float): 1.0
> Enter down-reg threshold (float): -1.0
> Enter file to store up-reg entries: up_reg
> Enter file to store down-reg entries: down_reg
Up-reg: 5
Down-reg: 2
```

B.1.8 Generate constraints and obtain PO curve

`pathz3`

Description

Generates the constraints and solves them. Uses the process discussed in Section 5.5.2 to generate the points on the PO curve. When run in mode 0, it finds the whole PO curve from scratch. When run in mode 1, it looks for satisfiability at each of the points provided in a file. If some point returns sat it indicates that the curves have not separated completely, otherwise if all points are unsat, then the second curve has moved away from the first curve entirely. The command automatically generates and displays the PO curve. It also stores the points in a file for future use.

Arguments

graph id	id of graph on which the constraint solving is to be performed. The graph is expected to be already pruned between source and target nodes
mode	mode of operation of the command. 0 is for computing the PO curve from scratch. 1 is for checking against existing PO points
connect pairs file	file containing a pair of nodes per line that must be connected
edge relax bound lower	x-coordinate of left lower corner of user defined window
node relax bound lower	y-coordinate of left lower corner of user defined window
edge relax bound upper	x-coordinate of right upper corner of user defined window
node relax bound upper	y-coordinate of right upper corner of user defined window
reach bound	maximum length to be considered for paths while obtaining solutions

filename prefix	short string to be used as prefix for all files generated in the process
up reg filename	file with the hsa-ids of nodes that are up-regulated in the microarray data
down reg filename	file with the hsa-ids of nodes that are down-regulated in the microarray data
essential nodes file	file with hsa-ids of the nodes that have to be present in the solution
avoid nodes file	file with hsa-ids of nodes to be excluded from every solution
essential edges file	file with edges specified in a certain format that must be present in a solution. Format is as follows: source_hsa_id <tab> target_hsa_id <tab> edge_types_separated_by_underscores eg, hsa100 <tab> hsa200 <tab> activation_phosphorylation
avoid edges file	file with edges specified in the same format that are to be excluded from solutions
active nodes file	file with hsa-ids of nodes that have to be activated in a solution
inactive nodes file	file with hsa-ids of nodes that have to be inactivated in a solution
confirmed up reg filename	file with hsa-ids of nodes that are confirmed as up-reg, hence its status cannot be changed
confirmed down reg filename	file with hsa-ids of nodes that are confirmed as down-reg, hence its status cannot be changed
relaxed nodes file	file with hsa-ids of nodes that are to be relaxed in a solution
nonrelaxed nodes file	file with hsa-ids of nodes that are not to be relaxed in a solution
relaxed edges file	file with edges specified in the previously described format that are to be relaxed in a solution
nonrelaxed edges file	file with edges specified in the previously described format that are not to be relaxed in a solution

log fold change file	tab separated two column file where each row contains the hsa-id of one node and its log-fold change value
count solns until	at each PO point, count until number of solutions exceeds given number
print solns until	print solns and relaxations until number of solutions exceeds given number
inc solver timeout	timeout (in millisecs) of the incremental solver
overall solver timeout	timeout (in millisecs) of the overall solving process
PO points file	file with already found PO points for use in mode 1

Input

```
> pathz3 <graph-id>
> Operation mode: <mode>
> Enter connect pairs file: <connect-pairs-file>
> Node relax bounds: <edge-relax-bound-lower> <node-relax-bound-lower>
  <edge-relax-bound-upper> <node-relax-bound-upper>
> Enter path bound: <reach-bound>
> Enter preferred file prefix: <filename-prefix>
> Enter up regulated filename: <up-reg-filename>
> Enter down regulated filename: <down-reg-filename>
> Enter essential nodes filename: <essential-nodes-file>
> Enter avoid nodes filename: <avoid-nodes-file>
> Enter essential edges filename: <essential_edges_file>
> Enter avoid edges filename: <avoid-edges-file>
> Enter active nodes filename: <active-nodes-file>
> Enter inactive nodes filename: <inactive-nodes-file>
> Enter confirmed up-reg filename: <confirmed_up_reg_filename>
> Enter confirmed down-reg filename: <confirmed_down_reg_filename>
> Enter relaxed nodes filename: <relaxed-nodes-file>
> Enter non-relaxed nodes filename: <nonrelaxed-nodes-file>
> Enter relaxed edges filename: <relaxed-edges-file>
> Enter non-relaxed edges filename: <nonrelaxed-edges-file>
> Enter fold change file: <log-fold-change-file>
> Enter solns to count: <count-solns-until>
```

```
> Enter solns to print: <print-solns-until>
> Enter incremental solver timeout (millisecs): <inc-solver-timeout>
> Enter overall solver timeout (millisecs): <overall-solver-timeout>
```

Only for mode 1

```
> Enter file with PO points: <PO-points-file>
```

Output

```
<num-of-connect-pairs> connect pairs

Up: <num-of-up-reg-nodes> Down: <num-of-down-reg-nodes>
Beginning to generate constraints
Source: <source-node> Target: <target-node>
Finished generating constraints

<num-of-PO-points> PO points found
<PO-point-1> is not dominated
<PO-point-2> is not dominated
.
.
.
<PO-point-n> is not dominated
```

Example: mode 0

```
> pathz3 1
> Operation mode: 0
> Enter connect pairs file: connect_pairs_file
> Node relax bounds: 0 0 7 7
> Enter path bound: 6
> Enter preferred file prefix: baseline
> Enter up regulated filename: up_reg
> Enter down regulated filename: down_reg
> Enter essential nodes filename: essential_nodes
> Enter avoid nodes filename: avoid_nodes
> Enter essential edges filename: essential_edges
> Enter avoid edges filename: avoid_edges
> Enter active nodes filename: active_nodes
> Enter inactive nodes filename: inactive_nodes
> Enter confirmed up-reg filename: confirmed_up_reg
```

```
> Enter confirmed down-reg filename: confirmed_down_reg
> Enter relaxed nodes filename: relaxed_nodes
> Enter non-relaxed nodes filename: nonrelaxed_nodes
> Enter relaxed edges filename: relaxed_edges
> Enter non-relaxed edges filename: nonrelaxed_edges
> Enter fold change file: fold_change_as_log_file
> Enter solns to count: 1
> Enter solns to print: 1
> Enter incremental solver timeout (millisecs): 30000
> Enter overall solver timeout (millisecs): 120000

1 connect pairs

Up: 37 Down: 19
Beginning to generate constraints
Source: TNF(hsa7124) Target: NFKBIA(hsa4792)
Finished generating constraints

5 PO points found
(1,7) is not dominated
(2,6) is not dominated
(4,5) is not dominated
(5,4) is not dominated
(6,3) is not dominated
```

Example: mode 1

```
> pathz3 1
> Operation mode: 1
> Enter connect pairs file: connect_pairs_file
> Node relax bounds: 0 0 7 7
> Enter path bound: 6
> Enter preferred file prefix: baseline
> Enter up regulated filename: up_reg
> Enter down regulated filename: down_reg
> Enter essential nodes filename: essential_nodes
> Enter avoid nodes filename: avoid_nodes
> Enter essential edges filename: essential_edges
> Enter avoid edges filename: avoid_edges
> Enter active nodes filename: active_nodes
> Enter inactive nodes filename: inactive_nodes
```

```
> Enter confirmed up-reg filename: confirmed_up_reg
> Enter confirmed down-reg filename: confirmed_down_reg
> Enter relaxed nodes filename: relaxed_nodes
> Enter non-relaxed nodes filename: nonrelaxed_nodes
> Enter relaxed edges filename: relaxed_edges
> Enter non-relaxed edges filename: nonrelaxed_edges
> Enter fold change file: fold_change_as_log_file
> Enter solns to count: 1
> Enter solns to print: 1
> Enter incremental solver timeout (millisecs): 30000
> Enter overall solver timeout (millisecs): 120000
> Enter file with PO points: baseline_PO.dat
1 connect pairs

Up: 37 Down: 19
Beginning to generate constraints
Source: TNF(hsa7124) Target: NFKBIA(hsa4792)
Finished generating constraints

PO point found: (1,7)

Sat point obtained on previous curve
```

The results showed throughout this thesis has been derived from various versions of this tool developed over the past few years.

Bibliography

[1] M. Kanehisa, Y. Sato, M. Kawashima, M. Furumichi, and M. Tanabe, "KEGG as a reference resource for gene and protein annotation," *Nucleic Acids Res.*, vol. 44, pp. D457–D462, 2016. 3, 15, 29

[2] S.-J. Dunn, G. Martello, B. Yordanov, S. Emmott, and A. G. Smith, "Defining an essential transcription factor program for naïve pluripotency," *Science*, vol. 344, no. 6188, pp. 1156–1160, 2014. 5, 23

[3] H. D. Jong, "Modeling and simulation of genetic regulatory systems: A literature review," *Journal of Computational Biology*, vol. 9, pp. 67–103, 2002. 8

[4] R.-S. Wang, A. Saadatpour, and R. Albert, "Boolean modeling in systems biology: an overview of methodology and applications," *Physical Biology*, vol. 9, no. 5, p. 055001, 2012. 8

[5] C. Soul, "Mathematical approaches to differentiation and gene regulation," *Comptes Rendus Biologies*, vol. 329, no. 1, pp. 13 – 20, 2006. Modlisation de systmes complexes en agronomie et environnement. 8

[6] A. Siegel, O. Radulescu, M. L. Borgne, P. Veber, J. Ouy, and S. Lagarrigue, "Qualitative analysis of the relation between {DNA} microarray data and behavioral models of regulation networks," *Biosystems*, vol. 84, no. 2, pp. 153 – 174, 2006. Dynamical Modeling of Biological Regulatory Networks. 8

[7] C. Guziolowski, M. L. Borgne, and O. Radulescu, "Checking consistency between expression data and large scale regulatory networks: A case study," 2007. 9

[8] J. Guerra and I. Lynce, "Reasoning over biological networks using maximum satisfiability," in *Principles and Practice of Constraint Programming* (M. Milano, ed.), (Berlin, Heidelberg), pp. 941–956, Springer Berlin Heidelberg, 2012. 9, 11

[9] F. Corblin, L. Bordeaux, Y. Hamadi, E. Fanchon, and L. Trilling, "A sat-based approach to decipher gene regulatory networks," in *Integrative Post-Genomics, RIAMS'07*, (Lyon), 2007. 9

[10] S. Steel and R. Alami, *Recent Advances in AI Planning: 4th European Conference on Planning, ECP'97, Toulouse, France, September 24 - 26, 1997, Proceedings*. Lecture Notes in Computer Science / Lecture Notes in Artificial Intelligence, Springer, 1997. 11

[11] V. Lifschitz, "What is answer set programming?," in *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence, AAAI 2008, Chicago, Illinois, USA, July 13-17, 2008*, pp. 1594–1597, 2008. 11

[12] M. Gebser, T. Schaub, S. Thiele, and P. Veber, "Detecting inconsistencies in large biological networks with answer set programming," *CoRR*, vol. abs/1007.0134, 2010. 11

[13] M. Gebser, T. Schaub, S. Thiele, and P. Veber, "Detecting inconsistencies in large biological networks with answer set programming," *TPLP*, vol. 11, no. 2-3, pp. 323–360, 2011. 11

[14] A. Subramanian, P. Tamayo, V. K. Mootha, S. Mukherjee, B. L. Ebert, M. A. Gillette, A. Paulovich, S. L. Pomeroy, T. R. Golub, E. S. Lander, and J. P. Mesirov, "Gene set enrichment analysis: A knowledge-based approach for interpreting genome-wide expression profiles," *Proceedings of the National Academy of Sciences*, vol. 102, no. 43, pp. 15545–15550, 2005. 11

[15] L. Wang, B. Zhang, R. D. Wolfinger, and X. Chen, "An integrated approach for the analysis of biological pathways using mixed models," *PLOS Genetics*, vol. 4, pp. 1–9, 07 2008. 11

[16] R. Braun, L. Cope, and G. Parmigiani, "Identifying differential correlation in gene/pathway combinations," *BMC Bioinformatics*, vol. 9, p. 488, Nov 2008. 11

[17] F. Tai and W. Pan, "Incorporating prior knowledge of gene functional groups into regularized discriminant analysis of microarray data," *Bioinformatics*, vol. 23, no. 23, pp. 3170–3177, 2007. 11

[18] L. Beltrame, L. Rizzetto, R. Paola, P. Rocca-Serra, L. Gambineri, C. Battaglia, and D. Cavalieri, "Using pathway signatures as means of identifying similarities among microarray experiments," *PLOS ONE*, vol. 4, pp. 1–11, 01 2009. 11

[19] M. Clment-Ziza, C. Malabat, C. Weber, I. Moszer, T. Aittokallio, C. Letondal, and S. Rousseau, "Genoscape: a cytoscape plug-in to automate the retrieval and integration of gene expression data and molecular networks," *Bioinformatics*, vol. 25, no. 19, pp. 2617–2618, 2009. 11

[20] M. Smoot, K. Ono, T. Ideker, and S. Maere, "Pingo: a cytoscape plugin to find candidate genes in biological networks," *Bioinformatics*, vol. 27, no. 7, pp. 1030–1031, 2011. 11

[21] M. S. Cline, M. Smoot, E. Cerami, A. Kuchinsky, N. Landys, C. Workman, R. Christmas, I. Avila-Campilo, M. Creech, B. Gross, K. Hanspers, R. Isserlin, R. Kelley, S. Killcoyne, S. Lotia, S. Maere, J. Morris, K. Ono, V. Pavlovic, A. R. Pico, A. Vailaya, P.-L. Wang, A. Adler, B. R. Conklin, L. Hood, M. Kuiper, C. Sander, I. Schmulevich, B. Schwikowski, G. J. Warner, T. Ideker, and G. D. Bader, "Integration of biological networks and gene expression data using cytoscape.," *Nat Protoc*, vol. 2, pp. 2366–82, 2007 2007. 11

[22] B. Zhang, Y. Tian, L. Jin, H. Li, I.-M. Shih, S. Madhavan, R. Clarke, E. P. Hoffman, J. Xuan, L. Hilakivi-Clarke, and Y. Wang, "Ddn: a cabig analytical tool for differential network analysis," *Bioinformatics*, vol. 27, no. 7, pp. 1036–1038, 2011. 11

[23] E. Glaab, A. Baudot, N. Krasnogor, and A. Valencia, "Topogsa: network topological gene set analysis," *Bioinformatics*, vol. 26, no. 9, pp. 1271–1272, 2010. 11

[24] E. Glaab, A. Baudot, N. Krasnogor, R. Schneider, and A. Valencia, "Enrichnet: network-based gene set enrichment analysis," *Bioinformatics*, vol. 28, no. 18, pp. i451–i457, 2012. 11

[25] S. Drăghici, P. Khatri, R. P. Martins, G. Ostermeier, and S. A. Krawetz, "Global functional profiling of gene expression," *Genomics*, vol. 81, no. 2, pp. 98 – 104, 2003. 11

[26] D. R. Rhodes, S. Kalyana-Sundaram, V. Mahavisno, R. Varambally, J. Yu, B. B. Briggs, T. R. Barrette, M. J. Anstet, C. Kincead-Beal, P. Kulkarni, S. Varambally, D. Ghosh, and A. M. Chinnaiyan, "Oncomine 3.0: Genes, pathways, and networks in a collection of 18,000 cancer gene expression profiles," *Neoplasia*, vol. 9, no. 2, pp. 166 – 180, 2007. 11

[27] D. Cavalieri, C. Castagnini, S. Toti, K. Maciag, T. Kelder, L. Gambineri, S. Angioli, and P. Dolara, "Eu.gene analyzer a tool for integrating gene

expression data with pathway databases," *Bioinformatics*, vol. 23, no. 19, pp. 2631–2632, 2007. 11

[28] S. Ma and M. R. Kosorok, "Detection of gene pathways with predictive power for breast cancer prognosis," *BMC Bioinformatics*, vol. 11, p. 1, Jan 2010. 11

[29] B. P. Kelley, B. Yuan, F. Lewitter, R. Sharan, B. R. Stockwell, and T. Ideker, "Pathblast: a tool for alignment of protein interaction networks," *Nucleic Acids Res*, vol. 32, pp. 83–88, 2004. 11

[30] D. Warde-Farley, S. L. Donaldson, O. Comes, K. Zuberi, R. Badrawi, P. Chao, M. Franz, C. Grouios, F. Kazi, C. T. Lopes, A. Maitland, S. Mostafavi, J. Montojo, Q. Shao, G. Wright, G. D. Bader, and Q. Morris, "The genemania prediction server: biological network integration for gene prioritization and predicting gene function," *Nucleic Acids Research*, vol. 38, no. suppl_2, pp. W214–W220, 2010. 11

[31] c. Nacu, R. Critchley-Thorne, P. Lee, and S. Holmes, "Gene expression network analysis and applications to immunology," *Bioinformatics*, vol. 23, no. 7, pp. 850–858, 2007. 11

[32] X. Chen, J. Xu, B. Huang, J. Li, X. Wu, L. Ma, X. Jia, X. Bian, F. Tan, L. Liu, S. Chen, and X. Li, "A sub-pathway-based approach for identifying drug response principal network," *Bioinformatics*, vol. 27, no. 5, pp. 649–654, 2011. 11

[33] I. Ulitsky, A. Krishnamurthy, R. M. Karp, and R. Shamir, "Degas: De novo discovery of dysregulated pathways in human diseases," *PLOS ONE*, vol. 5, pp. 1–14, 10 2010. 11

[34] N. Alcaraz, H. Kck, J. Weile, A. Wipat, and J. Baumbach, "Keypathwayminer: Detecting case-specific biological pathways using expression data," *Internet Mathematics*, vol. 7, no. 4, pp. 299–313, 2011. 11

[35] N. Alcaraz, J. Pauling, R. Batra, E. Barbosa, A. Junge, A. G. Christensen, V. Azevedo, H. J. Ditzel, and J. Baumbach, "Keypathwayminer 4.0: condition-specific pathway analysis by combining multiple omics studies and networks with cytoscape," *BMC Systems Biology*, vol. 8, p. 99, Aug 2014. 11

[36] T. Ideker, O. Ozier, B. Schwikowski, and A. F. Siegel, "Discovering regulatory and signalling circuits in molecular interaction networks," *Bioinformatics*, vol. 18, no. suppl_1, pp. S233–S240, 2002. 11

[37] F. Farfán, J. Ma, M. A. Sartor, G. Michailidis, and H. V. Jagadish, "Think back: Knowledge-based interpretation of high throughput data," *BMC bioinformatics*, vol. 13 Suppl 2, p. S4, March 2012. 11

[38] G. Wu and L. Stein, "A network module-based method for identifying cancer prognostic signatures," *Genome Biology*, vol. 13, p. R112, Dec 2012. 11

[39] P. Martini, G. Sales, M. S. Massa, M. Chiogna, and C. Romualdi, "Along signal paths: an empirical gene set approach exploiting pathway topology," *Nucleic Acids Research*, vol. 41, no. 1, p. e19, 2013. 11

[40] C. Li, X. Li, Y. Miao, Q. Wang, W. Jiang, C. Xu, J. Li, J. Han, F. Zhang, B. Gong, and L. Xu, "Subpathwayminer: a software package for flexible identification of pathways," *Nucleic Acids Research*, vol. 37, no. 19, p. e131, 2009. 11

[41] J. Xia and D. S. Wishart, "Metpa: a web-based metabolomics tool for pathway analysis and visualization," *Bioinformatics*, vol. 26, no. 18, pp. 2342–2344, 2010. 11

[42] A. L. Tarca, S. Draghici, P. Khatri, S. S. Hassan, P. Mittal, J.-s. Kim, C. J. Kim, J. P. Kusanovic, and R. Romero, "A novel signaling pathway impact analysis," *Bioinformatics*, vol. 25, no. 1, pp. 75–82, 2009. 11

[43] T. Judeh, C. Johnson, A. Kumar, and D. Zhu, "Teak: Topology enrichment analysis framework for detecting activated biological subpathways," *Nucleic Acids Research*, vol. 41, no. 3, pp. 1425–1437, 2013. 11

[44] C. J. Vaske, S. C. Benz, J. Z. Sanborn, D. Earl, C. Szeto, J. Zhu, D. Haussler, and J. M. Stuart, "Inference of patient-specific pathway activities from multi-dimensional cancer genomics data using paradigm," *Bioinformatics*, vol. 26, no. 12, pp. i237–i245, 2010. 11

[45] S. Nam, H. Chang, K. Kim, M. Kook, D. Hong, C. Kwon, H. Jung, H. Park, G. Powis, H. Liang, T. Park, and Y. Kim, "Pathome: An algorithm for accurately detecting differentially expressed subpathways," *Oncogene*, vol. 33, pp. 4941–4951, 3 2014. 11

[46] L. Geistlinger, G. Csaba, R. Kffner, N. Mulder, and R. Zimmer, "From sets to graphs: towards a realistic enrichment analysis of transcriptomic systems," *Bioinformatics*, vol. 27, no. 13, pp. i366–i373, 2011. 11

[47] L. Koumakis, A. Kanterakis, E. Kartsaki, M. Chatzimina, M. Zervakis, M. Tsiknakis, D. Vassou, D. Kafetzopoulos, K. Marias, V. Moustakis, and

G. Potamias, "Minepath: Mining for phenotype differential sub-paths in molecular pathways," *PLOS Computational Biology*, vol. 12, pp. 1–40, 11 2016. 11

[48] S. Lee, Y. Park, and S. Kim, "Midas: Mining differentially activated subpaths of kegg pathways from multi-class rna-seq data," *Methods*, vol. 124, no. Supplement C, pp. 13 – 24, 2017. Integrative Analysis of Omics Data. 11

[49] H. Lee and M. Shin, "Mining pathway associations for disease-related pathway activity analysis based on gene expression and methylation data," *BioData Mining*, vol. 10, p. 3, Feb 2017. 11

[50] A. Butte and I. S Kohane, "Unsupervised knowledge discovery in medical databases using relevance networks," vol. 1999, pp. 711–5, 02 1999. 12

[51] A. A. Margolin, I. Nemenman, K. Basso, C. Wiggins, G. Stolovitzky, R. D. Favera, and A. Califano, "Aracne: An algorithm for the reconstruction of gene regulatory networks in a mammalian cellular context.," *BMC Bioinformatics*, vol. 7, no. S-1, 2006. 13

[52] A. A. Margolin, K. Wang, W. K. Lim, M. Kustagi, I. Nemenman, and A. Califano, "Reverse engineering cellular networks," *Nat Protoc*, vol. 1, no. 2, pp. 662–671, 2006. 13

[53] J. Zola, M. Aluru, A. Sarje, and S. Aluru, "Parallel information-theory-based construction of genome-wide gene regulatory networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 21, no. 12, pp. 1721–1733, 2010. 13

[54] M. Aluru, J. Zola, D. Nettleton, and S. Aluru, "Reverse engineering and analysis of large genome-scale gene networks," *Nucleic Acids Research*, vol. 41, no. 1, p. e24, 2013. 13

[55] D. Marbach, J. Costello, R. Küffner, N. Vega, R. Prill, D. Camacho, K. Allison, M. Kellis, J. Collins, A. Aderhold, G. Stolovitzky, R. Bonneau, Y. Chen, F. Cordero, M. Crane, F. Dondelinger, M. Drton, R. Esposito, R. Foygel, A. De La Fuente, J. Gertheiss, P. Geurts, A. Greenfield, M. Grzegorczyk, A. Haury, B. Holmes, T. Hothorn, D. Husmeier, V. Huynh-Thu, A. Irrthum, G. Karlebach, S. Lèbre, V. De Leo, A. Madar, S. Mani, F. Mordelet, H. Ostrer, Z. Ouyang, R. Pandya, T. Petri, A. Pinna, C. Poultney, S. Rezny, H. Ruskin, Y. Saeys, R. Shamir, A. Sîrbu, M. Song, N. Soranzo, A. Statnikov, N. Vega, P. Vera-Licona, J. Vert, A. Visconti, H. Wang, L. Wehenkel, L. Windhager, Y. Zhang, and R. Zimmer, "Wisdom of crowds for robust gene network inference," *Nature Methods*, vol. 9, pp. 796–804, 8 2012. 13

[56] "KEGG pathway database webpage." http://www.genome.jp/kegg/pathway.html. 15

[57] "BioCarta database webpage." http://www.biocarta.com/genes/index.asp. 15

[58] "Reactome database webpage." http://www.reactome.org/. 15

[59] M. Kanehisa and S. Goto, "KEGG: Kyoto Encyclopedia of Genes and Genomes," *Nucleic Acids Res.*, vol. 28, pp. 27–30, 2000. 15

[60] R. Diestel, *Graph Theory (Graduate Texts in Mathematics)*. Springer, August 2005. 21

[61] A. Bavelas, "Communication Patterns in TaskOriented Groups," *The Journal of the Acoustical Society of America*, vol. 22, pp. 725–730, Nov. 1950. 22

[62] G. Sabidussi, "The centrality index of a graph," *Psychometrika*, vol. 31, pp. 581–603, December 1966. 22

[63] L. C. Freeman, "A set of measures of centrality based on betweenness," *Sociometry*, vol. 40, pp. 35–41, Mar. 1977. 22

[64] H. Yu, M. P. Kim, E. Sprecher, V. Trifonov, and M. Gerstein, "The importance of bottlenecks in protein networks: Correlation with gene essentiality and expression dynamics," *PLoS Computational Biology*, vol. 3, no. 4, 2007. 22, 23, 68

[65] L. Page, S. Brin, R. Motwani, and T. Winograd, "The pagerank citation ranking: Bringing order to the web," 1998. 23

[66] E. Ramadan, S. Alinsaif, and M. R. Hassan, "Network topology measures for identifying disease-gene association in breast cancer," *BMC Bioinformatics*, vol. 17, p. 274, Jul 2016. 23

[67] M. Giacobbe, C. C. Guet, A. Gupta, T. A. Henzinger, T. Paixão, and T. Petrov, "Model checking gene regulatory networks," in *Tools and Algorithms for the Construction and Analysis of Systems* (C. Baier and C. Tinelli, eds.), (Berlin, Heidelberg), pp. 469–483, Springer Berlin Heidelberg, 2015. 23

[68] R. Sharan and R. M. Karp, "Reconstructing boolean models of signaling," in *Research in Computational Molecular Biology* (B. Chor, ed.), (Berlin, Heidelberg), pp. 261–271, Springer Berlin Heidelberg, 2012. 23

[69] P. Yu, "A class of solutions for group decision problems," vol. 19, 04 1973. 25, 27

[70] P. L. Yu and G. Leitmann, "Compromise solutions, domination structures, and salukvadze's solution," *Journal of Optimization Theory and Applications*, vol. 13, pp. 362–378, Mar 1974. 25

[71] M. Salukvadze, "On the existence of solutions in problems of optimization under vector-valued criteria," *Journal of Optimization Theory and Applications*, vol. 13, pp. 203–217, Feb 1974. 25

[72] C.-L. Hwang and A. Syed Md. Masud, *Multiple Objective Decision Making-Methods and Applications: A State-of-the-Art Survey*, vol. 164. 01 1979. 25, 26, 27

[73] M. Zeleny, *Multiple Criteria Decision Making*. McGraw-Hill Series in Quantitative Methods for Management, McGraw-Hill, 1982. 25, 26, 27

[74] V. Chankong and Y. Haimes, *Multiobjective decision making: theory and methodology*. North-Holland series in system science and engineering, North Holland, 1983. 25

[75] A. Osyczka, "Multicriterion optimization in engineering with fortran programs," 01 1984. 25

[76] J. Kaski and R. Silvennoinen, "Norm methods an partial weighting in multicriterion optimization of structures," *International Journal for Numerical Methods in Engineering*, vol. 24, pp. 1101–1121, 1987. Contribution: organisation=tme,FACT1=0.5¡br/¿Contribution: organisation=mat,FACT2=0.5. 25, 26

[77] T. W. ATHAN and P. Y. PAPALAMBROS, "A note on weighted criteria methods for compromise solutions in multi-objective optimization," *Engineering Optimization*, vol. 27, no. 2, pp. 155–176, 1996. 25, 26

[78] A. Messac, "Physical programming - effective optimization for computational design," *AIAA Journal*, vol. 34, pp. 149–158, Feb 1996. 25, 26

[79] A. Messac, C. Puemi-Sukam, and E. Melachrinoudis, "Aggregate objective functions and pareto frontiers: Required relationships and practical implications," *Optimization and Engineering*, vol. 1, pp. 171–188, Jul 2000. 25, 26

[80] R. T. Eckenrode, "Weighting multiple criteria," vol. 12, pp. 180–192, 11 1965. 25

[81] C.-L. Hwang and Y. Kwangsun, *Multiple Attribute Decision Making: Methods and Applications A State-of-the-Art Survey*, vol. 186. Springer-Verlag Berlin Heidelberg, 1981. 25

[82] H. Voogd, *Multicriteria Evaluation for Urban and Regional Planning*. Pion, 1983. 25

[83] R. Steuer, *Multiple Criteria Optimization: Theory, Computation, and Application*. Wiley series in probability and mathematical statistics, Krieger, 1989. 25

[84] K. Yoon, C. Hwang, and i. Sage Publications, *Multiple Attribute Decision Making: An Introduction*. No. nos. 102-104 in Multiple Attribute Decision Making: An Introduction, SAGE Publications, 1995. 25

[85] I. Das and J. E. Dennis, "A closer look at drawbacks of minimizing weighted sums of objectives for pareto set generation in multicriteria optimization problems," *Structural optimization*, vol. 14, pp. 63–69, Aug 1997. 25

[86] W. Stadler, *Fundamentals of Multicriteria Optimization*, pp. 1–25. Boston, MA: Springer US, 1988. 25, 26, 27

[87] F. Waltz, "An engineering approach: Hierarchical optimization criteria," *IEEE Transactions on Automatic Control*, vol. 12, pp. 179–180, April 1967. 25

[88] M. J. Rentmeesters, W. K. Tsai, and K.-J. Lin, "A theory of lexicographic multi-criteria optimization," in *Proceedings of ICECCS '96: 2nd IEEE International Conference on Engineering of Complex Computer Systems (held jointly with 6th CSESAW and 4th IEEE RTAW)*, pp. 76–79, Oct 1996. 25

[89] K. Miettinen, *Nonlinear Multiobjective Optimization*, vol. 12 of *International Series in Operations Research & Management Science*. Boston, USA: Kluwer Academic Publishers, 1999. 26

[90] R. E. Steuer and E.-U. Choo, "An interactive weighted tchebycheff procedure for multiple objective programming," *Mathematical Programming*, vol. 26, pp. 326–344, Oct 1983. 26

[91] C. Romero, M. Tamiz, and D. F. Jones, "Goal programming, compromise programming and reference point method formulations: linkages and utility interpretations," *Journal of the Operational Research Society*, vol. 49, pp. 986–991, Sep 1998. 26

[92] J. Tind and M. M. Wiecek, "Augmented lagrangian and tchebycheff approaches in multiple objective programming," *Journal of Global Optimization*, vol. 14, pp. 251–266, May 1999. 26

[93] P. Bridgman, *Dimensional Analysis*. Yale University Press, 1922. 26

[94] A. Charnes, W. W. Cooper, and R. O. Ferguson, "Optimal estimation of executive compensation by linear programming," *Management Science*, vol. 1, no. 2, pp. 138–151, 1955. 26

[95] A. Charnes and W. Cooper, *Management models and industrial applications of linear programming [by] A. Charnes [and] W. W. Cooper*. No. v. 1 in Management models and industrial applications of linear programming [by] A. Charnes [and] W. W. Cooper, John Wiley & Sons, 1961. 26

[96] Y. Ijiri, *Management goals and accounting for control*. Studies in mathematical and managerial economics, North Holland Pub. Co., 1965. 26

[97] F. Gembicki, *Performance and Sensitivity Optimization: A Vector Index Approach*. Case Western Reserve University, 1979. 26

[98] W. Ogryczak, "A goal programming model of the reference point method," *Annals of Operations Research*, vol. 51, pp. 33–44, Jan 1994. 26

[99] Y. HAIMES YV, L. LASDON LS, and D. WISMER DA, "On a bicriterion formation of the problems of integrated system identification and system optimization," *IEEE Transactions on Systems, Man and Cybernetics*, vol. SMC-1, pp. 296–297, 1 1971. 26

[100] A. Goicoechea, L. Duckstein, and M. M. Fogel, "Multiobjective programing in watershed management: A study of the charleston watershed," *Water Resources Research*, vol. 12, no. 6, pp. 1085–1092, 1976. 26

[101] J. Cohon, *Multiobjective Programming and Planning*. Mathematics in Science and Engineering, Elsevier Science, 1978. 26

[102] P. Ruíz-Canales and A. Rufián-Lizana, "A characterization of weakly efficient points," *Mathematical Programming*, vol. 68, pp. 205–212, Jan 1995. 26

[103] J. G. Lin, "Proper equality constraints and maximization of index vectors," *Journal of Optimization Theory and Applications*, vol. 20, pp. 215–244, Oct 1976. 26

[104] W. Chen, A. Sahai, A. Messac, and G. Sundararaj, "Exploration of the effectiveness of physical programming in robust design," 2000. 26

[105] A. Messac and C. A. Mattson, "Generating well-distributed sets of pareto points for engineering design using physical programming," *Optimization and Engineering*, vol. 3, pp. 431–450, Dec 2002. 26, 27

[106] A. Messac, C. P. Sukam, and E. Melachrinoudis, "Mathematical and pragmatic perspectives of physical programming," *AIAA Journal*, vol. 39, no. 5, pp. 885–893, 2001. 27

[107] I. Das and J. E. Dennis, "Normal-boundary intersection: A new method for generating the pareto surface in nonlinear multicriteria optimization problems," *SIAM J. Optim*, vol. 8, no. 3, pp. 631–657, 1998. 27

[108] A. Messac, A. Ismail-Yahaya, and C. Mattson, "The normalized normal constraint method for generating the pareto frontier," *Structural and Multidisciplinary Optimization*, vol. 25, pp. 86–98, Jul 2003. 27

[109] C.-L. Hwang, Y.-J. Lai, and T.-Y. Liu, "A new approach for multiple objective decision making," *Computers and Operations Research*, vol. 20, no. 8, pp. 889–899, 1993. 27

[110] K. Deb, *Multi-Objective Optimization using Evolutionary Algorithms*. John Wiley & Sons, 2001. 24, 27, 28

[111] M. W. Krentel, "The complexity of optimization problems," *Journal of Computer and System Sciences*, vol. 36, no. 3, pp. 490 – 509, 1988. 60

[112] S. Toda, "Pp is as hard as the polynomial-time hierarchy," *SIAM J. Comput.*, vol. 20, pp. 865–877, Oct. 1991. 62

[113] L. He, A. M. Friedman, and C. Bailey-Kellogg, "A divide-and-conquer approach to determine the pareto frontier for optimization of protein engineering experiments," *Proteins: Structure, Function, and Bioinformatics*, vol. 80, no. 3, pp. 790–806. 81

[114] J. Adams, "The proteasome: structure, function, and role in the cell," *Cancer treatment reviews*, vol. 29 Suppl 1, pp. 3–9, 06 2003. 92

[115] T. Kaneko, J. Hamazaki, S. ichiro Iemura, K. Sasaki, K. Furuyama, T. Natsume, K. Tanaka, and S. Murata, "Assembly pathway of the mammalian proteasome base subcomplex is mediated by multiple specific chaperones," *Cell*, vol. 137, no. 5, pp. 914 – 925, 2009. 92

[116] R. Sen and D. Baltimore, "Multiple nuclear factors interact with the immunoglobulin enhancer sequences," *Cell*, vol. 46, no. 5, pp. 705 – 716, 1986. 92

[117] A. S. Baldwin, "The nf-kb and ikb proteins: New discoveries and insights," *Annual Review of Immunology*, vol. 14, no. 1, pp. 649–681, 1996. PMID: 8717528. 92

[118] M. Karin, "NF-kappaB as a critical link between inflammation and cancer," *Cold Spring Harb Perspect Biol*, vol. 1, p. a000141, Nov 2009. 92

[119] N. Sangith, K. Srinivasaraghavan, I. Sahu, A. Desai, S. Medipally, A. K. Somavarappu, C. Verma, and P. Venkatraman, "Discovery of novel interacting partners of PSMD9, a proteasomal chaperone: Role of an atypical and versatile PDZ-domain motif interaction and identification of putative functional modules," {*FEBS*} *Open Bio*, vol. 4, no. 0, pp. 571 – 583, 2014. 92