



Universiteit Gent
Faculteit Wetenschappen
Vakgroep Toegepaste Wiskunde, Informatica en
Statistiek

Network-based modelling for omics data

Netwerk-gebaseerde modellering voor omics-data

Mushthofa Mushthofa



Proefschrift tot het bekomen van de graad van
Doctor in de Wetenschappen: Informatica
Academiejaar 2017-2018



Universiteit Gent
Faculteit Wetenschappen
Vakgroep Toegepaste Wiskunde, Informatica en
Statistiek

Promotoren: Prof. Dr. Martine De Cock
Prof. Dr. Kathleen Marchal

Universiteit Gent
Faculteit Wetenschappen

Vakgroep Toegepaste Wiskunde, Informatica en Statistiek
S9, Krijgslaan 281, Gent 9000, België

Tel.: +32-9 264 47 57
Fax.: +32-9 264 49 95
<http://www.twist.ugent.be/>

This work has been supported by the Ghent University Multidisciplinary Research Partnership Bioinformatics: from Nucleotides to Networks (BIG N2N) and the Research Foundation Flanders - FWO: grant nr. G046318N, G0A5315N, and G042813N.



Proefschrift tot het bekomen van de graad van
Doctor in de Wetenschappen: Informatica
Academiejaar 2017-2018

Acknowledgement

Most people will agree that doing research or completing a PhD is not an easy task, and I feel that my own experience is no exception to this. Being able to complete such a challenging task not only requires determination and perseverance, but also a great deal of support from the people around us. Here, I would like give my sincerest thanks to all the people who have made this journey possible for me.

First and foremost, I would like to offer my gratitude to both of my supervisors: prof. Kathleen Marchal and prof. Martine De Cock, not only for their guidance and lessons, both in academic and non-academic matters, but also for being there when I needed them the most, for fueling my spirit and determination (when they are low) and to keep me going through during the difficult times. One cannot feel but very lucky to have such kind and supportive mentors.

A significant portion of my time during the PhD was spent at the lab of prof. Yves Van de Peer, and I am grateful for the support he (and his lab) gave me. Working at his lab together with the other members of his research team has been greatly helpful for accelerating my understanding of biological concepts that I needed for my research. I will certainly miss the discussions and the friendly atmosphere there.

The part of my research dealing with network-based predictive model was heavily inspired by the network-based data integration method by Dr. Lieven Verbeke. Not only did his work make the work done in thesis possible, I had also benefited greatly from the many discussions we had throughout my PhD study. Aside from this, I have enjoyed having him as senior and mentor in the lab as well. Most of the research work I had within the topics of (Fuzzy) Answer Set Programming have been done in collaboration with prof. Steven Schockaert. His deep understanding of the subject and keen observations have helped solve many problems I had faced when tackling this subject.

This PhD has been generously supported by the grants from the Ghent University Multidisciplinary Research Partnership “Bioinformatics: from Nucleotides to Networks (BIG N2N)” and the Research Foundation Flanders (FWO) grant numbers G046318N, G0A5315N, and G042813N. I would like to thank the respective granting bodies and all the people involved that made it possible for me to complete the PhD under these grants.

I’ve been blessed to have had the chance to work and discuss with people from

different labs. At the Department of Applied Mathematics, Computer Science and Statistics, I mainly worked with fellow members of prof. Martine De Cock's lab: Nele, Sofie, Lynn, Elie, Golnoosh and Sarah, all of which have contributed one way or another to help me complete the PhD. At prof. Marchal's lab, I have also had the chance to work with (and learn from) some of her lab members: Maarten, Dries, Bram, Sergio, thank you for all the wonderful discussions.

My life as a PhD student would have been bland and stressful if it were not because of the friendly faces and smiles around me. I would like to thank all the colleagues who have shared some time together in this journey: Johan, Bashir, Hilmar, Beatriz, Oliver. Living as an Indonesian student in Belgium, I feel so grateful for having such a wonderful and supportive family of Indonesian students. We owe a great deal to all of you for all the nice things you did and we had together. A special thanks and mention should go to our "elders": Wa Ann, Pak Bakhtiar & Kak Lely, Koh Juliando and Ci Lanny, for being our guidance and voices of wisdom.

Last but certainly not least, I would like to mention my wife, Fita, and my three beautiful children: Sofia, Amar and Azam, for always being there supporting me, never failing to cheer me up when the going gets tough. I would not have made it without you. You are my spirit and my sunshine. Thank you.

Antwerp, June 2018
Mushthofa

English summary

Biological systems represent some of the most complex objects in nature to study. In order to gain a deeper understanding of a biological system, accurate measurements and observations have to be performed on different aspects of the systems, from genomic, epigenomic, transcriptomic, proteomic, metabolomic, interactomic as well as on the phenotypic level. These measurements have produced, at an increasingly faster pace, a large quantity of data (commonly referred to as the “omics” data). Along with the opportunities that come with the data, scientists are also presented with the great challenge of how to analyze these data so as to gain the best possible understanding of the biological systems they study. Consequently, biology has become increasingly more quantitative, demanding an increasingly more computational way of thinking, as well as concrete methods to make sense of the growing size and complexity of the data. The technological growth for computational power and storage alone is not enough to cope with this, and different methods of data processing and analysis are continuously being developed to address a wide variety of needs, and to answer important questions in biology, evidenced by the fast growing number of publications in computational biology.

As in other scientific fields, quantitative data analysis and modelling have been used to reduce the complexity and noise in the data, and to extract valuable information from it. In molecular biology, the omics data obtained from a system represent different partial points of view of the whole system. Thus, it is often the case that combining different types of omics data in an integrated analysis will bring an advantage in comparison to methods that only look at different omics data independently. In particular, biological interaction networks (interactome) offer much potential to help make sense of the observations present in other omics data, since they represent a high-level view of the “players” (typically genes and/or proteins) and the interactions known between them. This motivates the development of many methods that attempt to use knowledge about underlying biological networks to obtain more insightful conclusions from particular omics data. It is within this spirit that the research presented in thesis was done.

Our research comprises of two main parts. In the first part, we investigate the use of discrete computational modelling methods to gain a high-level understanding of the dynamic behavior of gene regulatory networks and the genes’ activation states based on the knowledge/assumptions about the interactions between the genes. In this scope, we provide methods and tools to support such a modelling, and we perform an assessment on the feasibility of such methods and tools when

applied to real and synthetic data. In the second part, we turn our attention to the problem of predictive modeling on omics data, and in particular the application of machine learning for predicting cancer drug response. In this respect, we propose a method to perform an integrative analysis to select genomic features (“biomarkers”) relevant for the prediction of drug response by employing an underlying prior knowledge interaction network to steer the feature selection process. We perform a validation analysis of the method and highlight its strength and weakness. The following provides a more detailed summary of the content of this thesis.

Chapter 1 provides the introductory backgrounds and context of the thesis, with an emphasis on the introduction of the needed biological concepts, especially catered to people with a computer science background who, perhaps (like me), get fascinated by biology, but also daunted by its complexity.

Chapter 2, 3 and 4 constitute the first part of the thesis. We first describe our proposed methods for the simulation of the dynamics of gene regulatory networks under the Boolean network formalism using a declarative programming paradigm called Answer Set Programming (ASP). In Chapter 3, we describe our proposed method for the design and implementation of a solver for Fuzzy Answer Set Programming (FASP), which extends the expressiveness of ASP by allowing multi-valued propositions to be encoded in the program. This is needed to extend our proposed encoding described in Chapter 2 to allow for multi-level activation states of the gene regulatory model, which is then explained in Chapter 4.

Chapter 5 presents the results of our investigation into the use of integrative analysis using prior knowledge of interaction networks within the context of drug response prediction. Here, we describe our proposed method that was built upon an existing omics data integration method and adapted towards the goal of predictive modelling within the context of cancer drug response. We then report the results of applying the method to publicly-available drug response data sets, and we discuss the interesting findings from the methods from the biological point of view. Finally, Chapter 6 provides some concluding remarks and further outlook on the topics presented in this thesis.

Nederlandse samenvatting

–Summary in Dutch–

Biologische systemen maken deel uit van de meest complexe objecten in de natuur om te bestuderen. Om een beter begrip van een biologisch systeem te krijgen, moeten nauwkeurige metingen en observaties worden uitgevoerd op verschillende aspecten van de systemen, van het genomisch, epigenomisch, transcriptomisch, proteomisch, metabolomisch, interactomisch tot en met het fenotypisch niveau toe. Deze metingen hebben, in een steeds sneller tempo, een grote hoeveelheid gegevens geproduceerd (gewoonlijk aangeduid als de “omics”-gegevens). Naast de mogelijkheden die de gegevens bieden, krijgen wetenschappers ook de grote uitdaging voorgeschoteld hoe ze deze gegevens kunnen analyseren om zo het beste inzicht te krijgen in de biologische systemen die ze bestuderen. Bijgevolg is onderzoek in de biologie steeds meer kwantitatief geworden, gekenmerkt door een groeiende vraag naar meer computationeel denken en methoden om betekenis te geven aan de steeds omvangrijkere en complexere gegevens. De technologische groei in rekenkracht en gegevensopslag alleen is niet voldoende om hiermee om te gaan, en nieuwe methoden voor dataverwerking en -analyse worden continu ontwikkeld om tegemoet te komen aan de behoeften voor verschillende doeleinden, en om allerlei soorten vragen in de biologie te beantwoorden, wat blijkt uit het snel groeiend aantal publicaties in de computationele biologie.

Net als in andere wetenschappelijke gebieden worden kwantitatieve gegevensanalyse en modellering in de biologie gebruikt om de complexiteit en de ruis in de gegevens te verminderen en er waardevolle informatie uit te extraheren. In de moleculaire biologie vertegenwoordigen de omics data die werden verkregen uit een systeem, verschillende gedeeltelijke gezichtspunten van het hele systeem. De combinatie van verschillende soorten omics data in een geïntegreerde analyse biedt dikwijls veel meer mogelijkheden in vergelijking met methoden die alleen naar verschillende omics-gegevens in isolatie kijken. In het bijzonder bieden biologische interactienetwerken (interactome) een groot potentieel om de waarnemingen in andere omics-gegevens beter te kunnen begrijpen, omdat ze een weergave op hoog niveau van de “spelers” vertegenwoordigen (meestal genen en/of eiwitten) en de interacties die tussen hen bekend zijn. Dit motiveert de ontwikkeling van vele methoden die onderliggende biologische netwerken proberen te gebruiken om meer inzichtelijke conclusies te trekken uit bepaalde omics-data. Het is binnen dit kader dat het onderzoek gepresenteerd in dit proefschrift past.

Ons onderzoek bestaat uit twee hoofdonderdelen. In het eerste deel onder-

zoeken we het gebruik van discrete computationele modelleringsmethoden om op hoog niveau inzicht te krijgen in het dynamische gedrag van gen-regulerende netwerken en in de activeringsstaten van genen, op basis van kennis en veronderstellingen over de interacties tussen de genen. In dit kader stellen we nieuwe methoden en hulpmiddelen ter ondersteuning van een dergelijke modellering voor, en evalueren we de haalbaarheid en het nut van deze methoden en hulpmiddelen aan de hand van echte en synthetische gegevens. In het tweede deel richten we onze aandacht op het probleem van voorspellende modellering van omics data, en in het bijzonder de toepassing van machine learning voor het voorspellen van de respons van kankergeneesmiddelen. In dit opzicht stellen we een methode voor om een integrale analyse uit te voeren om genomische kenmerken (“biomarkers”) te selecteren die relevant zijn voor het voorspellen van de medicatierespons, gebruik makend van een onderliggend interactienetwerk met achtergrondkennis dat het selectieproces van de kenmerken kan sturen. We presenteren een validerende analyse van onze methode met aandacht voor de sterke en zwakke punten. Hieronder geven we een meer gedetailleerde samenvatting van de inhoud van dit proefschrift.

Hoofdstuk 1 schetst de achtergrond en context van dit proefschrift, met de nadruk op de introductie van de benodigde biologische concepten, vooral gericht op mensen met een achtergrond in de informatica die, misschien (zoals ik), gefascineerd raken door de biologie, maar ook ontmoedigd door de complexiteit ervan.

Hoofdstuk 2, 3 en 4 vormen het eerste deel van het proefschrift. We beschrijven eerst onze voorgestelde methoden voor de simulatie van het dynamisch gedrag van gen-regulerende netwerken onder het Booleaanse netwerkformalisme met behulp van een declaratief programmeerparadigma genaamd Answer Set Programming (ASP). In Hoofdstuk 3 beschrijven we onze voorgestelde methode voor de implementatie van een “solver” (oplosser) voor Fuzzy Answer Set Programming (FASP), dat de expressiviteit van ASP uitbreidt door toe te staan dat meerwaardige proposities worden gecodeerd in het programma. Dit is nodig om onze voorgestelde codering zoals beschreven in Hoofdstuk 2 uit te breiden om activering op meerdere niveaus mogelijk te maken voor het gen-regulerend model, dat vervolgens wordt uitgelegd in Hoofdstuk 4.

Hoofdstuk 5 presenteert de resultaten van ons onderzoek naar het gebruik van een integrale analyse met behulp van een interactienetwerk met achtergrondkennis binnen de context van de voorspelling van geneesmiddelreacties. We stellen een methode voor die is gebaseerd op een bestaande omics data-integratietechniek en aangepast om de respons van kankergeneesmiddelen te voorspellen. Vervolgens rapporteren we de resultaten van de toepassing van de methode op openbaar beschikbare gegevensreeksen voor geneesmiddelenreacties en bespreken we de interessante bevindingen van de methoden vanuit biologisch oogpunt. Hoofdstuk 6 tenslotte geeft enkele afsluitende opmerkingen en een verdere kijk op de onderwerpen die in dit proefschrift worden gepresenteerd.

Table of Contents

Acknowledgement	i
English summary	iii
Nederlandse samenvatting	v
1 Introduction	1-1
1.1 Preamble	1-1
1.2 A computer scientist's point of view of molecular biology	1-5
1.2.1 DNA, AKA the source code of life	1-5
1.2.2 Central dogma, AKA program execution	1-7
1.2.3 Biological networks, AKA program inter-dependencies	1-8
1.3 Network-based methods for omics data	1-8
1.4 Problem statement	1-9
1.5 Thesis structure	1-10
References	1-12
2 Modelling GRNs with Boolean Networks and ASP	2-1
2.1 Introduction	2-1
2.2 Methods and Models	2-3
2.2.1 Boolean network modelling of GRNs	2-3
2.2.2 Representing Boolean network models of GRNs and computing their attractors using ASP	2-5
2.3 Results	2-12
2.3.1 ASP-G: a novel framework for the simulation and attractor computation of GRNs with Boolean network models	2-12
2.3.2 Simulation Results	2-12
2.4 Conclusion	2-14
References	2-18
3 Modelling and solving problems using FASP	3-1
3.1 Introduction	3-1
3.2 Modelling problems as logic programs	3-2
3.3 Syntax and Semantics of FASP	3-4
3.4 Solving finite satisfiability of FASP using ASP	3-6
3.4.1 FASP rule rewriting	3-7

3.4.2	Translation to classical ASP	3-8
3.5	Evaluating disjunctive rules	3-11
3.5.1	SRCF programs	3-12
3.5.2	Non-SRCF programs	3-15
3.5.3	Incorporating program decomposition	3-16
3.6	Experimental benchmark	3-18
	References	3-20
4	Modelling multi-valued networks using FASP	4-1
4.1	Introduction	4-1
4.2	Related Work	4-3
4.3	Multi-valued Networks	4-4
4.3.1	Modelling multi-valued networks using FASP	4-4
4.3.2	Finding steady states	4-7
4.3.3	Finding fixed-size cyclic attractors	4-10
4.4	Automatic encoding of network descriptions	4-13
4.5	Benchmark and Experiments	4-15
4.5.1	Experiments on real networks	4-15
4.5.2	Experiments on synthetic networks	4-16
4.6	Conclusion	4-21
	References	4-24
5	Network-based predictive modelling for cancer drug response	5-1
5.1	Introduction	5-1
5.2	Results	5-3
5.2.1	Network-based strategy molecular to identify sub-networks driving a clinical phenotype	5-3
5.2.2	Application of the method to drug response prediction	5-5
5.2.3	Validation of the pathway-based features by assessing predictive power	5-7
5.2.4	Biological Relevance of the pathway-based features	5-13
5.3	Discussion	5-17
5.4	Methods and Data	5-19
5.4.1	Cell lines and data processing	5-19
5.4.2	Network model	5-20
5.4.3	Calculating the similarity scores	5-21
5.4.4	Assigning node ranks	5-22
5.4.5	Training and testing the classification model	5-22
5.4.6	Implementation of the method	5-23
	References	5-25
6	Concluding Remarks	6-1
6.1	Future Perspective	6-2
	References	6-4

A List of Publications

A-1

List of Figures

1.1	The increase in capacity for DNA sequencing in both the total number of human genomes sequenced (left axis) as well as the worldwide annual sequencing capacity, taken from [4]	1-2
1.2	The decrease in cost for sequencing the whole human genome, as calculated by the National Human Genome Research Institute (NHGRI).	1-3
1.3	The standard genetic code (codon table). U stands for Uracil, which replaces Thymin in RNA. (Credit: The OpenStax College) .	1-6
1.4	The flow of genetic information (credits: Wikipedia)	1-7
2.1	A Boolean network model with three genes. Edges with arrowed tips are activating interactions and edges with blunt tips are repressing (inhibiting) links.	2-4
2.2	State Transition Graph of the example Boolean network given in Figure 2.1, when we assume synchronous update and the r^+ activation rule. The nodes in this graph are states of the network. Since the network has 3 nodes, there are $2^3 = 8$ possible states, as shown by the 8 nodes in this graph. There are two attractors in this network : $\{(000)\}$ and $\{(010), (101)\}$	2-4
2.3	State Transition Graph of the example Boolean network given in Figure 2.1, when we assume synchronous update and the r^* activation rule.	2-5
2.4	Architecture of ASP-G.	2-6

2.5	An example of a State Transition Graph of the Boolean network given in Figure 2.1, when we assume asynchronous update. Instead of the two attractors as in the synchronous case, here we only have one attractors: $\{(000)\}$. As can be seen, transitions are non-deterministic. For example: there are two possible next states after (001): (000) or (011). To find the attractors in such case, we apply the following STG reduction principle: remove states from the STG which contain more than one outgoing (asynchronous) transition. It is safe to remove such states, since they cannot satisfy the second condition of the definition of a state in an attractor (cf. Definition 1). Therefore, we need only to look attractors in the reduced STG obtained from such removal process. For example, applying this principle to the STG in Figure A, we obtain the reduced STG (Figure B), containing the only attractor $\{(000)\}$. .	2-9
(a)	Figure A	2-9
(b)	Figure B	2-9
2.6	These figures illustrate the process of applying Algorithm 1 on the STG in Figure 2.2. In (a), we have the original STG. We repeatedly find path of arbitrary length, k , where k varies from 2 to 2^n , where n is the number of nodes in the Boolean network. Consider the case when $k = 4$. One path we might obtain is $(110) \rightarrow (111) \rightarrow (010) \rightarrow (101) \rightarrow (010)$ (Figure (b)). By extracting two identical states from this path (010), we obtain the endpoints of a cycle, indicating an attractor, in this case $\{(010), (101)\}$. By disallowing these states (Figure (c)), we find the only other path of length 4, which is $(000) \rightarrow (000) \rightarrow (000) \rightarrow (000) \rightarrow (000)$, from which we extract the second attractor, $\{(000)\}$. Disallowing this state, we find no further path of length 4 (Figure (d)). Continuing in this manner until $k = 2^n$ guarantees that all attractors are identified. The process of generating and checking the path are performed by using ASP rules. The addition of disallowed states is performed by using ASP constraints.	2-11
3.1	Work flow for solving problems using (F)ASP	3-2
3.2	Example graph	3-3
4.1	Diagram for the network of <i>P. aeruginosa</i>	4-6
4.2	State transition graph for the network of <i>P. aeruginosa</i> using the synchronous update	4-6
4.3	FASPG work flow	4-14
4.4	Running time for computing steady states. Missing nodes indicate failure due to time-outs/memory-outs.	4-18
4.5	Memory usage for computing steady states. Missing nodes indicate failure due to time-outs/memory-outs.	4-18

4.6	Running time for computing synchronous cyclic attractors. Missing nodes indicate failure due to time-outs/memory-outs. The singleton node represents $k = 4$. All instances with $k > 4$ failed due to time-outs/memory-out.	4-19
4.7	Memory usage for computing synchronous cyclic attractors. Missing nodes indicate failure due to time-outs/memory-outs. The singleton node represents $k = 4$. All instances with $k > 4$ failed due to time-outs/memory-outs.	4-19
4.8	Running time for computing asynchronous cyclic attractors. Missing nodes indicate failure due to time-outs/memory-outs.	4-20
4.9	Memory usage for computing asynchronous cyclic attractors. Missing nodes indicate failure due to time-outs/memory-outs.	4-21
5.1	Overview of method. 1) A pair of thresholds is chosen to define the resistant and responsive cell lines for a certain drug, 2) molecular data from the set of cell lines on each set are combined and integrated on top of the biological network, 3) kernel scores are computed and averaged for all cell lines within each set, 4) rankings of features are determined by the largest differential scores between the two sets, 5) Train a classifier, and then test using new data with/without diffusion.	5-4
5.2	Genes associated to the top ranking nodes selected for the MEK inhibitor PD-0325901, mapped on the gene-gene interaction network. Blue nodes indicate the genes that were selected as top ranking using the network based data integration based on their status nodes, while yellow nodes indicates the nodes associated to top predictive features. Red nodes were not selected by our method, but are displayed to show the connectivity between the selected nodes.	5-6
5.3	Comparison of the performance of random forest classifiers using the statistical feature selection (blue), random forest using network-based feature selection (green) and random forest using diffused data (yellow) in the case where a clear signal was present in the data and where the network feature selection method performed statistically better than random (part 1).	5-8
5.4	Comparison of the performance of random forest classifiers using the statistical feature selection (blue), random forest using network-based feature selection (green) and random forest using diffused data (yellow) in the case where a clear signal was present in the data and where the network feature selection method performed statistically better than random (part 2).	5-9

- 5.5 Comparison of the performance of random forest classifiers using the statistical feature selection (blue), random forest using network-based feature selection (green) and random forest using diffused data (yellow) in the case where a clear signal was present in the data and where the network feature selection method did not perform statistically better than random. 5-10
- 5.6 Comparison of the performance of random forest classifiers using the statistical feature selection (blue), random forest using network-based feature selection (green) and random forest using diffused data (yellow) in the case where no clear signal was present in the data but where the network feature selection method could still perform statistically better than random. 5-12
- 5.7 List of known drug-feature associations, and the rankings of the features obtained using our network-based feature ranking method 5-14
- 5.8 Top-ranking features selected using our method for OLAPARIB. Rows represent the selected features, whereas column represent cell lines, sorted from the most non-responsive to most responsive. Mutations and amplifications are shown as 1s, while deletions as -1s. 5-16
- 5.9 The sub-network selected by our feature selection method for cyclin-inhibitor drug PD-0332991. Yellow nodes are the nodes which are also with high predictive values according to the random forest classifier, while red nodes are the drug's targets. 5-19
- 5.10 Classification performances of random forest using all features (red), top 30 statistically-selected features (blue), network-selected features (green) and using network-based data transformation/diffusion (yellow), compared to the background distribution of random forest performances using randomly shuffled features on drug AZD8055. 5-24

List of Tables

2.1	ASP-G results and running times for common GRNs found in the literature. Network: describes the original network model. Genes: number of genes present in the network; Attractors: number of detected attractors; Update mechanism: synchronous versus asynchronous updating was used as described in the methods section; Activation rules: r^* activation rules indicates that a gene becomes active when it has more active activating interactors than active inhibiting ones, while the r^+ activation rules indicate that a gene becomes active if it has at least one active activating gene and no inhibiting ones. Time: running time on a Dell Latitude D820 notebook.	2-16
2.2	Running times and the number of synchronous and asynchronous attractors computed by ASP-G for the gene knockout experiments described by Pedicini et al. [25]. Inaccuracies in the original results in [25] are corrected in this work, resulting in minor discrepancies between result represented in this table (e.g., the number of attractors) and the one in [25].	2-17
3.1	Values in the cells indicate average execution times (over ten instances) in seconds for the non-timed-out executions. Cells labeled with '(TO)' indicates that all executions of corresponding instances exceeded time/memory limit	3-19
4.1	Regulatory relationship in the <i>P. aeruginosa</i> mucus development network	4-6
4.2	Benchmark results.	4-23

1

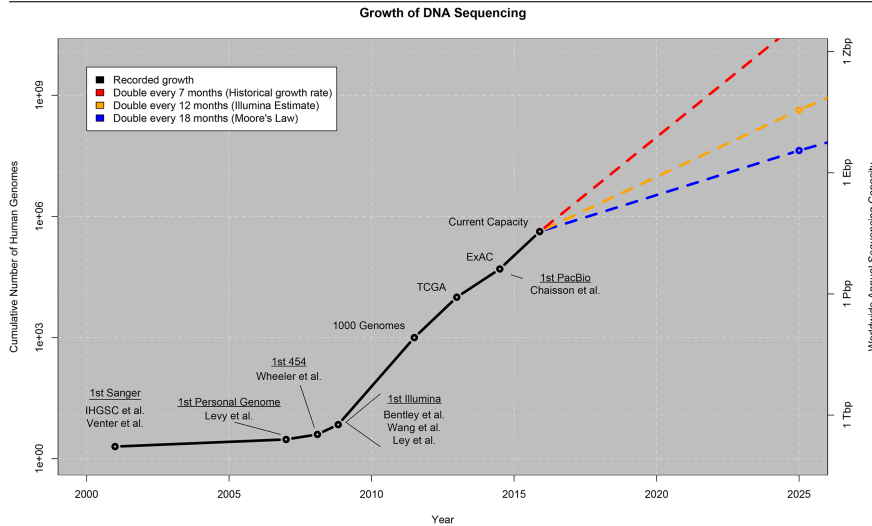
Introduction

1.1 Preamble

In the recent decades, biologists and computer scientists have seen Computational Biology grow from just a niche intersection between computer science and biology, into an exciting, rich and fast changing interdisciplinary field that provides huge promises as well as challenges to scientists in either fields. This development is in part driven by two very important factors: (1) the increasing scalability and precision in the measurement and extraction of molecular biological data, and (2) the increasing computational power (both in storage and processing capability) coupled with the appropriate methods and algorithms to process and extract information out of the data. A visible pattern of interplay between these two developments is that, usually, given a newly-developed method in (molecular) biology that generates a wealth of new types of data, a whole set of computational methods are then developed and evaluated in order to make sense of the newly-generated data and to extract meaningful biological insight from it.

One of the most well-known examples in this respect is the development of DNA sequencing technology. From the work that lead to the first draft of the human genome [1, 2] to recent developments in the high-throughput sequencing technologies (HTS) [3], we have seen an exciting growth of data and information accessible for computational biologists to make use of. Over the last two decades, the advancement in sequencing technologies has increased the capacity and scalability of genome sequencing significantly, (Figure 1.1, [4]) and at the same time,

Figure 1.1 The increase in capacity for DNA sequencing in both the total number of human genomes sequenced (left axis) as well as the worldwide annual sequencing capacity, taken from [4]

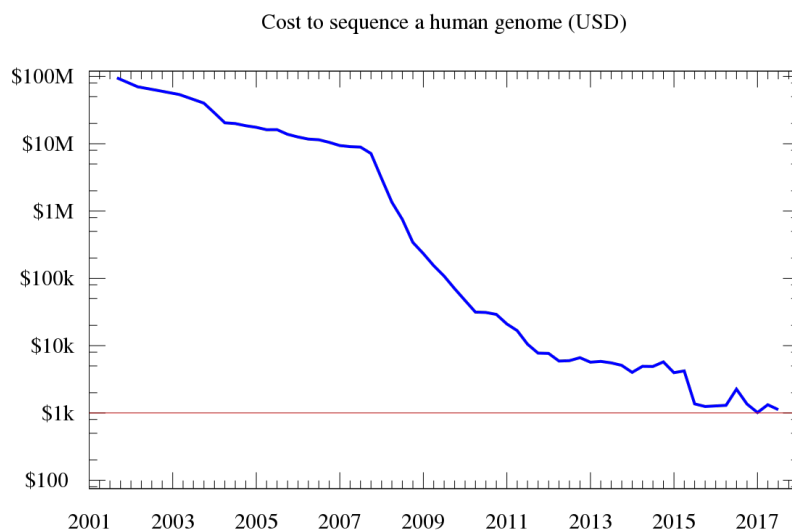


reduced the cost to a level that makes it possible for individuals to afford their own genome sequencing (Figure 1.2, NHGRI).

Parallel to the advances in genomics, high-throughput measurement methods have also been devised for the measurement and analysis of gene expressions and transcripts, leading to the so-called transcriptomics data. From the already-ubiquitous gene expression profiling techniques using microarrays, to the emerging technique of expression profiling using RNASeq, computational biologists have been able to observe, with more granularity and accuracy, the pattern of gene expressions in different conditions and contexts, allowing them to better understand the observed phenotypes. Publicly-available gene expression databases, such as the Gene Expression Omnibus (GEO) [5], have helped preserve and disseminate scientific records containing transcriptomics data to the scientific community. Similarly, techniques such as mass-spectrometry and NMR spectroscopy have been popularly used for measuring protein and metabolite abundances, generating huge amounts of proteomics and metabolomics data available for analysis by the computational biologists and bioinformaticians.

Within the bigger picture, all these different types of data (generally being referred-to as the “omics” data) are ultimately different aspects and different points of view used by scientists to observe and measure the system they are trying to understand. As such, in many cases, it is often useful (or even necessary) to integrate these different data types in order to obtain a more thorough understanding of the

Figure 1.2 The decrease in cost for sequencing the whole human genome, as calculated by the National Human Genome Research Institute (NHGRI).



system [6, 7].

Another important aspect of a biological system is the interrelationships between the different molecular components within the systems (DNA, RNA, genes, proteins, etc). It is a well established fact that these components generally do not function independently, but rather interdependently in a complex manner. These inter-dependencies are captured as biological network data, in the form of, e.g., gene regulatory networks, protein-protein interaction networks, coexpression networks, etc. A more distilled (and curated) form of such biological networks usually takes the form of the so-called biological pathways and signaling networks, where one typically needs to first filter the data, reduce the noise, and select only the interactions with a high confidence value, and then integrate between different types of networks (e.g., GRNs, PPI networks and metabolic networks) in order to have a more complete picture of the interactions within the context of a particular biological function. A large amount of efforts and resources have been dedicated to generating and analyzing these different types of biological networks. Generic biological networks/pathway databases such as KEGG [8] and Reactome [9] have helped preserve and disseminate data on biological networks that can benefit researchers from many different areas, while more specific-purpose biological networks such as the CMAP & LINCS [10, 11] or the Atlas of Cancer Signaling Networks (ACSN) [12] have helped researchers in a specific domain (e.g., cancer

omics analysis) leverage a more detailed knowledge of interactions specifically tailored for their needs.

The availability of these network data presents an opportunity to better analyze and understand the available omics data, e.g., by providing more context and/or driving the analyses and their interpretation. This is particularly important when we have limited and/or noisy omics data to work with. By integrating the relevant underlying interaction data, we hope to be able to incorporate domain knowledge into our analysis, and ultimately to minimize the effects of limited and/or noisy data in the analysis. It is within the spirit of this idea, that the research in this thesis was done. In the first part, we deal with a commonly-used logical based modelling of biological regulatory networks called Boolean networks [13]: given a known regulatory interaction network between genes/components, the model simulates the trajectory of the activation states of the genes and determines the state(s) towards which the system will eventually settle. Such a simulation is useful when one has reliable data on the regulatory interactions between the genes involved in certain biological functions, and wants to understand the dynamic behavior of the system as well as find its potentially-“unobserved” stable states. We extend this formalism to allow for more flexibility in the number of activation levels that each node can have in the states, and we provide tools and benchmark tests to assess the proposed solutions.

In the second part, we turn our attention to the problem of network-based genomic feature selection and phenotype prediction: given available omics data and a relevant underlying prior-knowledge network, we present a method that attempts to select the “best” set of features that are both predictive and “relevant” within the context of the network. The word “relevant” here roughly means that the selected features are closely connected to subsets of the networks that are known *a priori* to be relevant to the biological function that is being studied. In particular, we apply this method to the problem of predicting cancer drug response. Given a set of tumour cells (with their respective genomic/transcriptomic profiles), as well as their response towards a particular drug, the method performs an integrative analysis combining these data with the available signaling pathway/network information in order to prioritize the most relevant features. By using such an integrative method that incorporates a prior knowledge interaction network, we are not only able to derive statistically-predictive features, but also biologically-relevant features that can be used to understand the mode-of-action of the drug.

1.2 A computer scientist's point of view of molecular biology

Coming from a pure computer science background, the doctoral research we have conducted has been a journey of constant learning of fascinating biological concepts. Just like the saying goes, if all you have is a hammer, everything looks like a nail, I could not help but to see all these new concepts and principles from the eyes of a computer scientist. As such, as part of this thesis, this section is written (in a somewhat less formal way) to provide the necessary biological background knowledge necessary to fully understand this thesis, and particularly tailored to resonate with those who have a computer science background. Fully knowing the risks of receiving criticisms (especially from more biology-adept readers) for sometimes making inappropriate analogies and possibly “dumbing down” the biological concepts, this section will try to make a comparison/analogy between a computer system and a biological system. For simplicity, we will assume that the smallest unit of a biological system is the cell, which we will compare to a single unit of a computer (software) system.

1.2.1 DNA, AKA the source code of life

Just like a single machine instruction is the basic unit of a computer program, deoxyribonucleic acid (DNA) is the basic unit of genetic code in life. DNA forms as a linked string of biochemical molecules known as the nucleic acids. Similar to how digital instructions for computers are encoded in binary format, the genetic instructions for biological systems are encoded in a 4-letter code: the different types of nucleobases on each of the DNA positions: Adenin (A), Thymin (T), Guanine (G) and Cytosine (C). Using the combinations of these 4 letters, life (or rather biochemistry) encodes all the information necessary to provide the molecular mechanisms needed during the growth, development, and functioning of a biological system. This “genetic code” has been mapped and generally holds for all known-living systems¹. Just as the machine instruction set for computer systems is used as a reference for the available instruction sets and how they are encoded digitally in a computer architecture, the genetic code mapping (often referred as the codon table) serves as a reference for decoding the DNA.

As can be seen in Figure 1.3, each combination of a 3-letter string codes for either one of the 20 amino acids or the start/stop instruction. These amino acids can combine in different ways (depending on which amino acids are present and in which order) into the macro-molecules we commonly know as proteins. These proteins in turn function as the main molecular actors in a biological action, performing functions such as: catalyzing other biochemical reactions, carrying sig-

¹There are known exceptions to this general code that will not be relevant for this discussion.

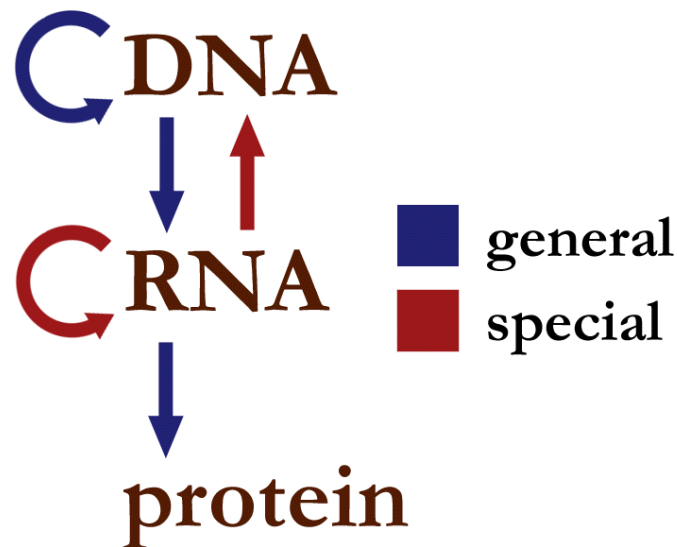
Figure 1.3 The standard genetic code (codon table). U stands for Uracil, which replaces Thymine in RNA. (Credit: The OpenStax College)

		Second letter				
		U	C	A	G	
First letter	U	UUU } Phe UUC } UUA } Leu UUG }	UCU } Ser UCC } UCA } UCG }	UAU } Tyr UAC } UAA Stop UAG Stop	UGU } Cys UGC } UGA Stop UGG Trp	U C A G
	C	CUU } Leu CUC } CUA } CUG }	CCU } Pro CCC } CCA } CCG }	CAU } His CAC } CAA } Gln CAG }	CGU } Arg CGC } CGA } CGG }	U C A G
	A	AUU } Ile AUC } AUA } AUG Met	ACU } Thr ACC } ACA } ACG }	AAU } Asn AAC } AAA } Lys AAG }	AGU } Ser AGC } AGA } Arg AGG }	U C A G
	G	GUU } Val GUC } GUA } GUG }	GCU } Ala GCC } GCA } GCG }	GAU } Asp GAC } GAA } Glu GAG }	GGU } Gly GGC } GGA } GGG }	U C A G

nals from one cell to another, regulating the rate of certain reactions, as well as forming the necessary structural components for the living system (tissues, organs). The start and stop codons function as a kind of *begin* and *end* marker for reading and executing of set of instructions. A *gene* is a stretch of DNA, typically containing several hundreds to thousands of base letters, marked with a start and an end codon. A gene plays a role similar to a single procedure in a computer program, which may be “executed” (or transcribed, in the case of the gene) to perform a specific functionality. There is typically a one-to-one relationship between a gene and the protein it produces. However, just like a single program procedure may be invoked in a different context, a gene maybe transcribed in different conditions and contexts as well.

In computer software development, we are familiar with the term “bug” to refer to a small defect in the underlying code of a software that leads to unintended consequences (reduced functionality, incorrect behavior etc.). In a similar way, DNA code may have a “defect”, occurring as either the substitution, deletion or insertion of a single or multiple bases in a region (usually referred to as “mutations”). These abnormalities typically manifest as diseases that hamper the individual’s biological functions in one or a few ways. For example, β -thalassemia is a genetic disorder affecting the blood caused by a mutation in the gene HBB that causes hemoglobin to not be synthesized properly. Cancer is a set of diseases typically caused by a set of genetic and/or epigenetic aberrations that promote the cells to undergo uncontrolled growth and non-differentiation, eventually disrupting the functions of other cells, tissues and organs.

Figure 1.4 The flow of genetic information (credits: Wikipedia)



1.2.2 Central dogma, AKA program execution

As mentioned previously, the code in the DNA needs to be “executed” in order for it to take effect. This process of execution is described as the central dogma of molecular biology, of which the flow of information can be summarized in Figure 1.4.

The basic flow starts from the DNA, which gets transcribed into ribonucleic acid (RNA) which carries the information from the DNA. Within the context of a gene, transcription means that the gene is “expressed” (i.e., executed) and will therefore, potentially have effects on the system. Some genes may only be expressed in certain contexts, and the extent to which this expression takes place may vary between conditions. The resulting RNA molecules are then “translated” (i.e. decoded) into the appropriate sequence of amino acids according to the genetic code, which will then assemble and fold into the corresponding protein. Another basic flow of information is from DNA to DNA, which is typically “just” a process of copying the DNA into a new identical copy (typically in cell division processes). This copying process, however, can be imperfect. Some stretch of code may not be properly copied, resulting in a non-identical copy, or some parts may even be missing, leading to the genetic aberrations/mutations mentioned earlier. In the reproductive context, these “faulty” copies can lead to slight variations in the offspring, which is actually one of the main driving forces for evolution (the other being selection of the fit). In non-reproductive contexts, the faulty copies may lead

to abnormal cells, such as tumor/cancer cells.

1.2.3 Biological networks, AKA program inter-dependencies

In complex software, we usually need to break down the functionalities into different modules, sub-modules, and eventually individual procedures, in order to cope with the overall complexity. Once built, the complete software package will exhibit a complex structure of interactions and inter-dependencies between different parts in a module, or even between different modules. A similar situation occurs in a biological system, where genes may depend on other genes, interact with and regulate the transcription of other genes, or whose proteins interact with the proteins of other genes. Together with possibly other types of molecules (e.g., metabolites), these genes and proteins play a role in certain biological functions and pathways. From a broader perspective, we can picture a network of interactions between the genes that spans the whole genome.

In the molecular biology literature, we recognize at least four types of biological networks: (1) gene regulatory networks (GRNs), where a network of genes regulate (determine the transcription levels of) each other, (2) gene co-expression networks, where correlations between the expression levels of the genes indicate similarity in role/function, (3) protein-protein interaction networks (PPIs), where proteins may activate/deactivate, or fuse with other proteins to perform their functions, and (4) metabolite networks, representing the network of interactions between all metabolic molecules that are involved in a particular metabolic pathway

There are various methods used to obtain/gather/infer such networks. For example CHIP-chip and CHIP-seq are typically used to determine regulatory interactions between genes by detecting transcription binding sites. Statistical methods that infer such regulatory interaction networks from expression (typically microarray) data are also quite commonly-found in the literature [14]. Various other methods have also been developed to analyze omics data to infer other types of biological networks.

1.3 Network-based methods for omics data

Given available biological networks, and interesting question then is, what can we do with it to further our understanding regarding the biological system being studied. One common theme that can be found in the literature is to typically use the biological network information to provide context for analyzing other omics data and to generate interesting hypothesis from it. In order to do this, one would typically use the network in some kind of formal model that can be integrated with the other omics data. Here we provide a very brief summary of the methods that perform such modelling and integration.

The first type of formal modelling using biological network that is typically used in combination with a regulatory network is the steady-states dynamics of GRN, for example using the Boolean network models [13] or the Piece-wise linear models [15]. A review of available models on this topic is given in [16]. Boolean networks model was first introduced by Kauffman ([17] in the context of computational evolutionary models, but have since be used in many applications, for example, in understanding the dynamics of GRN involved in the development of flowers in *Arabidopsis thaliana* [18, 19], or the GRN involved in the cell cycle process in yeast [20]. These types of modelling generally require tools that can perform discrete simulations of the activation states of the genes in the GRN, for example [21, 22]. It is within this topic that the first part of the research was conducted.

Another type of network-based model typically used in integrating and “driving” the analysis of omics data is where biological networks (typically from different sources) are used as prior knowledge network that can provide additional benefits for the data when used in statistical/predictive analyses. By adding information this prior knowledge, statistical analyses performed on the data might be able to produce better results by reducing the effects of noise and/or the limited number of samples. Examples of such methods are the use of networks to derive important pathways in the context of cancer [23–25]. It is within the context of this topic that the second part of this research was performed.

1.4 Problem statement

In this thesis, we are focusing our attention to two main problems:

- **Simulation of GRN state dynamics using Boolean/discrete modelling:** given a known GRN structure, can we simulate its state dynamics and identify its *attractors* under the (discrete) multi-valued settings? The attractors of a dynamic system refer to a set of states of the system to which the dynamics of the system converges in the longer term. In the context of biology, the attractors of a GRN usually refer to the stable gene activation states that the GRN can attain, and will usually correspond to the different phenotypic states that the system can take, e.g., the different cell types. Being able to simulate the dynamics of a GRN and identify its attractors are then quite important in understanding the behavior of the system. By having a tool that is capable of simulating the dynamics of not only GRN models under Boolean activation levels, but also multi-valued activation levels would serve to increase the scope on which such a modeling approach can be applied.
- **Network-based predictive modelling:** One of the biggest challenges in predictive modelling for biological data is the fact that the data often come

in a less-than-ideal conditions: biological data are often obtained from noisy measurements that can lead to lower data quality. Also, some data are just too rare or too expensive to obtain, and hence they suffer from small sample sizes, often diminishing the statistical power of any predictive modelling analysis. This problem is often exacerbated by the fact that omics data often come with a high dimensionality. Transcriptomic data typically consist of thousands of gene expression level measurements for each sample, while the number of samples is quite often below the thousands. The combination of small sample sizes and high data dimensionality can cause a decrease in the predictive power of any model derived from the data, a phenomenon often called “the curse of dimensionality”. In the context of computational biology, interaction networks may serve as an underlying context to correlate between the different molecular entities involved in a system, and thus can provide information on how the different variables measured from the system are correlated with each other (even with a low sample size), potentially reducing the complexity of the data and allowing us to focus on the more relevant variables. A predictive modelling approach that takes into account these underlying biological interactions may be able to reduce the effects of the noisy data and the small sample sizes, as well as to derive a more biologically relevant set of predictors associated with the phenotype of interest. In the context of this thesis, we want explore to what extent we can use an interaction network (consisting of different types of molecular interactions) between genetic entities to drive integrative genotype-phenotype association of tumor cohorts. A tumor cohort typically consists of a set of patients with distinct phenotypic behavior (e.g. drug responders and non-responders) for which the tumors have been genotyped and molecularly phenotyped (using transcriptomics etc).

1.5 Thesis structure

This content of this thesis is the result of four years of research within the context of network-based modelling for omics data. These results can be broken down into two major parts, described next. The first part deals with the logical modelling of biological networks and the simulation of their behavior, and in particular, the development of the framework for computing the attractors of the simulated biological networks. In Chapter 2, we first start with the development of ASPG [26], which is a framework for simulating the behaviour of gene regulatory networks, as well as computing their trajectory and attractor states, using the formalism of Boolean networks and implemented using Answer Set Programming (ASP). ASP [27] is a popular declarative programming paradigm commonly used for solving combinatorial search and optimization problems, such as attrac-

tor state computation. Our ASPG framework allows computational biologists to easily and flexibly encode the Boolean network model of a GRN and simulate its state-transitions as well compute its steady state and attractors under flexible conditions (different activation rules and update schemes).

For many different purposes, simulating the behavior of gene regulatory networks using a Boolean network (i.e., only two-levels of activation per gene) is sufficient. However, in some conditions, this may not be enough [18, 28–30]. Indeed, even the simulation of “simple” systems such as the regulation of *lac operon* in bacteria may require at least an intermediary level of activation, besides the standard “on” and “off” level in a Boolean network. This has motivated us to extend the ASPG framework to allow for the representation of multiple levels of activation of each node, leading to the definition of what we call “multi-valued networks”. To this end we use an extension of ASP, known as Fuzzy ASP (FASP), which allows for the encoding of multi-valued propositions (instead of just Boolean propositions, as in the case of ASP). We pursued this direction by first developing our own solvers for evaluating FASP programs (Chapter 3, [31, 32]). In this respect, we proposed a new method to translate FASP into ASP and characterized the conditions by which we can obtain the answer sets of the former from the latter, and we implemented a solver based on this idea. In Chapter 4, we use this newly-developed solver to implement our framework for the simulation of multi-valued networks [33, 34]. We then perform extensive benchmark simulations to assess the correctness and computational feasibility of the implementation using both real biological networks as well as synthetic ones.

In the second part of the thesis, we turn our attention to another kind of network-based data modelling, and in particular, network-based feature selection and prediction. Within this topic, we work mainly with cancer omics and drug data, obtained from publicly-available repositories [35]. In Chapter 5, we describe our proposed network-based genomics feature selection method for modelling cancer drug response [36, 37]. The method integrates omics data (mutations, copy number alterations, gene expressions) and prior knowledge in the form of biological interaction networks in order to prioritize a set of features that are predictive of a certain cancer drug response, while at the same time being closely-connected within the network. Finally, in the last chapter, we discuss some conclusions and further outlook on these topics.

References

- [1] International Human Genome Sequencing Consortium et al. *Initial sequencing and analysis of the human genome*. Nature, 409(6822):860, 2001.
- [2] J Craig Venter, Mark D Adams, Eugene W Myers, Peter W Li, Richard J Mural, Granger G Sutton, Hamilton O Smith, Mark Yandell, Cheryl A Evans, Robert A Holt, et al. *The sequence of the human genome*. science, 291(5507):1304–1351, 2001.
- [3] Jason A Reuter, Damek V Spacek, and Michael P Snyder. *High-throughput sequencing technologies*. Molecular cell, 58(4):586–597, 2015.
- [4] Zachary D Stephens, Skylar Y Lee, Faraz Faghri, Roy H Campbell, Chengxiang Zhai, Miles J Efron, Ravishankar Iyer, Michael C Schatz, Saurabh Sinha, and Gene E Robinson. *Big data: astronomical or genomics?* PLoS biology, 13(7):e1002195, 2015.
- [5] Emily Clough and Tanya Barrett. *The gene expression omnibus database*. Statistical Genomics: Methods and Protocols, pages 93–110, 2016.
- [6] Sijia Huang, Kumardeep Chaudhary, and Lana X Garmire. *More is better: recent progress in multi-omics data integration methods*. Frontiers in genetics, 8:84, 2017.
- [7] Ana Conesa and Rafael Hernandez. *Chapter 16 - Omics Data Integration in Systems Biology: Methods and Applications*. In Virginia Garca-Caas, Alejandro Cifuentes, and Carolina Sim, editors, Applications of Advanced Omics Technologies: From Genes to Metabolites, volume 64 of *Comprehensive Analytical Chemistry*, pages 441 – 459. Elsevier, 2014.
- [8] Minoru Kanehisa and Susumu Goto. *KEGG: kyoto encyclopedia of genes and genomes*. Nucleic acids research, 28(1):27–30, 2000.
- [9] G Joshi-Tope, Marc Gillespie, Imre Vastrik, Peter D’Eustachio, Esther Schmidt, Bernard de Bono, Bijay Jassal, GR Gopinath, GR Wu, Lisa Matthews, et al. *Reactome: a knowledgebase of biological pathways*. Nucleic acids research, 33(suppl.1):D428–D432, 2005.
- [10] Jagodnik KM Koplev S He E Torre D Wang Z Dohlman AB Silverstein MC Lachmann A Kuleshov MV Ma’ayan A et al Keenan AB, Jenkins SL. *The Library of Integrated Network-Based Cellular Signatures NIH program: System-level cataloging of human cells response to perturbations*. Cell Systems, pii: S2405-4712(17):30490–30498, 2017.

- [11] Aravind Subramanian, Rajiv Narayan, Steven M Corsello, David D Peck, Ted E Natoli, Xiaodong Lu, Joshua Gould, John F Davis, Andrew A Tubelli, Jacob K Asiedu, et al. *A next generation connectivity map: L1000 platform and the first 1,000,000 profiles*. Cell, 171(6):1437–1452, 2017.
- [12] I Kuperstein, E Bonnet, HA Nguyen, D Cohen, E Viara, L Grieco, S Fourquet, L Calzone, C Russo, M Kondratova, et al. *Atlas of Cancer Signalling Network: a systems biology resource for integrative analysis of cancer data with Google Maps*. Oncogenesis, 4(7):e160, 2015.
- [13] Stuart A Kauffman. *The origins of order: Self-organization and selection in evolution*. In Spin glasses and biology, pages 61–100. World Scientific, 1992.
- [14] Michael Hecker, Sandro Lambeck, Susanne Toepfer, Eugene Van Someren, and Reinhard Guthke. *Gene regulatory network inference: data integration in dynamic models a review*. Biosystems, 96(1):86–103, 2009.
- [15] Hidde De Jong, Jean-Luc Gouzé, Céline Hernandez, Michel Page, Tewfik Sari, and Johannes Geiselmann. *Qualitative simulation of genetic regulatory networks using piecewise-linear models*. Bulletin of mathematical biology, 66(2):301–340, 2004.
- [16] Hidde De Jong. *Modeling and simulation of genetic regulatory systems: a literature review*. Journal of computational biology, 9(1):67–103, 2002.
- [17] Stuart A Kauffman. *Metabolic stability and epigenesis in randomly constructed genetic nets*. Journal of theoretical biology, 22(3):437–467, 1969.
- [18] Carlos Espinosa-Soto, Pablo Padilla-Longoria, and Elena R Alvarez-Buylla. *A gene regulatory network model for cell-fate determination during Arabidopsis thaliana flower development that is robust and recovers experimental gene expression profiles*. The Plant Cell Online, 16(11):2923–2939, 2004.
- [19] Luis Mendoza, Denis Thieffry, and Elena R Alvarez-Buylla. *Genetic control of flower morphogenesis in Arabidopsis thaliana: a logical analysis*. Bioinformatics, 15(7):593–606, 1999.
- [20] Maria I Davidich and Stefan Bornholdt. *Boolean network model predicts cell cycle sequence of fission yeast*. PloS one, 3(2):e1672, 2008.
- [21] Elena Dubrova and Maxim Teslenko. *A SAT-based algorithm for finding attractors in synchronous Boolean networks*. IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB), 8(5):1393–1399, 2011.

- [22] Ferhat Ay, Fei Xu, and Tamer Kahveci. *Scalable Steady State Analysis of Boolean Biological Regulatory Networks*. PLoS ONE, 4(12):e7992, 12 2009.
- [23] Mark DM Leiserson, Fabio Vandin, Hsin-Ta Wu, Jason R Dobson, Jonathan V Eldridge, Jacob L Thomas, Alexandra Papoutsaki, Younhun Kim, Beifang Niu, Michael McLellan, et al. *Pan-cancer network analysis identifies combinations of rare somatic mutations across pathways and protein complexes*. Nature genetics, 47(2):106, 2015.
- [24] Lieven PC Verbeke, Jimmy Van den Eynden, Ana Carolina Fierro, Piet Demeester, Jan Fostier, and Kathleen Marchal. *Pathway relevance ranking for tumor samples through network-based data integration*. PloS one, 10(7):e0133503, 2015.
- [25] Matan Hofree, John P Shen, Hannah Carter, Andrew Gross, and Trey Ideker. *Network-based stratification of tumor mutations*. Nature methods, 10(11):1108, 2013.
- [26] Mushthofa Mushthofa, Gustavo Torres, Yves Van de Peer, Kathleen Marchal, and Martine De Cock. *ASP-G: an ASP-based method for finding attractors in genetic regulatory networks*. Bioinformatics, 30(21):3086, 2014.
- [27] Vladimir Lifschitz. *What Is Answer Set Programming?*. In AAI, volume 8, pages 1594–1597, 2008.
- [28] Gilles Didier, Elisabeth Remy, and Claudine Chaouiya. *Mapping multivalued onto Boolean dynamics*. Journal of Theoretical Biology, 270(1):177 – 184, 2011.
- [29] Abhishek Garg, Luis Mendoza, Ioannis Xenarios, and Giovanni DeMicheli. *Modeling of multiple valued gene regulatory networks*. In Proceedings of the 29th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBS 2007), pages 1398–1404. IEEE, 2007.
- [30] Lucas Sanchez and Denis Thieffry. *Segmenting the fly embryo: a logical analysis of the pair-rule cross-regulatory module*. Journal of Theoretical Biology, 224(4):517–537, 2003.
- [31] Mushthofa Mushthofa, Steven Schockaert, and Martine De Cock. *A finite-valued solver for disjunctive fuzzy answer set programs*. In Proceedings of European Conference in Artificial Intelligence 2014, pages 645–650, 2014.
- [32] Mushthofa Mushthofa, Steven Schockaert, and Martine De Cock. *Solving Disjunctive Fuzzy Answer Set Programs*. In Proceedings of the 13th International Conference on Logic Programming and Non-monotonic Reasoning, pages 453–466, 2015.

- [33] Mushthofa Mushthofa, Steven Schockaert, and Martine De Cock. *Computing Attractors of Multi-Valued Gene Regulatory Networks using Fuzzy Answer Set Programming*. In Proceedings of the 2016 IEEE World Congress on Computational Intelligence, 2016.
- [34] Mushthofa Mushthofa, Steven Schockaert, Ling-Hong Hung, Kathleen Marchal, and Martine De Cock. *Modeling multi-valued biological interaction networks using fuzzy answer set programming*. Fuzzy Sets and Systems, 2018.
- [35] Francesco Iorio, Theo A Knijnenburg, Daniel J Vis, Graham R Bignell, Michael P Menden, Michael Schubert, Nanne Aben, Emanuel Gonçalves, Syd Barthorpe, Howard Lightfoot, et al. *A landscape of pharmacogenomic interactions in cancer*. Cell, 166(3):740–754, 2016.
- [36] Mushthofa Mushthofa, Lieven Verbeke, and Kathleen Marchal. *Network-based feature selection and prediction of cancer drug response*. Under preparation, 2018.
- [37] Mushthofa Mushthofa, Lieven Verbeke, and Kathleen Marchal. *Network-based method for feature selection and prediction of cancer drug response*. In Poster session of the 2017 Conference on Intelligent Systems for Molecular Biology, 2017.

2

Modelling GRNs with Boolean Networks and ASP

2.1 Introduction

Gene Regulatory Networks (GRNs) consist of genes, proteins and other regulatory molecules that undergo complex and dynamic interactions which drive gene expression, and ultimately, complex cellular behaviour. To be able to understand and predict this behaviour, various mathematical models have been developed that describe the dynamics of these GRNs. Different model formalisms have been used, as reviewed in [1]. One of the earliest models to describe GRNs are Boolean network models ([2]). Boolean network models are attractive because of their simplicity ([3]): by reducing the complexity of GRNs to qualitative logical models, Boolean network models are able to cope with the largely incomplete kinetic information of biological networks. Despite their highly simplified representation of biological reality, Boolean network models were shown to still grasp the important dynamic properties of GRNs, such as the networks' *attractors*. An attractor represents a stable set of states towards which the transiently changing gene expression values converge. Evolving towards an attractor thus corresponds to reaching a specific developmental stage (cell types, development stages of cells, *etc.*) or functional mode ([4], [2]), and the analysis of attractors in a regulatory network thus hints towards the functional modes of the regulatory network ([4]).

Current knowledge regarding GRNs is generally incomplete ([5]). Comparing simulated with observed attractors (states, e.g. developmental stages) of a

certain network model can aid in evaluating existing network models and/or predict missing information in the current knowledge. For instance, [6], [7] and more recently, [8] and [9] studied flower development in *Arabidopsis thaliana* using Boolean network models of which the network attractors corresponded to stable gene expression levels during the different stages of flower development. These models helped predicting mutant phenotypes and the existence of a yet-uncharacterized gene involved in the transition from the non-flowering to the flowering state. [10] and [11] used a Boolean network model and its steady states to describe the different stages in yeast cell cycle, where the stages of the cycle correspond to the strong attractors of the network. [12] explained the various states of the immune system with Boolean network models and [13], [14] and [15] used Boolean network models and their attractors to describe the cellular development of *Drosophila melanogaster*.

Key to simulating GRNs with Boolean network models is the choice of the proper assumptions. These assumptions refer to the activation rules and update scheme. Activation rules determine the way the activation state of each gene depends on the activation states of its interactors in the previous transition step. The update scheme determines how these activation states are updated, i.e., either synchronously or asynchronously. The exact choice of these assumptions largely determines the number and characteristics of the attractors. As in most cases, the true biological activation rules and update scheme are not known, one should be able to easily test different activation rules and schemes, as this allows to have an idea on the conditions under which the simulated network model would be able to capture an observed biological phenomena (boundary conditions).

Several computational tools have been developed to perform the computation of attractors in Boolean network models. [16] developed **genYsis**, which uses techniques involving *Binary Decision Diagrams* (BDD) to compute attractors. [17] used techniques based on *Temporal Logic* model checking in **Antelope**. [18] used state-space pruning and randomized state-space traversal methods to improve the scalability of the attractor computation. [19] used a Boolean Satisfiability (SAT) solver, typically used for combinatorial modelling and problem solving, to compute attractors of GRNs and obtained a better computational time and space efficiency compared to the BDD-based approach. More recently, [20] developed **geneFatt** based on the Reduced Order BDD (ROBDD) data structure which further improves the efficiency of the attractor computation.

Most of the above-mentioned systems to simulate Boolean network models in principle can perform simulations with different assumptions. However, changing these assumptions would require quite tedious modifications on these systems. For instance, using the SAT-approach ([19]), modifying the structure of the network and the updating rules would require updating the truth tables in the *cnet* format.

To allow for a more flexible simulation framework we developed ASP-G which

makes use of the declarative-programming paradigm Answer Set Programming (ASP) ([21]). The declarative nature of ASP allows one to specify and modify the domain-specific logic (here the definition of the network interactions, activation rules and update schemes) required to represent and solve the computational problem at hand (here dynamical modelling and attractor calculation) in an intuitive and modular way ([22]). To illustrate the flexibility of our approach, we applied it to calculate attractors of previously published Boolean network models of GRNs of different sizes and complexity, and different simulation assumptions. By trying to mimic previous results under diverse settings, we can show that the main advantage of our approach consists of making the modelling more flexible and less error prone, and therefore helps delineate the boundary conditions under which the biological conclusions based on simulations of Boolean network models are valid. At the same time, we also show that, with the use of fast and efficient ASP solvers, the computational efficiency of our method is in the same range as that of the most efficient dedicated methods for the simulation of Boolean network models and the calculation of their attractors.

2.2 Methods and Models

2.2.1 Boolean network modelling of GRNs

In our work, we adopt the definition of Boolean networks as described in [2]: a Boolean network model consists of network elements (nodes, here representing structural and regulatory genes/proteins) which can either be active (ON) or inactive (OFF) and interactions between these elements (edges, which represent the directed regulatory interactions between the genes). We define two types of regulatory interactions between interacting nodes i.e. activation (up-regulation) and inhibition (down-regulation). The activation state of a certain node at a certain time step is determined by a logical function of the activation states at the previous time step of its *interactors* (where the interactors of a node are defined as the incoming edges of a certain network node).

Formally, a Boolean network model $G(V, F)$ is defined by a set of nodes $V = \{x_1, \dots, x_n\}$ and a list of Boolean functions $F = (f_1, \dots, f_n)$. Each $x_i \in \{0, 1\}$, $i = 1, \dots, n$ is a binary variable and its value at time $t + 1$ can be determined by the values of some other nodes $x_{j_1(i)}, x_{j_2(i)}, \dots, x_{j_{k_i}(i)}$ at time t by means of a Boolean function $f_i \in F$. That is, there are k_i nodes assigned to x_i that determine its state. The activation state of every node x_i changes over time according to

$$x_i(t+1) = f_i \left(x_{j_1(i)}(t), x_{j_2(i)}(t), \dots, x_{j_{k_i}(i)}(t) \right)$$

A *state* s of a Boolean network is an assignment of $\{0, 1\}$ to each node x_i . Its *successor state* is the state resulting of applying f_i to each node x_i . The dynamics

Figure 2.1 A Boolean network model with three genes. Edges with arrowed tips are activating interactions and edges with blunt tips are repressing (inhibiting) links.

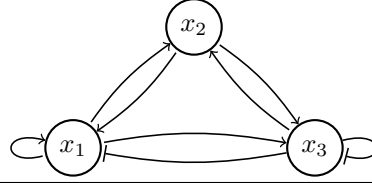
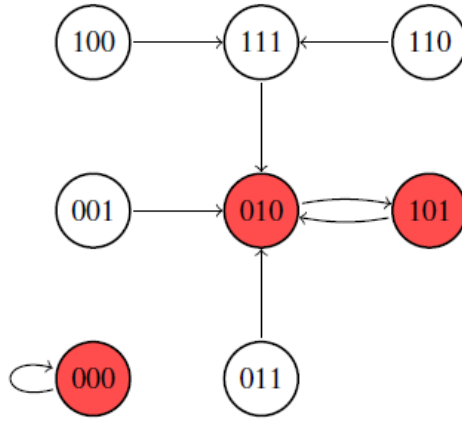


Figure 2.2 State Transition Graph of the example Boolean network given in Figure 2.1, when we assume synchronous update and the r^+ activation rule. The nodes in this graph are states of the network. Since the network has 3 nodes, there are $2^3 = 8$ possible states, as shown by the 8 nodes in this graph. There are two attractors in this network : $\{(000)\}$ and $\{(010), (101)\}$



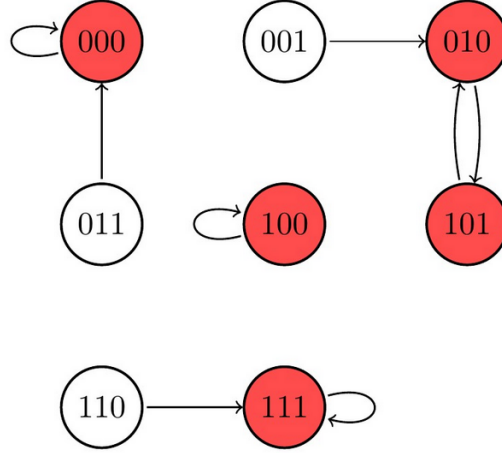
of the network consists of transitions between network states. An example of a Boolean network is given in Figure 2.1. This network has 3 nodes, denoted by x_1, x_2 and x_3 , and interactions between these nodes, represented by the edges. The dynamics of the network can be described using a State Transition Graph (STG) as given in Figure 2.2 & 2.3.

Attractors in a Boolean network model are defined as in [16] and [18].

Definition 1. Let S be a set of states of a Boolean network model. S is an attractor if and only if the following conditions are satisfied:

1. The set of the successor states of all the states in S is equal to S .
2. For each $s_i \in S$, once it is visited, the probability of revisiting s_i in a finite number of state transitions is equal to 1.

Figure 2.3 State Transition Graph of the example Boolean network given in Figure 2.1, when we assume synchronous update and the r^* activation rule.



2.2.2 Representing Boolean network models of GRNs and computing their attractors using ASP

ASP-G framework

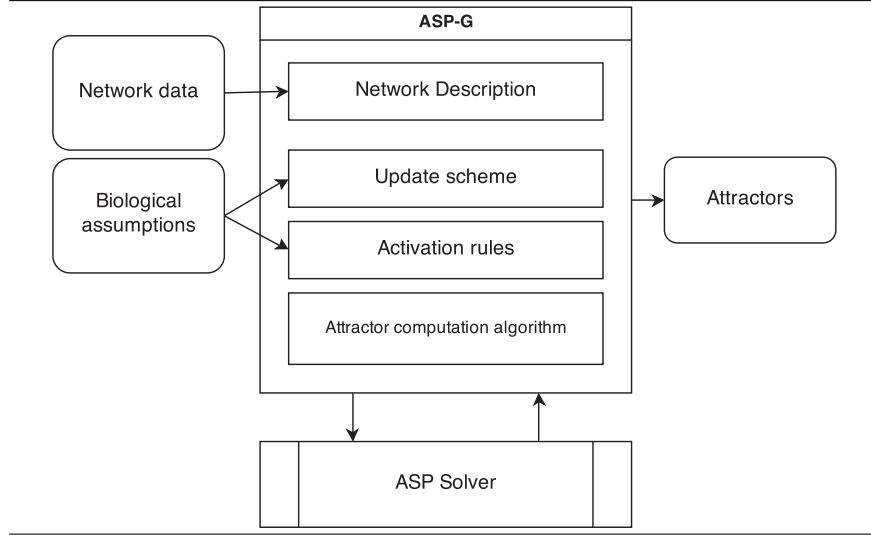
Our framework for modelling GRNs and computing attractors, called ASP-G, uses Answer Set Programming (ASP). ASP is a declarative programming paradigm ([21]), which is typically used to solve combinatorial search problems ([22]). The architecture of ASP-G is shown in Figure 2.4. ASP-G consists of four main modules/parts of the system: the network description, the update scheme, the activation rules and the attractor computation algorithm. The following describes each of these modules.

Network description module

This module contains the description of the structure of the network, encoded in a set of facts provided by the user. Because of the declarative nature of ASP, such a description can be written succinctly in an intuitive format. For example, the following set of facts is used to describe the network depicted in Figure 2.1:

$$\begin{aligned}
 &gene(x_1). \quad gene(x_2). \quad gene(x_3). \\
 &activates(x_1, x_1). \quad activates(x_1, x_2). \quad activates(x_1, x_3). \\
 &activates(x_2, x_1). \quad activates(x_2, x_3). \quad inhibits(x_3, x_1). \\
 &activates(x_3, x_2). \quad inhibits(x_3, x_3).
 \end{aligned}$$

To compare the declarative specification of the network with the more classically-used SAT notation [19], we give the network specification corresponding to the

Figure 2.4 Architecture of ASP-G.

Boolean network depicted in in Figure 2.1 in *cnet* format, as follows:

```
.n 1 3 1 2 3      .n 2 2 1 3      .n 3 3 1 2 3
--1 0              1- 1              --1 0
1-0 1              -1 1              1-0 1
-10 1              00 0              -10 1
000 0              000 0
```

Each gene in the network is written with its label, e.g. `.n 1` for node x_1 , followed by the number of regulators and then their labels, e.g. `(3 1 2 3)`. Next a truth table follows that determines the behavior of the network (e.g. `--1 0` in the truth table for x_1 means that whenever x_3 is active then x_1 will be inhibited irrespective of the values of x_1 and x_2).

Activation rules

This module determines the activation rules used to update the activation state of each gene at each transition step, based on the intended assumption by the user. We implemented two frequently used activation rules:

1. A gene will be active in a subsequent time step $t + 1$ if at time step t at least one active interactor is an activator and no active interactors that act as inhibitors are present.
2. A gene will be active in a subsequent time step $t + 1$ if there are more active activators than active inhibitors among its interactors at time step t .

In ASP-G this first type of activation rule is encoded as

$$\begin{aligned}
 r_1^+ : \text{active}(X, T) &\leftarrow A > 0, I = 0, \#act(X, A, T - 1), \\
 &\quad \#inh(X, I, T - 1), gene(X), \\
 &\quad T > 0. \\
 r_2^+ : \text{inhibited}(X, T) &\leftarrow \text{not } active(X, T), gene(X), \\
 &\quad T > 0.
 \end{aligned}$$

The second type of activation rule is encoded as

$$\begin{aligned}
 r_1^* : \text{active}(X, T) &\leftarrow A - I > 0, \#act(X, A, T - 1), \\
 &\quad \#inh(X, I, T - 1), gene(X), \\
 &\quad T > 0. \\
 r_2^* : \text{inhibited}(X, T) &\leftarrow I - A \geq 0, \#act(X, A, T - 1), \\
 &\quad \#inh(X, I, T - 1), gene(X), \\
 &\quad T > 0.
 \end{aligned}$$

To illustrate the flexibility of ASP to express nearly any possible activation rule, we implemented a more specific type of rules in which the activation of a certain gene is expressed as a free-form gene-specific Boolean function of all the interactors of that gene, such as the one used in [9] and [15]. For example, the activation rule of a gene G1 might be expressed as a Boolean function of the form:

$$G1 = G1 \text{ and } (\text{not } G2 \text{ or not } G3)$$

where G2 and G3 are two other genes involved in the network. In this case, G1 has a self-activating interaction, while G2 and G3 act as inhibitors for G1. To accommodate such an activation rule in ASP-G, we first convert the Boolean function into a Disjunctive Normal Form (DNF). For example, the activation rule given above is rewritten into the following:

$$\begin{aligned}
 G1 &= G1 \text{ and not } G2 \\
 G1 &= G1 \text{ and not } G3
 \end{aligned}$$

and then encoded in ASP as follows:

$$\begin{aligned}
 \text{active}(G1, T) &\leftarrow T > 0, \text{active}(G1, T - 1), \\
 &\quad \text{inhibited}(G2, T - 1) \\
 \text{active}(G1, T) &\leftarrow T > 0, \text{active}(G1, T - 1), \\
 &\quad \text{inhibited}(G3, T - 1)
 \end{aligned}$$

Update scheme

Two update schemes were adopted in ASP-G: synchronous and asynchronous updates. In the synchronous update scheme, we assume that all genes in the network are updated simultaneously per evaluated time step. This implies that the transitions between the network states are deterministic, i.e., for every state visited over time, only one possible successor state exists. In the asynchronous update, no assumptions of *synchronicity* is made, and genes may be updated at different time steps. Therefore, transitions are non-deterministic: there may be several possible next state after a certain transition. As illustrated in the Figure 2.2 and 2.5, an asynchronous update scheme can result in a drastically different STG and leads to different attractors.

Both the synchronous and the asynchronous updating were implemented in ASP-G. For the synchronous update scheme, we use the following rules to generate initial activation states of the genes:

$$\begin{aligned} active(X, 0) &\leftarrow gene(X), not\ inhibited(X, 0). \\ inhibited(X, 0) &\leftarrow gene(X), not\ active(X, 0). \end{aligned}$$

and then use the following rules (in relation with the activation rules described previously) to determine the activation state of each gene at each time step t :

$$\begin{aligned} active(X, t) &\leftarrow active(X, t-1), not\ inhibited(X, t). \\ inhibited(X, t) &\leftarrow inhibited(X, t-1), not\ active(X, t). \end{aligned}$$

For the asynchronous update scheme, we need to add the following rules, in addition to the previously described rules:

$$\begin{aligned} changed(X, T) &\leftarrow active(X, T), inhibited(X, T-1), \\ &\quad gene(X), T > 0. \\ changed(X, T) &\leftarrow inhibited(X, T), active(X, T-1), \\ &\quad gene(X), T > 0. \\ &\leftarrow \#changed(N, X, T), N \geq 2, \\ &\quad gene(X), T > 0. \end{aligned}$$

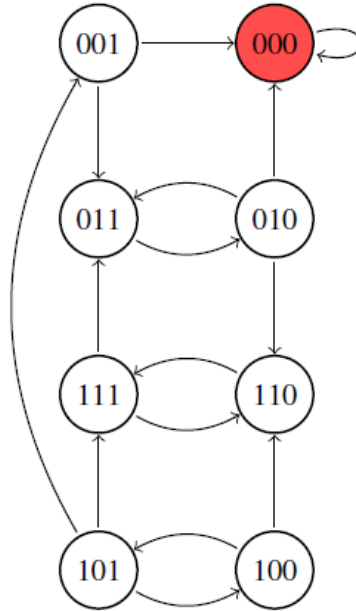
To obtain a better performance in the asynchronous case, we apply the STG reduction technique, as described in Figure 2.5.

Computing the attractors

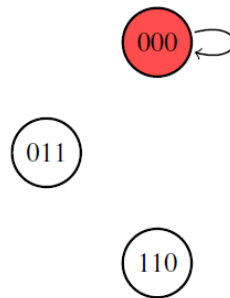
The attractor computation in ASP-G is performed by Algorithm 1, which is based on the algorithm by [19]. The main idea used in Algorithm 1 is to identify attractors by looking at identical states in transition paths of certain lengths in the STG of the network. Since there are exponentially many possible states in an STG

Figure 2.5 An example of a State Transition Graph of the Boolean network given in Figure 2.1, when we assume asynchronous update. Instead of the two attractors as in the synchronous case, here we only have one attractors: $\{(000)\}$. As can be seen, transitions are non-deterministic. For example: there are two possible next states after (001): (000) or (011). To find the attractors in such case, we apply the following STG reduction principle: remove states from the STG which contain more than one outgoing (asynchronous) transition. It is safe to remove such states, since they cannot satisfy the second condition of the definition of a state in an attractor (cf. Definition 1). Therefore, we need only to look attractors in the reduced STG obtained from such removal process. For example, applying this principle to the STG in Figure A, we obtain the reduced STG (Figure B), containing the only attractor $\{(000)\}$

(a) Figure A



(b) Figure B



(in relation to the number of nodes in the network), explicit enumeration of all the states in an STG is unfeasible for larger networks. ASP-G avoids this explicit enumeration by implicitly simulating the dynamics of the network. Furthermore, once a state is identified as part of an attractor, it can be removed from the STG to prune the search space. An illustration on how the algorithm works is given in Figure 2.6.

Path generation and state removal are being done using ASP rules and ASP constraints, respectively, as shown in Figure S5. To further increase the efficiency of the computation, we use the incremental ASP approach described in [23] and the *clasp* solver from the Potassco ASP suite ([24]).

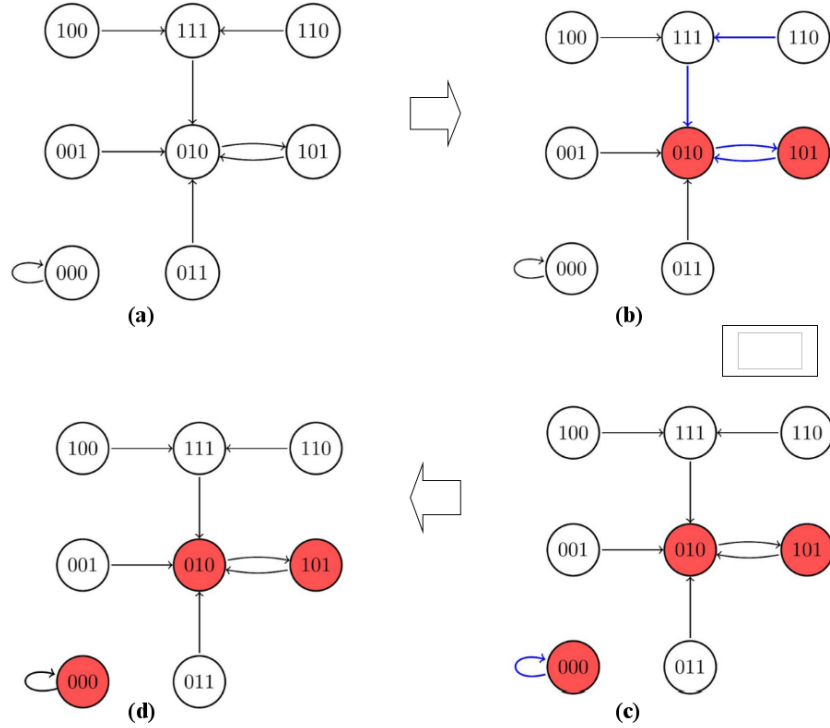
Algorithm 1 Algorithm to compute attractors in ASP-G

```

1:  $\{P$  is the ASP program with the rules of the network $\}$ 
2:  $k = n$ 
3: attractor_is_found = False
4: attractors =  $\emptyset$ 
5: while ASP finds a path of length  $k$  as an answer set in  $P$  do
6:    $\{s = \langle s_1, s_2, \dots, s_k \rangle$  is the path found $\}$ 
7:   for  $j = k - 1$  to 1 do
8:     if  $s_j = s_k$  then
9:       attractors = attractors  $\cup \{\langle s_{j+1}, \dots, s_k \rangle\}$ 
10:      attractor_is_found = True
11:      {The attractors already found are forbidden in  $P$ }
12:      for  $s$  in  $\{s_k, \dots, s_{j+1}\}$  do
13:        {The states are added as constraints for the next path}
14:         $P = P \cup \{\leftarrow \text{active}(X_1, T), \text{inhibited}(X_2, T), \dots | X_i \in s\}$ 
15:      end for
16:      break
17:    end if
18:  end for
19:  if attractor_is_found then
20:    attractor_is_found = False
21:  else
22:     $k = 2 \cdot k$ 
23:  end if
24: end while
25: return attractors

```

Figure 2.6 These figures illustrate the process of applying Algorithm 1 on the STG in Figure 2.2. In (a), we have the original STG. We repeatedly find path of arbitrary length, k , where k varies from 2 to 2^n , where n is the number of nodes in the Boolean network. Consider the case when $k = 4$. One path we might obtain is $(110) \rightarrow (111) \rightarrow (010) \rightarrow (101) \rightarrow (010)$ (Figure (b)). By extracting two identical states from this path (010) , we obtain the endpoints of a cycle, indicating an attractor, in this case $\{(010), (101)\}$. By disallowing these states (Figure (c)), we find the only other path of length 4, which is $(000) \rightarrow (000) \rightarrow (000) \rightarrow (000) \rightarrow (000)$, from which we extract the second attractor, $\{(000)\}$. Disallowing this state, we find no further path of length 4 (Figure (d)). Continuing in this manner until $k = 2^n$ guarantees that all attractors are identified. The process of generating and checking the path are performed by using ASP rules. The addition of disallowed states is performed by using ASP constraints.



2.3 Results

2.3.1 ASP-G: a novel framework for the simulation and attractor computation of GRNs with Boolean network models

Simulating Boolean network models implies that activation rules have to be defined to decide how a gene is activated by its interactors. The exact choice of the assumptions largely determines the outcome and the number and characteristics of the attractors. Figures 2.2, 2.5 & 2.3 show that, for the example Boolean network in Figure 2.1, the choices of activation rules and update scheme can result in a different network behavior, and thus, different sets of attractors. As it is often not known in advance which assumptions best match the biological reality of the modeled GRN, testing different assumptions is advisable.

To have a generic framework that allows testing different assumptions, ASP-G implements three different activation rules: the first one adopted by [10], [18] and [11] in which a gene is considered to be activated if the majority of its active incoming interactors (interactors that are themselves active) has an activating role. Otherwise, the gene will become inactive. This is referred to as the r^* rule in ASP-G. A second one adopted by [25] and [16] assumes that a gene is activated only when there is at least one active activator amongst its active incoming interactors and no inhibitor. This is referred to as the r^+ rule in ASP-G. In addition, we implemented a third more detailed activation rule in which the activation for each gene is expressed as a free-form Boolean function of its activators and inhibitors. This type of rules better grasps the complexity of true biological interactions and includes more detailed information on the specifics of the interactions, and was used for instance in [9], [14] and [15].

Related to the update scheme, a choice has to be made between updating the network elements simultaneously (synchronously) versus at different points (asynchronously). Earlier work by [26] and [2] assumed synchronicity in their modelling, mainly because of computational efficiency reasons. However, the assumption of synchronicity was challenged in [27] and [28], who argued that, for many biological systems, assuming an asynchronous update scheme is more realistic. Subsequent work on Boolean network models ([29], [16], [18], [30] and [15]) mainly applied asynchronous update schemes. Therefore, in ASP-G we implemented both the synchronous and the asynchronous update scheme.

2.3.2 Simulation Results

To test the correctness of ASP-G in simulating Boolean network models and computing attractors, we applied ASP-G on previously published Boolean network models of GRNs and compared the obtained attractors with the originally published ones. Like any other formalism to find attractors in Boolean network mod-

els, ASP-G detects exhaustively all attractors of the network. Therefore, a correct result corresponds to an exact match in the attractor set identified by ASP-G and that present in the reference publications.

First we tested ASP-G on the relatively small GRNs involved in budding and fission yeast cell cycle analyzed in [10], [18] and [11], as well as on the network model of the GRN from the Th cell differentiation described in [16]. We used the activation rules and update schemes that were also applied in the original studies, except for the data set of [16], in which we applied also the synchronous update scheme in addition to the originally applied asynchronous one (see Table 2.1).

As expected for each of these results, the attractors found by ASP-G match exactly the ones found by the reference papers (data not shown). The table also shows that ASP-G performs relatively fast for these small networks. We also observe that the attractors in the synchronous case often coincide with those in the asynchronous case. This is due to the fact that simple attractors (i.e., attractors that have only one state) are more commonly found, and that they are shared between synchronous and asynchronous update schemes. Only for the fission yeast cell network under the r^* activation rules there is a difference in the number of synchronous attractors (13) and asynchronous attractors (15). The fact that there are more attractors in the asynchronous case might seem counter-intuitive, since the reduced STG used to calculate these attractors contains at most as many nodes as the synchronous STG, and often less. However, the original STG for the asynchronous case typically contains more edges than in the synchronous case, and these additional edges can account for more attractors.

To show the flexibility of ASP-G in expressing different types of activation rules, we also encoded the Boolean network model of the GRN involved in *A. thaliana* flower development originally described in [9] with our ASP-G framework. This network model consists of 13 genes and uses gene specific update rules as described in Section 2.2.2 **Activation rules**. As in the original publication, we applied a synchronous update scheme. ASP-G correctly recapitulated all 10 attractors of the network as described in the original paper [9]. The computation took only 0.479 seconds.

To test ASP-G on a larger network, we use the network data from the T-helper (Th) cell differentiation described by [25]. The purpose of the study was to find evidence supporting (or contradicting) the traditional view that the genes involved in the regulation of the two types of Th cells, Th1 and Th2, have counter-regulatory interaction. Similar to what has been done in [25], we first computed the attractors of the original network in the presence of all genes and then searched for attractors in different single-gene knock-out networks *in silico* to test the effect of knocking out intracellular genes towards the attractors of the network. To show the added value of using different simulation assumptions, we also performed the computations using an asynchronous update mechanism, as opposed to only synchronous

update scheme, as performed by [25]. The results are presented in the Table 2.2.

In terms of computational efficiency, the result shows that ASP-G is able to perform relatively well for the moderately-sized Th cell network. For the asynchronous case, attractor computation required 0.6 seconds on average. For the synchronous case where more attractors were found, the computation times took 71.9 seconds on average. When applying the synchronous update scheme as used in [25], ASP-G reproduced the four attractors: Th0, Th1, Th2 and ThX, as in the original paper. However, when trying to reproduce the attractors in the gene knock-out setting, we found discrepancies with the results reported by [25]. These discrepancies were caused by mistakes in the SAT-based truth table used in the original publication. Table 2.2 shows the corrected results for completeness. Note that these mistakes do not affect the biological conclusions made in the original publication. However, it illustrates that specifying larger networks with rather complicated behavior becomes cumbersome and error prone in paradigms like SAT, whereas this is much less the case for a declarative approach such as ASP-G.

The existence of the ThX attractor and the observed pattern of gene activities in the attractors of the knock-out networks caused the authors of [25] to conclude that the active genes in Th1 and Th2 cells do not play a counter-regulatory roles towards each other, contrary to what is traditionally believed. However, when using the asynchronous update scheme, we noted that the attractor ThX is no longer obtained. A similar pattern occurs for the knock-out networks, where the use of the asynchronous update scheme drastically changes the set of detected attractors compared to those detected using a synchronous update scheme: the number of attractors found with the asynchronous update scheme for the knock-out networks ranges only between 2-5, whereas the number of attractors found in the synchronous case ranged between 286-1154. This finding suggests that the occurrence of the ThX attractor and the existence of a large number of attractors in the knock-out networks as found in [25] are only valid under the synchronous update scheme. It thus defines the boundary conditions under which the conclusions of [25] are valid and highlights the relevance of performing modelling under different scenarios, as offered by ASP-G, to put biological conclusions in perspective.

2.4 Conclusion

In this chapter, we presented ASP-G, a modular system to simulate Boolean network models of GRNs and to subsequently compute their attractors. ASP-G is based on the declarative ASP programming paradigm which has already been previously applied in the context of biological network data analysis and modelling (see, e.g., [31], [32], [33], [34] and [35]). Recently, [36] showed in a theoretical comparison between Boolean networks and the underlying semantics of ASP, that a strong mathematical relation exists between the attractors/steady states of

Boolean networks and the notion of *stable models* commonly used in ASP. We built upon this earlier result in our proposed method, ASP-G.

The main added value of ASP-G is in its declarativity and modularity: it allows users to easily test different update schemes and activation rules when simulating the dynamics of their Boolean network model by selecting and modifying the appropriate modules. In addition, the fact that ASP-G is based on a declarative language makes it less error prone than other approaches such as SAT which depend on the definition of difficult to interpret and tedious to construct truth tables. Using an underlying declarative programming paradigm also makes ASP-G easily extendable to other parameter settings. Decoupling the problem definition from its solution thus allows for a greater flexibility compared to other *ad hoc* systems such as **genYsis** ([16]) and **geneFAtt** ([20]), where assumptions such as update scheme and activation rules are already built into the system.

We showed the correctness of ASP-G in simulating Boolean network models and obtaining attractors under different assumptions by successfully recapitulating the detection of attractors of previously published studies. Relying on a modular and flexible declarative programming paradigm definitely comes at the expense of being slower than the more dedicated systems to compute attractors, such as **genYsis** ([16]) and **geneFAtt** ([20]). However, in terms of computational efficiency, ASP-G proved to be quite fast (for small networks i.e. up to 23 genes computations are below a second, for larger networks i.e. up to 51 genes, the longest computation time took under 4.5 minutes). Also, ongoing research in ASP solvers ([37]) will make it possible for ASP to reach a point where it outperforms other logic paradigms. This is definitely the case when comparing ASP with Binary (or multiple) Decision Diagrams ([38]) used to calculate attractors in Boolean Networks models for GRNs ([17], [29]) as they suffer from memory explosion when the size of the network starts to become large ([39]).

For larger-sized networks, any exhaustive method will face a challenge, as the state space of the network increases exponentially w.r.t. the number of nodes in the network. When dealing with such larger networks, methods that avoid an exhaustive search as in [18] might become more suitable under these conditions. Conclusively, ASP-G is tailored to simulate Boolean network models of GRNs and to compute attractors in a *diagnostic* mode, where one wants to test different update schemes and activation rules in order to find the setting that best matches experimental data or to correctly delineate the boundary conditions under which the biological conclusions based on these simulations are valid.

Table 2.1 ASP-G results and running times for common GRNs found in the literature. Network: describes the original network model. Genes: number of genes present in the network; Attractors: number of detected attractors; Update mechanism: synchronous versus asynchronous updating was used as described in the methods section; Activation rules: r^* activation rules indicates that a gene becomes active when it has more active activating interactors than active inhibiting ones, while the r^+ activation rules indicate that a gene becomes active if it has at least one active activating gene and no inhibiting ones. Time: running time on a Dell Latitude D820 notebook.

Network	Reference	Genes	Attractors	Update Mechanism	Activation Rules	Time
Common Yeast cell cycle	[10]	11	7	Synchronous	r^*	1.507
	[18]	11	7	Asynchronous	r^*	0.134
Fission Yeast cell cycle	[11]	10	13	Synchronous	r^*	1.653
	[18]	10	15	Asynchronous	r^*	0.371
Th cell differentiation	/	23	3	Synchronous	r^+	0.270
	[16]	23	3	Asynchronous	r^+	0.206

Table 2.2 Running times and the number of synchronous and asynchronous attractors computed by ASP-G for the gene knockout experiments described by Pedicini et al. [25]. Inaccuracies in the original results in [25] are corrected in this work, resulting in minor discrepancies between result represented in this table (e.g., the number of attractors) and the one in [25].

knock-out	Sync #	Sync time (s)	Async #	Async time (s)
COT	594	71.048	3	0.650
GATA3	322	42.019	3	0.610
IKBKB	594	71.068	3	0.570
IRAK	612	71.948	3	0.580
IRF4	604	31.906	3	0.550
ITK	596	70.156	3	0.560
JAK1	594	69.040	3	0.620
JAK3	560	63.204	3	0.610
LCK	595	70.276	3	0.550
MAF	594	70.852	3	0.580
NFAT	604	27.458	3	0.590
NFKB	594	69.536	3	0.570
NIK	594	70.220	3	0.610
PI3K	594	70.128	3	0.610
PLCPG	596	72.365	3	0.600
SHP1	594	68.616	3	0.610
SLP76	594	70.536	3	0.630
SOCS1	978	104.907	5	1.000
STAT1	1154	267.881	3	0.570
STAT4	612	71.116	3	0.550
STAT6	286	8.709	2	0.500
TBET	358	54.211	3	0.540
VAV1	595	70.972	3	0.660
ZAP70	594	68.268	3	0.530

References

- [1] Hidde De Jong. *Modeling and simulation of genetic regulatory systems: a literature review*. Journal of computational biology, 9(1):67–103, 2002.
- [2] Stuart A. Kauffman. *The origins of order: Self-organization and selection in evolution*. Oxford university press, 1993.
- [3] Ilya Shmulevich, Edward R Dougherty, and Wei Zhang. *From Boolean to probabilistic Boolean networks as models of genetic regulatory networks*. Proceedings of the IEEE, 90(11):1778–1792, 2002.
- [4] Hidde De Jong and Michel Page. *Search for steady states of piecewise-linear differential equation models of genetic regulatory networks*. Computational Biology and Bioinformatics, IEEE/ACM Transactions on, 5(2):208–222, 2008.
- [5] Richard Rottger, Ulrich Ruckert, Jan Taubert, and Jan Baumbach. *How little do we actually know? On the size of gene regulatory networks*. Computational Biology and Bioinformatics, IEEE/ACM Transactions on, 9(5):1293–1300, 2012.
- [6] Luis Mendoza and Elena R Alvarez-Buylla. *Dynamics of the Genetic Regulatory Network for Arabidopsis thaliana Flower Morphogenesis*. Journal of theoretical biology, 193(2):307–319, 1998.
- [7] Luis Mendoza, Denis Thieffry, and Elena R Alvarez-Buylla. *Genetic control of flower morphogenesis in Arabidopsis thaliana: a logical analysis*. Bioinformatics, 15(7):593–606, 1999.
- [8] Carlos Espinosa-Soto, Pablo Padilla-Longoria, and Elena R Alvarez-Buylla. *A gene regulatory network model for cell-fate determination during Arabidopsis thaliana flower development that is robust and recovers experimental gene expression profiles*. The Plant Cell Online, 16(11):2923–2939, 2004.
- [9] Yara-Elena Sanchez-Corrales, Elena R Alvarez-Buylla, and Luis Mendoza. *The Arabidopsis thaliana flower organ specification gene regulatory network determines a robust differentiation process*. Journal of theoretical biology, 264(3):971–983, 2010.
- [10] Fangting Li, Tao Long, Ying Lu, Qi Ouyang, and Chao Tang. *The yeast cell-cycle network is robustly designed*. Proceedings of the National Academy of Sciences of the United States of America, 101(14):4781–4786, 2004.
- [11] Maria I Davidich and Stefan Bornholdt. *Boolean network model predicts cell cycle sequence of fission yeast*. PloS one, 3(2):e1672, 2008.

- [12] Marcelle Kaufman, Jacques Urbain, and René Thomas. *Towards a logical analysis of the immune response*. Journal of theoretical biology, 114(4):527–561, 1985.
- [13] Lucas Sánchez and Denis Thieffry. *A Logical Analysis of the Drosophila Gap-gene System*. Journal of theoretical Biology, 211(2):115–141, 2001.
- [14] Réka Albert and Hans G Othmer. *The topology of the regulatory interactions predicts the expression pattern of the segment polarity genes in Drosophila melanogaster*. Journal of theoretical biology, 223(1):1–18, 2003.
- [15] Aitor González, Claudine Chaouiya, and Denis Thieffry. *Logical modelling of the role of the Hh pathway in the patterning of the Drosophila wing disc*. Bioinformatics, 24(16):i234–i240, 2008.
- [16] Abhishek Garg, Ioannis Xenarios, Luis Mendoza, and Giovanni DeMicheli. *An efficient method for dynamic analysis of gene regulatory networks and in silico gene perturbation experiments*. In Research in Computational Molecular Biology, pages 62–76. Springer, 2007.
- [17] Gustavo Arellano, Julian Argil, Eugenio Azpeitia, Mariana Benitez, Miguel Carrillo, Pedro Gongora, David Rosenblueth, and Elena Alvarez-Buylla. *“Antelope”: a hybrid-logic model checker for branching-time Boolean GRN analysis*. BMC Bioinformatics, 12(1):490, 2011.
- [18] Ferhat Ay, Fei Xu, and Tamer Kahveci. *Scalable Steady State Analysis of Boolean Biological Regulatory Networks*. PLoS ONE, 4(12):e7992, 12 2009.
- [19] Elena Dubrova and Maxim Teslenko. *A SAT-based algorithm for finding attractors in synchronous Boolean networks*. IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB), 8(5):1393–1399, 2011.
- [20] Desheng Zheng, Guowu Yang, Xiaoyu Li, Zhicai Wang, Feng Liu, and Lei He. *An Efficient Algorithm for Computing Attractors of Synchronous And Asynchronous Boolean Networks*. PloS one, 8(4):e60593, 2013.
- [21] Vladimir Lifschitz. *What Is Answer Set Programming?*. In AAI, volume 8, pages 1594–1597, 2008.
- [22] Thomas Eiter, Giovambattista Ianni, and Thomas Krennwallner. *Answer set programming: A primer*. In Reasoning Web. Semantic Technologies for Information Systems, pages 40–110. Springer, 2009.
- [23] Martin Gebser, Roland Kaminski, Benjamin Kaufmann, Max Ostrowski, Torsten Schaub, and Sven Thiele. *Engineering an incremental ASP solver*. In Logic Programming, pages 190–205. Springer, 2008.

- [24] Martin Gebser, Benjamin Kaufmann, Roland Kaminski, Max Ostrowski, Torsten Schaub, and Marius Schneider. *Potassco: The Potsdam answer set solving collection*. AI Communications, 24(2):107–124, 2011.
- [25] Marco Pedicini, Fredrik Barrenäs, Trevor Clancy, Filippo Castiglione, Eivind Hovig, Kartiek Kanduri, Daniele Santoni, and Mikael Benson. *Combining network modelling and gene expression microarray analysis to explore the dynamics of Th1 and Th2 cell regulation*. PLoS computational biology, 6(12):e1001032, 2010.
- [26] René Thomas. *Boolean formalization of genetic control circuits*. Journal of theoretical biology, 42(3):563–585, 1973.
- [27] Rene Thomas. *Regulatory networks seen as asynchronous automata: a logical description*. Journal of Theoretical Biology, 153(1):1–23, 1991.
- [28] Inman Harvey and Terry Bossomaier. *Time out of joint: Attractors in asynchronous random boolean networks*. In Proceedings of the Fourth European Conference on Artificial Life, pages 67–75. MIT Press, Cambridge, 1997.
- [29] Aurélien Naldi, Denis Thieffry, and Claudine Chaouiya. *Decision diagrams for the representation and analysis of logical models of genetic networks*. In Computational Methods in Systems Biology, pages 233–247. Springer, 2007.
- [30] Abhishek Garg, Alessandro Di Cara, Ioannis Xenarios, Luis Mendoza, and Giovanni De Micheli. *Synchronous versus asynchronous modelling of gene regulatory networks*. Bioinformatics, 24(17):1917–1925, 2008.
- [31] Martin Gebser, Torsten Schaub, Sven Thiele, Björn Usadel, and Philippe Veber. *Detecting inconsistencies in large biological networks with answer set programming*. In Logic Programming, pages 130–144. Springer, 2008.
- [32] Steve Dworschak, Susanne Grell, Victoria J Nikiforova, Torsten Schaub, and Joachim Selbig. *Modeling biological networks by action languages via answer set programming*. Constraints, 13(1-2):21–65, 2008.
- [33] Martin Gebser, Carito Guziolowski, Mihail Ivanchev, Torsten Schaub, Anne Siegel, Sven Thiele, and Philippe Veber. *Repair and Prediction (under Inconsistency) in Large Biological Networks with Answer Set Programming*. In KR, 2010.
- [34] Martin Gebser, Arne König, Torsten Schaub, Sven Thiele, and Philippe Veber. *The BioASP library: ASP solutions for systems biology*. In Tools with Artificial Intelligence (ICTAI), 2010 22nd IEEE International Conference on, volume 1, pages 383–389. IEEE, 2010.

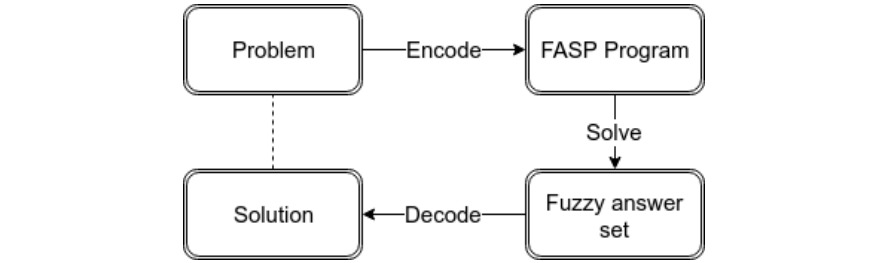
- [35] Fabien Corblin, Eric Fanchon, Laurent Trilling, Claudine Chaouiya, and Denis Thieffry. *Automatic inference of regulatory and dynamical properties from incomplete gene interaction and expression data*. In *Information Processing in Cells and Tissues*, pages 25–30. Springer, 2012.
- [36] Katsumi Inoue. *Logic programming for Boolean networks*. In *Proceedings of the Twenty-Second international joint conference on Artificial Intelligence-Volume Volume Two*, pages 924–930. AAAI Press, 2011.
- [37] Martin Gebser, Roland Kaminski, Benjamin Kaufmann, Torsten Schaub, and Bettina Schnor. *Cluster-based ASP solving with claspar*. In *Logic Programming and Nonmonotonic Reasoning*, pages 364–369. Springer, 2011.
- [38] Chang-Yeong Lee. *Representation of Switching Circuits by Binary-Decision Programs*. *Bell System Technical Journal*, 38(4):985–999, 1959.
- [39] Edmund Clarke, Orna Grumberg, Somesh Jha, Yuan Lu, and Helmut Veith. *Progress on the state explosion problem in model checking*. In *Informatics*, pages 176–194. Springer, 2001.

3

Modelling and solving problems using FASP

3.1 Introduction

Fuzzy Answer Set Programming (FASP) is a declarative programming framework aimed at solving combinatorial search/optimization problems in continuous domains [1, 2]. It extends Answer Set Programming (ASP [3, 4]), a well known declarative language that allows users to specify combinatorial search and optimization problems in an intuitive way, as we did in Chapter 2. ASP stands out among other logic-based programming frameworks by being more purely declarative (compared to, e.g., Prolog), allowing for a more concise and intuitive encoding, while at the same time being highly expressive. The availability of efficient solvers, which are able to solve hard real-world problems, has also significantly contributed to the popularity of this modelling language. In the wake of ASP's extensive development over the last decades [5], FASP has recently been gaining more attention as well, including the development of FASP solvers that enable the use of FASP for real-world problem solving beyond toy examples. In [6], the authors developed a prototype FASP solver using the method of fuzzy set approximations. They improved the solver further by using a translation to Satisfiability Modulo Theory (SMT) [7], which increased the performance of their solver significantly on many test instances. We have also developed our own FASP solver, based on the idea of a translation to ASP, and making use of currently available ASP solvers [8, 9].

Figure 3.1 Work flow for solving problems using (F)ASP

In general, the work-flows used for problem solving with FASP or ASP are quite similar, and can be summarized as follows (Figure 3.1): (1) first we encode the problem as a (fuzzy) logic program, containing all the required facts, rules and/or constraints required to define the conditions of the problem; (2) we then call a (F)ASP solver, which will generate (any/all) answer sets from the specified program; and (3) finally we decode the answer sets to obtain the solutions.

In this chapter, we give an introduction to FASP, as well as a description of our state-of-the-art FASP solver and its use in practice. The remainder of this chapter is structured as follows. In the next section, we provide a high-level explanation of how ASP is typically used for solving problems, and the role that an extension to FASP can play in applications. In Section 3.3, we present the syntax and semantics of FASP, and explain how FASP programs can be used to encode problems. Section 3.4 and 3.5 subsequently explain how our solver finds the answer sets of a FASP program. Finally, in Section 3.6, we report the results of some benchmark experiments we performed on the solver.

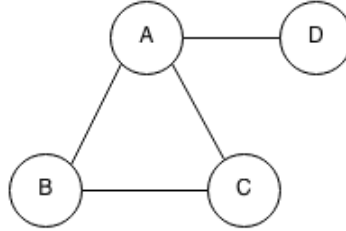
3.2 Modelling problems as logic programs

Declarative programming allows one to solve problems by encoding/expressing them, usually in a rule-based logical language, allowing the programmer to specify the problem in an intuitive manner, without having to explicitly say “how” to solve the problem. It is then the task of the “solver” (which is an algorithm, or its implementation) to find the solutions in accordance with the problem specification. For example, the well-known Graph 3-coloring problem can be tackled in ASP using the following encoding:

$$col(X, red) \vee col(X, green) \vee col(X, blue) \leftarrow node(X) \quad (3.1)$$

$$\leftarrow col(X, c), col(Y, c), edge(X, Y) \quad (3.2)$$

As is common in logic programming, rules are written in the form $\alpha \leftarrow \beta$ with β the antecedent (body) of the rule and α the consequent (head). When the

Figure 3.2 Example graph

consequent of a rule is *false*, it is left empty, as in the second rule above, enforcing that the antecedent of the rule must be *false* for the overall rule to be satisfied. Rule (1.1) intuitively expresses that we wish to find all possible 3-colorings of the nodes in a graph; the predicates *node/1* and *col/2* are used to encode the nodes available in the graph and the colors assigned to them, respectively. The second rule, which is a “logical constraint”, eliminates all the coloring schemes in which two nodes *X* and *Y* sharing an edge receive the same color *c*. Given the above program and a set of inputs representing all the nodes and edges of the graph, an ASP solver, such as *clasp* [10] or *DLV* [11] then searches the *answer sets* of the program, representing the possible solutions to the problem. For example, given a graph in Figure 3.2, we can encode the input to the program using the set of facts $\{node(a), \dots, node(d), edge(a, b), edge(a, c), edge(a, d), edge(b, c)\}$. An ASP solver will then compute the answer sets, each of which corresponds to a solution. For example, one answer set will contain the atoms $col(a, red)$, $col(b, blue)$, $col(c, green)$ and $col(d, blue)$, which indeed corresponds to a valid coloring of the given graph.

The declarative nature of the syntax of ASP and the efficiency of its solvers makes ASP a popular approach for solving combinatorial search problems. ASP has found applications in a wide range of areas, including cryptography [12], hardware design [13], data mining [14], the space shuttle decision system [14], bioinformatics [15–17] and many others [4, 18].

FASP extends the expressiveness of ASP by allowing the use of *fuzzy logic* in place of Boolean logic. The use of fuzzy logic in FASP means that predicates can have a continuum of possible truth degrees (usually taken from $[0, 1]$), rather than the discrete choices *false* and *true*. This enables the use of an ASP-like declarative specification of problems involving continuous variables. As a simple toy example, consider the problem of deciding whether to give a generous tip in a restaurant, depending on the quality of the food and service, as follows:

$$good_food \leftarrow \text{not } bland \quad (3.3)$$

$$generous \leftarrow good_food \otimes good_service \quad (3.4)$$

Here, it can be more natural to express criteria such as *bland* and *good_service* as gradual properties. The truth degree of e.g. *bland* then expresses to what extent the property is satisfied. These truth degrees can then be combined using fuzzy logic operators.

3.3 Syntax and Semantics of FASP

Several different variants of FASP have been considered by different authors. Here we will focus on the variant studied in [19], whose semantics is based on Łukasiewicz logic. Similar to ASP, FASP assumes the availability of a set of propositional atom symbols, $\mathcal{B}_{\mathcal{P}}$. Alternatively, we can also consider a first-order syntax with predicate symbols, in which case, $\mathcal{B}_{\mathcal{P}}$ is the set of *ground* atoms obtained from the available predicate and constant symbols. *Grounding* is essentially the process of replacing the variables occurring in any predicate symbols with the available constant symbols. A (classical) literal is either a constant symbol \bar{c} where $c \in [0, 1] \cap \mathbb{Q}$, an atom a or a classical negation literal $\neg a$. An *extended* literal is a classical literal l or a *negation-as-failure* (NAF) literal **not** l . Intuitively, classical negation differs from NAF in that the former expresses our **knowledge** about something being **not** true, whereas the latter expresses our inability to **prove** that something is true.

A *head/body expression* is a formula defined recursively as follows:

- any classical literal is a head expression;
- any extended literal is a body expression;
- if α and β are head (resp. body) expressions, then $\alpha \otimes \beta$, $\alpha \oplus \beta$, $\alpha \vee \beta$ and $\alpha \bar{\wedge} \beta$ are also head (resp. body) expressions.

A FASP program is a finite set of rules of the form:

$$\alpha \leftarrow \beta$$

where α is a head expression (called the *head* of the rule) and β is a body expression (called the *body* of the rule). As in classical ASP, we write $H(r)$ and $B(r)$ to denote the head and body of a rule r , respectively. A FASP rule of the form $a \leftarrow \bar{c}$ for a classical literal a and a constant c is called a *fact*.

A FASP rule of the form $\bar{c} \leftarrow \beta$, with $c \in [0, 1] \cap \mathbb{Q}$ is called a *constraint*. A rule which does not contain any application of the operator **not** is called a *positive* rule. A rule which has at most one literal in the head is called a *normal* rule. A FASP program is called [*positive*, *normal*] iff it only contains [positive, normal] rules, respectively. Conversely, a [rule/program] which is not normal is called a disjunctive [rule/program]. A positive normal program which has no constraints is called a *simple* program.

The semantics of FASP is usually defined in relation to a chosen truth lattice $\mathcal{L} = \langle L, \leq_L \rangle$ [20]. We consider two types of truth lattices: the infinitely valued lattice $\mathcal{L}_\infty = \langle [0, 1], \leq \rangle$ and the finitely valued lattices $\mathcal{L}_k = \langle \mathbb{Q}_k, \leq \rangle$, for an integer $k \geq 1$, where $\mathbb{Q}_k = \{\frac{0}{k}, \frac{1}{k}, \dots, \frac{k}{k}\}$. Such a choice is usually determined by the nature or the goal of the application. If each proposition can only take a finite number of different truth levels, then using the \mathcal{L}_k would be more appropriate. In this case, FASP is used for modelling discrete problems, and thus remains very close to classical ASP. For modelling continuous problems, or if we do not want to fix the number of truth degrees in advance, we need to use the semantics based on \mathcal{L}_∞ .

For any choice of lattice \mathcal{L} (among the considered possibilities \mathcal{L}_∞ or \mathcal{L}_k), an interpretation of a FASP program \mathcal{P} is a function $I : \mathcal{B}_\mathcal{P} \mapsto \mathcal{L}$ which can be extended to expressions and rules as follows:

- $I(\bar{c}) = c$, for a constant $c \in \mathcal{L}$
- $I(\alpha \otimes \beta) = \max(I(\alpha) + I(\beta) - 1, 0)$
- $I(\alpha \oplus \beta) = \min(I(\alpha) + I(\beta), 1)$
- $I(\alpha \vee \beta) = \max(I(\alpha), I(\beta))$
- $I(\alpha \bar{\wedge} \beta) = \min(I(\alpha), I(\beta))$
- $I(\mathbf{not} \alpha) = 1 - I(\alpha)$
- $I(\alpha \leftarrow \beta) = \min(1 - I(\beta) + I(\alpha), 1)$

for appropriate expressions α and β . Here, the operators **not**, \otimes , \oplus , \vee , $\bar{\wedge}$ and \leftarrow denote the Łukasiewicz negation, t-norm, t-conorm, maximum, minimum and implication, respectively.

An interpretation I is consistent iff $I(l) + I(\neg l) \leq 1$ for each $l \in \mathcal{B}_\mathcal{P}$. We say that a consistent interpretation I of \mathcal{P} satisfies a FASP rule r iff $I(r) = 1$. This condition is equivalent to $I(H(r)) \geq I(B(r))$. An interpretation is a model of a program \mathcal{P} iff it satisfies every rule of \mathcal{P} . For interpretations I_1, I_2 , we write $I_1 \leq I_2$ iff $I_1(l) \leq I_2(l)$ for each $l \in \mathcal{B}_\mathcal{P}$, and $I_1 < I_2$ iff $I_1 \leq I_2$ and $I_1 \neq I_2$. We call a model I of \mathcal{P} a *minimal* model if there is no other model J of \mathcal{P} such that $J < I$.

For a positive FASP program \mathcal{P} , a model I of \mathcal{P} is called a *fuzzy answer set* of \mathcal{P} iff it is a minimal model of \mathcal{P} . For non-positive programs, a common way to define the answer set semantics, in the case of classical ASP is to the so-called Gelfond-Lifschitz (GL) reduct to transform the program into a positive one, given a guess of which atoms are true. For a non-positive FASP program \mathcal{P} , a generalization of the GL reduct was defined in [21, 22] as follows: the reduct of a rule r w.r.t. an interpretation I is the positive rule r^I obtained by replacing each occurrence of

not a by the constant $\overline{I(\text{not } a)}$. The reduct of a FASP program \mathcal{P} w.r.t. an interpretation I is then defined as the positive program $\mathcal{P}^I = \{r^I \mid r \in \mathcal{P}\}$. A model I of \mathcal{P} is called a fuzzy answer set of \mathcal{P} iff I is a fuzzy answer set of \mathcal{P}^I . The set of all the fuzzy answer sets of a FASP program \mathcal{P} is denoted by $\mathcal{ANS}(\mathcal{P})$. A simple FASP program has exactly one fuzzy answer set. A positive FASP program may have no, one or several fuzzy answer sets. In particular, disjunctive rules can generate many fuzzy answer sets, in general. A FASP program is called *consistent* iff it has at least one fuzzy answer set, and *inconsistent* otherwise.

Example 1. Consider the FASP program \mathcal{P}_1 which has the following rules:

$$\{a \leftarrow \text{not } c, b \leftarrow \text{not } c, c \leftarrow a \oplus b\}$$

It can be seen that under both the truth-lattice \mathcal{L}_3 and \mathcal{L}_∞ , the interpretation $I_1 = \{(a, \frac{1}{3}), (b, \frac{1}{3}), (c, \frac{2}{3})\}$ is a minimal model of $\mathcal{P}_1^{I_1}$, and hence it is an answer set of \mathcal{P}_1 . However, the program admits no answer sets under any \mathcal{L}_k , where k is a positive integer not divisible by 3.

Once a problem has been specified, or encoded, as a FASP program, the next main steps are (1) to automatically determine the answer sets of the FASP program, and (2) to map them back to solutions of the original problem.

3.4 Solving finite satisfiability of FASP using ASP

The results in [23] and [24] suggest that solving FASP programs using finite methods could potentially be useful. Call a fuzzy answer set A of \mathcal{P} a k -answer set of \mathcal{P} iff the truth values of the atoms in A are taken from the set \mathbb{Q}_k . Then it can be seen that every k -answer set of a FASP program \mathcal{P} under the infinite-valued truth-lattice \mathcal{L}_∞ is also an answer set of \mathcal{P} under the finite-valued truth-lattice \mathcal{L}_k . This means that we can find every answer set of \mathcal{P} under \mathcal{L}_∞ by iteratively finding its answer sets under \mathcal{L}_k , for $k \geq 1$. A result in [23] shows that exponentially many k need to be checked to exhaustively find all answer sets of the program under \mathcal{L}_∞ . As we will see in Section 5, however, in practice usually only a small number of values for k needs to be checked.

We will show how answer sets of FASP programs under a finite-valued truth lattice \mathcal{L}_k can be found using a reduction to classical ASP. In the next sections, we will show how we can rewrite FASP rules into equivalent forms prior to the translation (to make the translation process more efficient) as well as the details of the translation itself. Finally, we will analyse the conditions under which this approach is also successful for finding answer sets in the infinitely-valued truth lattice \mathcal{L}_∞ .

3.4.1 FASP rule rewriting

Before we perform the translation to ASP, we rewrite the FASP rules into an equivalent set of rules which follow a certain “standardized form”, in order to make the translation simpler and more efficient. First, if a rule $r \in \mathcal{P}$ contains a constant symbol \bar{c} in the body, replace r with

$$H(r) \leftarrow B(r)[\bar{c}/p]$$

where p is a fresh atom symbol. Here, $x[y/z]$ is obtained by replacing each occurrence of y in x with z . If $c > 0$, add the rule $p \leftarrow c$ to the program. If a rule $r \in \mathcal{P}$ contains a constant symbol \bar{c} in the head, replace r with

$$H(r)[\bar{c}/p] \leftarrow B(r)$$

where a fresh atom symbol p is used for every constant appearing in the program. If $c < 1$, add the rules $\{p \leftarrow \bar{c}, \bar{c} \leftarrow p\}$ to the program. It is not hard to see that these replacements do not change the meaning of the program.

Next, we rewrite each rule such that the resulting rules contain at most one application of the logical operators ($\oplus, \otimes, \bar{\wedge}, \vee, \mathbf{not}$). The idea is to recursively split each application of the operators on composite expressions by defining new auxiliary atoms to capture the truth value of each of the composite expressions, and then replace the original rule with a set of equivalent rules. For example, for a rule $r \in \mathcal{P}$ of the form $a \leftarrow \beta * \gamma$ where $*$ $\in \{\oplus, \otimes, \bar{\wedge}, \vee\}$, a is a classical literal and β and γ are composite expressions, we replace r with the following set of rules $\{a \leftarrow p * q, p \leftarrow \beta, q \leftarrow \gamma\}$ where p and q are fresh atom symbols. While, for a rule $r \in \mathcal{P}$ of the form $\alpha * \beta \leftarrow c$ where $*$ $\in \{\oplus, \otimes, \bar{\wedge}, \vee\}$ and α and β are composite expressions and c is a classical literal, we replace r with the following rules: $\{p * q \leftarrow c, p \leftarrow \alpha, \alpha \leftarrow p, q \leftarrow \beta, \beta \leftarrow q\}$ where p and q are fresh atom symbols.

The following proposition holds.

Proposition 1. *Using a finite number of rewriting steps, we can convert any program \mathcal{P} into an equivalent program $Rw(\mathcal{P})$ containing only rules of the following forms:*

1. A fact $a \leftarrow \bar{c}$ for an atom a and a constant $\bar{c}, c \in (0, 1]$.
2. A constraint $\bar{c} \leftarrow a$ for an atom a and a constant $\bar{c}, c \in [0, 1)$.
3. A rule with no operator in the body nor in the head $a \leftarrow b$.
4. A rule with NAF-literal in the body $a \leftarrow \mathbf{not} b$ for atoms a and b .
5. A rule with a binary operator in the body $a \leftarrow b * c$, with a, b and c atoms and $*$ $\in \{\otimes, \oplus, \bar{\wedge}, \vee\}$.

6. A rule with a binary operator in the head $a * b \leftarrow c$, with a, b and c atoms and $*$ $\in \{\otimes, \oplus, \vee, \wedge\}$.

and that the size of $Rw(\mathcal{P})$ is $O(n \cdot m)$, where n is the number of rules in \mathcal{P} and m is the maximum number of atom occurrences in the rules of \mathcal{P} .

3.4.2 Translation to classical ASP

To find the answer sets of \mathcal{P} under \mathcal{L}_k , we perform a translation of each rule of \mathcal{P} into ASP rules parametrized by k . Consider a FASP program \mathcal{P} and an integer k . Assume that each rule of \mathcal{P} follows the “standardized” forms as described in Proposition 1. We will translate \mathcal{P} into a classical ASP program $Tr(\mathcal{P}, k)$. First, we assume the availability of atom symbols a_i for every $a \in \mathcal{B}_{\mathcal{P}}$ and $1 \leq i \leq k$ to be used in $Tr(\mathcal{P}, k)$.

We translate each rule of \mathcal{P} as follows:

1. For a fact $r \in \mathcal{P}$ of the form: $a \leftarrow \bar{c}, c \in (0, 1]$ we add the fact $a_j \leftarrow$ to $Tr(\mathcal{P}, k)$, where $j = k * c$.
2. For a constraint $r \in \mathcal{P}$ of the form $\bar{c} \leftarrow a, c \in [0, 1)$ we add a constraint $\leftarrow a_{j+1}$ in $Tr(\mathcal{P}, k)$, where $j = k * c$.
3. A FASP rule of the form: $a \leftarrow b$ can be easily translated into classical ASP as $\{a_i \leftarrow b_i \mid 1 \leq i \leq k\}$
4. A FASP rule of the form: $a \leftarrow b \otimes c$ is equivalent to saying that $I(a) \geq \max(I(b) + I(c) - 1, 0)$ for every answer set I of \mathcal{P} . This means that to obtain $I(a) \geq \frac{i}{k}$ for some $1 \leq i \leq k$, we can choose $I(b) = \frac{j}{k}$ for any $i \leq j \leq k$ and then $I(c) = 1 - \frac{j-i}{k}$. This corresponds to the following set of ASP rules:

$$\{a_i \leftarrow b_j \wedge c_{k-j+i} \mid 1 \leq i \leq k, i \leq j \leq k\}$$

5. A FASP rule of the form: $a \leftarrow b \oplus c$ is equivalent to saying that $I(a) \geq \min(I(b) + I(c), 1)$ for every answer set I of \mathcal{P} . This means that to obtain $I(a) \geq \frac{i}{k}$ for some $1 \leq i \leq k$, we can choose $I(b) \geq \frac{j}{k}$ for some $0 \leq j \leq i$ and then $I(c) \geq \frac{i-j}{k}$. This can be translated as the following set of ASP rules:

$$\{a_i \leftarrow b_i, a_i \leftarrow c_i, a_i \leftarrow b_j \wedge c_{i-j} \mid 1 \leq i \leq k, 0 < j < i\}$$

6. A FASP rule of the form $a \leftarrow b \vee c$ implies that $I(a) \geq \max(I(b), I(c))$ in every answer set I . This can be translated as the following set of ASP rules

$$\{a_i \leftarrow b_i, a_i \leftarrow c_i \mid 1 \leq i \leq k\}$$

7. A FASP rule of the form $a \leftarrow b \bar{\wedge} c$ can be translated into

$$\{a_i \leftarrow b_i \wedge c_i \mid 1 \leq i \leq k\}$$

8. For the FASP rule $a \oplus b \leftarrow c$ we first create fresh atom symbols $p_{s,t}$, where $0 \leq s, t \leq k$ such that $1 \leq s + t \leq k$. Each $p_{s,t}$ encodes the situation where a and b have truth values $\frac{s}{k}$ and $\frac{t}{k}$, respectively. We then encode the semantics saying that when c has a truth value of $\frac{i}{k}$, then the sum of the truth values of a and b should be at least $\frac{i}{k}$. We must also ensure that only “minimal choices” are generated in the answer sets. For example, if $I(c) = \frac{i}{k}$ and $I(a) = \frac{j}{k}$, we must eliminate the choices which generate $I(b) > \frac{i-j}{k}$. We use the following set of ASP rules for this translation.

$$\begin{aligned} &\{p_{0,i} \vee p_{1,i-1} \vee \dots \vee p_{i-1,1} \vee p_{i,0} \leftarrow c_i \mid 1 \leq i \leq k\} \cup \\ &\{a_i \leftarrow p_{i,j}, b_j \leftarrow p_{i,j} \mid 0 \leq i, j \leq k\} \cup \\ &\{p_{i+1,j-1} \leftarrow p_{i,j} \wedge a_{i+1} \mid 0 \leq i \leq k-1, 1 \leq i+j \leq k\} \cup \\ &\{p_{i-1,j+1} \leftarrow p_{i,j} \wedge b_{j+1} \mid 0 \leq j \leq k-1, 1 \leq i+j \leq k\} \end{aligned}$$

The first two sets of rules “distribute” the truth value of c to a and b , while the last two sets of rules ensure that only minimal models are generated by eliminating the non-minimal ones. For example, if we also have the fact $a \leftarrow \frac{1}{k}$ in \mathcal{P} , then the rule

$$p_{k-1,1} \leftarrow p_{k,0} \wedge a_1$$

will eliminate the (otherwise generated) non-minimal answer set A of $Tr(\mathcal{P}, k)$ containing a_1 and b_k , which corresponds to a (non-minimal) fuzzy model I of \mathcal{P} having $I(a) = \frac{1}{k}$ and $I(b) = 1$.

9. For the FASP rule $a \otimes b \leftarrow c$ a similar construct as the translation scheme for $a \oplus b \leftarrow c$ can be used, as follows: create atom symbols $p_{s,t}$, where $1 \leq s, t \leq k$ such that $s + t > k$, with a similar meaning as before. The rule $a \otimes b \leftarrow c$ can then be translated as:

$$\begin{aligned} &\{p_{k,i} \vee p_{k-1,i+1} \vee \dots \vee p_{i,k} \leftarrow c_i \mid 1 \leq i \leq k\} \cup \\ &\{a_i \leftarrow p_{i,j}, b_j \leftarrow p_{i,j} \mid 1 \leq i, j \leq k, i+j > k\} \cup \\ &\{p_{i+1,j-1} \leftarrow p_{i,j} \wedge a_{i+1} \mid 1 \leq i < k, 1 \leq j \leq k, i+j > k\} \cup \\ &\{p_{i-1,j+1} \leftarrow p_{i,j} \wedge b_{j+1} \mid 1 \leq i \leq k, 1 \leq j < k, i+j > k\} \end{aligned}$$

10. A FASP rule of the form $a \vee b \leftarrow c$ states that $\max(I(a), I(b)) \geq I(c)$ in every answer set I of \mathcal{P} . Hence, we can translate it into the following set of ASP rules: $\{a_i \vee b_i \leftarrow c_i \mid 1 \leq i \leq k\}$

11. Similar to the previous translation, we can translate the FASP rule $a \bar{\wedge} b \leftarrow c$ into the following set of ASP rules: $\{a_i \leftarrow c_i, b_i \leftarrow c_i \mid 1 \leq i \leq k\}$
12. For a rule $a \leftarrow \mathbf{not} b$ which states that $I(a) \geq 1 - I(b)$ for every answer set I , we use the following set of ASP rules:

$$\{a_i \leftarrow \mathbf{not} b_{k-i+1} \mid 1 \leq i \leq k\}$$

which enforces the constraint $I(a) \geq 1 - I(b)$ while at the same time preserving the NAF-semantics.

Finally, we must add the set of rules

$$\{a_i \leftarrow a_{i+1} \mid a \in \mathcal{B}_{\mathcal{P}}, 1 \leq i \leq k-1\}$$

into $Tr(\mathcal{P}, k)$ to ensure that the atoms a_i are consistent with the interpretation that the truth value of a is at least $\frac{i}{k}$. We can show the following result.

Proposition 2. *The number of rules in $Tr(\mathcal{P}, k)$ is $O(n \cdot k^2)$, where n is the number of rules in \mathcal{P} .*

Now, consider a function \mathcal{M}_k , mapping a classical interpretation A to a fuzzy interpretation I , defined as follows:

$$I(a) = \mathcal{M}_k(A)(a) = \max\{\frac{i}{k} \mid a_i \in A\}$$

We can show that the following proposition holds:

Proposition 3. *A is an answer set of $Tr(\mathcal{P}, k)$ iff $I = \mathcal{M}_k(A)$ is an answer set of \mathcal{P} under the truth-lattice \mathcal{L}_k .*

For the case where the truth-lattice \mathcal{L}_{∞} is assumed, one must perform the translation and find k -answer sets for (possibly exponentially) many values of k . If no constant symbols appear in \mathcal{P} , we can start looking for k -answer sets for $k = 1, 2, \dots$ and so on. However, if \mathcal{P} contains a constant symbol \bar{c} , where $c = \frac{a}{b}$ for some integers a and b with $\gcd(a, b) = 1$, then translating a rule such as $a \leftarrow \frac{\bar{a}}{b}$ into an ASP rule $a_j \leftarrow$ where $j = a/b * k$ requires k to be a multiple of b . Therefore, in the search for k -answer sets using the translation above, one must always choose a value of k which is a multiple of every denominator of the constants appearing in the program. In other words, if there are n constant symbols $\{\frac{a_1}{b_1}, \dots, \frac{a_n}{b_n}\}$ in \mathcal{P} , then we choose values of k which are divisible by the least common multiple of b_1, \dots, b_n . The following proposition provides the result for the infinite-valued truth lattice.

Proposition 4. *For every answer set I of a FASP program \mathcal{P} under the truth-lattice \mathcal{L}_{∞} , there exists a positive integer k such that $I = \mathcal{M}_k(A)$ for some answer set A of $Tr(\mathcal{P}, k)$.*

For normal programs, we additionally have the following proposition.

Proposition 5. *If \mathcal{P} is a normal FASP program and A is answer set of $Tr(\mathcal{P}, k)$, then $I = \mathcal{M}_k(A)$ is an answer set of \mathcal{P} under the \mathcal{L}_∞ .*

Example 2. *Consider again the program \mathcal{P}_1 from Example 1. Obviously, $Rw(\mathcal{P}_1) = \mathcal{P}_1$. Furthermore, one can check that $Tr(\mathcal{P}_1, 1)$ and $Tr(\mathcal{P}_1, 2)$ have no answer sets. However, the ASP program $Tr(\mathcal{P}_1, 3)$ containing the following rules:*

$$\begin{aligned} & \{a_i \leftarrow \text{not } c_{3-i+1} \mid 1 \leq i \leq 3\} \cup \\ & \{b_i \leftarrow \text{not } c_{3-i+1} \mid 1 \leq i \leq 3\} \cup \\ & \{c_i \leftarrow a_i, c_i \leftarrow b_i, c_i \leftarrow a_j \wedge b_{i-j} \mid 1 \leq i \leq 3, 1 \leq j < i\} \cup \\ & \{p_i \leftarrow p_{i+1} \mid i = 1, 2, p \in \{a, b, c\}\} \end{aligned}$$

does have an answer set, namely $A_1 = \{a_1, b_1, c_1, c_2\}$ which corresponds to the only answer set I_1 of \mathcal{P}_1 under \mathcal{L}_∞ (i.e. $\mathcal{M}_3(A_1) = I_1$).

For disjunctive programs, the result from Proposition 5 does not necessarily follow. Consider the following example.

Example 3. *Program \mathcal{P}_2 has the following rules: $\{a \oplus b \leftarrow \bar{1}, a \leftarrow b, b \leftarrow a\}$. $Tr(\mathcal{P}_2, 1)$ has one answer set, namely $A_1 = \{a_1, b_1\}$. However, $I_1 = \mathcal{M}_1(A_1) = \{(a, 1), (b, 1)\}$ is not an answer set of \mathcal{P}_2 , whose only answer set is $I_2 = \{(a, 0.5), (b, 0.5)\}$.*

For disjunctive programs, we only have the following weaker result.

Proposition 6. *If \mathcal{P} is a disjunctive FASP program and A is an answer set of $Tr(\mathcal{P}, k)$, then $I = \mathcal{M}_k(A)$ is an answer set of \mathcal{P} under the infinitely-valued truth lattice \mathcal{L}_∞ iff there is no other model J of \mathcal{P}^I such that $J < I$.*

This means that for disjunctive programs, when our method has found a k -answer set, we still need to verify whether it is an answer set under \mathcal{L}_∞ . This can easily be checked using a mixed integer programming solver, or any other method for entailment checking in Łukasiewicz logic.

3.5 Evaluating disjunctive rules

In this section we will identify a large fragment of the class of disjunctive FASP programs which can be reduced in polynomial time to a normal FASP program. Subsequently, we will show how this reduction can be used to develop a sound method for finding answer sets of general disjunctive FASP programs.

Following [25], the head-cycle free (HCF) ASP programs are programs whose positive dependency graphs (see Section 3.3) do not contain cycles that go through

two literals occurring in the head of a rule. In [26], it was shown that any HCF program can be reduced to an equivalent program using the *shift* operator. Briefly, the shift operator replaces any rule $a_1 \vee \dots \vee a_n \leftarrow B$ with the set of rules $R = \{a_i \leftarrow B \wedge NB_i \mid 1 \leq i \leq n\}$, where $NB_i = \bigwedge_{1 \leq j \leq n, j \neq i} \text{not } a_j$. For example, the program $\{a \vee b \leftarrow\}$ can be reduced to the equivalent program $\{a \leftarrow \text{not } b, b \leftarrow \text{not } a\}$. However, when we introduce head cycles, such as in the program $\mathcal{P}_3 = \{a \vee b \leftarrow, a \leftarrow b, b \leftarrow a\}$, shifting is no longer guaranteed to produce an equivalent normal program. Interestingly, in the case of FASP programs, the syntactically similar program $\mathcal{P}_4 = \{a \oplus b \leftarrow \bar{1}, a \leftarrow b, b \leftarrow a\}$ is equivalent to the shifted version: $\mathcal{P}'_4 = \{a \leftarrow \text{not } b, b \leftarrow \text{not } a, a \leftarrow b, b \leftarrow a\}$. In fact, we will show that any strict disjunctive FASP program can be reduced to an equivalent normal FASP program in this way. This explains the observation in [19] that allowing disjunction in the head does not affect the computational complexity of strict FASP programs. For programs with disjunction in the body, shifting does not always yield an equivalent FASP program, for e.g., $\mathcal{P}_4 \cup \{a \leftarrow a \oplus a\}$ is not equivalent to $\mathcal{P}'_4 \cup \{a \leftarrow a \oplus a\}$. Intuitively, we can safely shift disjunctive rules if there is no interaction between disjunctions in the body and a head cycle. We will now formalize this idea based on the notion of a self-reinforcing cycle.

3.5.1 SRCF programs

We first extend the notion of *proof* for classical disjunctive programs as defined in [25]. Let $\mathbf{0}$ denotes the interpretation that assigns zeros to all atoms. Let I be an interpretation of a program \mathcal{P} , and let a be any atom such that $I(a) > 0$. Then, a support of a in \mathcal{P} w.r.t. I is defined as a sequence of rules $r_1, r_2, \dots, r_n \in \mathcal{P}^I$ such that:

1. $\mathbf{0}(\text{Body}(r_1)) > 0$
2. $a \in \text{Head}(r_n)$
3. $\sum_{a \in \text{Lit}(\text{Head}(r_i))} I(a) = I(\text{Body}(r_i))$ for all $1 \leq i \leq n$
4. For every $x \in \text{Lit}(\text{Body}(r_i))$ there exists a $j < i$ such that $x \in \text{Lit}(\text{Head}(r_j))$

We characterize the non-existence of self-reinforcing cyclic rules in a program using the following definition: for a FASP program \mathcal{P} , we say that \mathcal{P} is self-reinforcing cycle free (SRCF) w.r.t. an atom a , iff we can find a stratification function $f : \mathcal{B}_{\mathcal{P}} \rightarrow \mathbb{N}$, such that for every rule $r \in \mathcal{P}$ which contains a , it holds that:

1. $f(x) \geq f(y)$ for every $x \in \text{Lit}(\text{Head}(r))$ and $y \in \text{Lit}(\text{Body}(r))$
2. If $r \equiv x \leftarrow y \oplus z$, then $f(x) > f(y)$ and $f(x) > f(z)$.

Intuitively, we can see that a program \mathcal{P} is SRCF w.r.t. atom a iff the dependency graph does not contain any cycle which goes through a and involves at least one rule with disjunction in the body. We say that a program \mathcal{P} is SRCF iff it is SRCF w.r.t every atom $a \in \mathcal{B}_{\mathcal{P}}$. The following theorem characterizes the notion of support for SRCF programs.

Theorem 3.5.1 (Support). *Let \mathcal{P} be an SRCF program and let I be a consistent interpretation. Then I is an answer set of \mathcal{P} iff:*

1. I is a model of \mathcal{P} .
2. Every $a \in \{x \mid I(x) > 0\}$ has a support in \mathcal{P} w.r.t. I .

The proof runs parallel to the proof of Theorem 2.3 in [25] by noting that support plays a similar role for the answer sets of SRCF FASP programs as proof does for HCF ASP programs.

Example 4. Consider program $\mathcal{P}_5 = \{a \oplus b \leftarrow \bar{1}, c \leftarrow b \otimes \text{not } a, c \leftarrow a\}$. It is clear that $I = \{(a, 0.3), (b, 0.7), (c, 0.4)\}$ is an answer set of \mathcal{P}_5 . In accordance with Theorem 1, for each of a, b and c , we can take $r_1 = a \oplus b \leftarrow \bar{1}, r_2 = c \leftarrow b \otimes 0.7$ as the support of these atoms in \mathcal{P}_5 w.r.t. I . Furthermore, any $J > I$ obtained by increasing the truth value of a, b or c will not have a support for that atom. On the other hand, the non-SRCF program $\mathcal{P}_6 = \{a \oplus b \leftarrow \bar{1}, a \leftarrow b, b \leftarrow a, a \leftarrow a \oplus a\}$ has only one answer set, namely $I = \{(a, 1), (b, 1)\}$. One can check that there is no support for each of a and b in \mathcal{P}_6 w.r.t. I , since in this case, $I(a) + I(b) > 1$.

The following lemma holds in both classical and fuzzy ASP.

Lemma 3.5.2 (Locality). *Let \mathcal{P}' be any subset of a program \mathcal{P} . If I is an answer set of \mathcal{P}' and it satisfies all the rules in $\mathcal{P} - \mathcal{P}'$, then I is also an answer set of \mathcal{P} .*

Proof. Since I is an answer set of \mathcal{P}' and I satisfies every rule of $\mathcal{P} - \mathcal{P}'$, I also satisfies every rule in \mathcal{P} . Then clearly, I satisfies \mathcal{P}^I as well. Suppose that I is not the minimal model of \mathcal{P}^I , i.e., that there is another model $J < I$ of \mathcal{P}^I . Since $\mathcal{P}' \subseteq \mathcal{P}$ (and hence $\mathcal{P}'^I \subseteq \mathcal{P}^I$), it must also be the case that J satisfies \mathcal{P}'^I . But this means that I is not the minimal model of \mathcal{P}'^I , contradicting the assumption that I is an answer set of \mathcal{P}' . \square

We now present the main result for this section.

Theorem 3.5.3. *Let $\mathcal{P}_1 = \mathcal{P} \cup \{a \oplus b \leftarrow c\}$ be any SRCF program w.r.t. a, b and c . Then, an interpretation I is an answer set of \mathcal{P}_1 iff it is also an answer set of $\mathcal{P}_2 = \mathcal{P} \cup \{a \leftarrow c \otimes \text{not } b, b \leftarrow c \otimes \text{not } a\}$.*

Proof. (a) “If”-part: Let I be an answer set of \mathcal{P}_2 . Then I is a minimal model of $\mathcal{P}^I \cup \{a \leftarrow c \otimes (1 - I(b)), b \leftarrow c \otimes (1 - I(a))\}$. Clearly, we have $I(a) + I(b) \geq I(c)$. We consider two cases:

- (i) $I(a) + I(b) = I(c)$. Let $p \in \{x \mid I(x) > 0\}$. By Theorem 1, there is a support R_p of p in \mathcal{P}_2 w.r.t I . If $\{a \leftarrow c \otimes (1 - I(b)), b \leftarrow c \otimes (1 - I(a))\} \cap R_p = \emptyset$, then we must have $R_p \subseteq \mathcal{P}^I$. This means that R_p is a support for p in \mathcal{P} w.r.t I . On the other hand, if any (or both) of $\{a \leftarrow c \otimes (1 - I(b)), b \leftarrow c \otimes (1 - I(a))\}$ occurs in R_p , we can replace it (them) with the rule $a \oplus b \leftarrow c$, to obtain the set R'_p which can serve as a support for p in \mathcal{P}_1 w.r.t I . In any case, each support R_p in \mathcal{P}_2 can be replaced with a support for p in \mathcal{P}_1 . By Theorem 1, this means that every answer set of \mathcal{P}_2 is also an answer set of \mathcal{P}_1 .
- (ii) $I(a) + I(b) > I(c)$. In this case, we have that $\{a \leftarrow c \otimes (1 - I(b)), b \leftarrow c \otimes (1 - I(a))\} \not\subseteq R_p$ for any support R_p of any $p \in \{x \mid I(x) > 0\}$, since it does not satisfy the first condition in the definition of support. Therefore, we have that $R_p \subseteq \mathcal{P}$ for any p , which, by Theorem 1 means that I is also an answer set of \mathcal{P} . Since I definitely satisfies $a \oplus b \leftarrow c$, by Lemma 1, I is also an answer set of \mathcal{P}_1 .

(b) “Only if”-part: Similar to the previous part, let I be an answer set of \mathcal{P}_1 . Then I is a minimal model of $\mathcal{P}^I \cup \{a \oplus b \leftarrow c\}$, and also $I(a) + I(b) \geq I(c)$. As before, we consider two cases:

- (i) $I(a) + I(b) = I(c)$. Let $p \in \{x \mid I(x) > 0\}$. By Theorem 1, there is a support R_p of p in \mathcal{P}_2 w.r.t I . If $a \oplus b \leftarrow c \notin R_p$, then we must have $R_p \subseteq \mathcal{P}^I$, which means that R_p is a support for p w.r.t \mathcal{P} . On the other hand, if $a \oplus b \leftarrow c \in R_p$, we can replace it with the two rules $\{a \leftarrow c \otimes (1 - I(b)), b \leftarrow c \otimes (1 - I(a))\}$ to obtain the set R'_p which can serve as a support for p in \mathcal{P}_2 w.r.t I . In any case, each support R_p in \mathcal{P}_1 w.r.t I can be replaced with a support for p in \mathcal{P}_2 w.r.t I . By Theorem 1, this means that every answer set of \mathcal{P}_1 is also an answer set of \mathcal{P}_2 .
- (ii) $I(a) + I(b) > I(c)$. Similar to case (a)(ii), here we have that $a \oplus b \leftarrow c \notin R_p$ for any support R_p of any $p \in \{x \mid I(x) > 0\}$. Again, using Theorem 1, we get that every answer set of \mathcal{P}_1 is also an answer set of \mathcal{P}_2 .

□

This result allows us to reduce an SRCF disjunctive FASP program to an equivalent normal program by performing the shifting operations, thus allowing the use of evaluation methods geared towards normal programs.

Example 5. The program \mathcal{P}_5 with the following rules:

$$\{a \oplus b \leftarrow \bar{1}, c \leftarrow b, c \leftarrow d \oplus e\}$$

is SRCF, since we can assign the stratification function $f(a) = f(b) = f(d) = f(e) = 1$ and $f(c) = 2$. Hence, by Theorem 2, it is equivalent to the normal program:

$$\{a \leftarrow \text{not } b, b \leftarrow \text{not } a, c \leftarrow b, c \leftarrow d \oplus e\}$$

However, program $\mathcal{P}_5 \cup \{d \leftarrow c\}$ is not SRCF, and the shifting method does not work.

As a corollary of Theorem 2, any strict disjunctive FASP program can be reduced to a normal FASP program by shifting.

Example 6. Consider program \mathcal{P}_2 from Example 2. It is a strict disjunctive FASP program, and hence it can be reduced to the equivalent normal program $\{a \leftarrow \text{not } b, b \leftarrow \text{not } a, a \leftarrow b, b \leftarrow a\}$.

3.5.2 Non-SRCF programs

For non-SRCF programs, finding an answer set in \mathcal{L}_∞ requires finding an answer set I in \mathcal{L}_k for some $k \geq 1$, and checking whether I is also an answer set for \mathcal{L}_∞ . We show in this section how the last step can be implemented using Mixed Integer Programming (MIP). For some background on MIP, one can consult, e.g., [27] and [28].

In [29], a representation of infinitely-valued Łukasiewicz logic using MIP was proposed by defining a translation of each of the Łukasiewicz expressions $x \oplus y$, $x \otimes y$ and $\neg x$ into a set of MIP inequality constraints characterizing the value of each of the expressions. Given a FASP program \mathcal{P} and an interpretation I , we can use the MIP representation of \mathcal{P} (denoted as $MIP(\mathcal{P})$) based on the representations proposed by [29] to check whether I is the minimal model of \mathcal{P}^I , as follows:

1. For each atom a in \mathcal{P}^I , we use a MIP variable $v_a \in [0, 1]$ in $MIP(\mathcal{P})$ to express the truth value that a can take.
2. For any expression $e \in \{a \oplus b, a \otimes b, a \vee b, a \bar{\wedge} b\}$ in any rule of \mathcal{P}^I , we create the appropriate set of constraints in MIP to represent the value of the expression, as suggested in [29]. For example, for $a \oplus b$, we have the

following MIP constraints:

$$\begin{aligned}
v_a + v_b + z_{a \oplus b} &\geq v_{a \oplus b} \\
v_a + v_b - z_{a \oplus b} &\leq v_{a \oplus b} \\
v_a + v_b - z_{a \oplus b} &\geq 0 \\
v_a + v_b - z_{a \oplus b} &\leq 1 \\
v_{a \oplus b} &\geq z_{a \oplus b}
\end{aligned}$$

In each case, z_e is a 0-1 variable and v_e is a variable representing the value of the expression e .

3. For each rule $\alpha \leftarrow \beta \in \mathcal{P}^I$, we add the constraint $v_\alpha \geq v_\beta$, where v_α and v_β are the variables corresponding to the values of the atoms/expressions α and β , respectively.
4. For each atom a , we add the constraint $v_a \leq I(a)$.
5. We set the objective function of the MIP program to minimise the value $\sum_{a \in \mathcal{B}_{\mathcal{P}}} v_a$.

Theorem 3.5.4. *The interpretation I is the minimal model of \mathcal{P}^I iff the solution returned in $MIP(\mathcal{P})$ is equal to I .*

3.5.3 Incorporating program decomposition

While in practical applications FASP programs will not always be SRCF, often it will be possible to decompose programs such that many of the resulting components are SRCF. In this section, we show how we can apply the reduction from Section 3.1 to these individual components, and thus efficiently solve the overall program.

Program modularity and decomposition using dependency analysis have been extensively studied and implemented in classical ASP. In [30], the concept of splitting sets for decomposing an ASP program was introduced. Dependency analysis and program decomposition using strongly connected components (SCC) was described in [31] and [32], and has been used as a framework for efficient evaluation of logic programs, such as in [33, 34] and [35]. In this section, we build on this idea to develop a more efficient evaluation framework for FASP programs by exploiting the program's modularity/decomposability.

For a (ground) FASP program \mathcal{P} , consider a directed graph $G_{\mathcal{P}} = \langle V, E \rangle$, called the *dependency graph* of \mathcal{P} , defined as follows: (i) $V = \mathcal{B}_{\mathcal{P}}$ and (ii) $(a, b) \in E$ iff there exists a rule $r \in \mathcal{P}$ s.t. $a \in \text{Lit}(\text{Body}(r))$ and $b \in \text{Lit}(\text{Head}(r))$. By SCC analysis, we can decompose $G_{\mathcal{P}}$ into SCCs C_1, \dots, C_n . With each SCC C_i , we associate a *program component* $PC_i \subseteq \mathcal{P}$, defined as the maximal set of

rules such that for every $r \in PC_i$, the literals in PC_i are contained in C_i . The set of all program components of \mathcal{P} is denoted as $PC(\mathcal{P})$. We define dependency between program components PC_i and PC_j as follows: PC_i depends on PC_j iff there is an atom a in PC_i and atom b in PC_j such that a depends on b in $G_{\mathcal{P}}$. The program component graph $C = \langle PC(\mathcal{P}), E_C \rangle$ is defined according to the dependency relation between the program components.

Similar to the case in classical ASP, the program component graph of a FASP program allows us to decompose the program into “modular components” that can be separately evaluated. For non-disjunctive components, the evaluation method described in [8] can be directly used. For SRCF disjunctive components, we can perform the shifting method as described in Section 3.1 to reduce the component into a normal program, and again use the evaluation method for normal programs. For non-SRCF disjunctive components, an extra minimality check as defined in Section 3.2 is needed after finding a k -answer set. Evaluation proceeds along the program components according to the topological sorting of the components in the program component graph, feeding the “partial answer sets” obtained from one component into the next. If a “complete answer set” is found, we stop. Otherwise, we backtrack to the previous component(s), obtaining k -answer sets for the next values of k until the stopping criterion is met.

Proposition 7. *Label an edge in $G_{\mathcal{P}}$ with the symbol \oplus if the edge corresponds to a rule containing a disjunction in the body. A component is non-SRCF w.r.t. the atoms in that component iff there is a cycle in the component containing a labelled edge.*

Example 7. *Consider the program \mathcal{P}_6 containing the rules:*

$$\{a \leftarrow b \oplus c, b \leftarrow a \otimes \overline{0.5}, c \leftarrow \overline{0.7}, d \oplus e \leftarrow a\}$$

Program \mathcal{P}_6 is not SRCF, hence we cannot directly use Theorem 2 to perform shifting. However, using SCC program decomposition, we obtain three components $PC_1 = \{a \leftarrow b \oplus c, b \leftarrow a \otimes \overline{0.5}\}$, $PC_2 = \{c \leftarrow \overline{0.7}\}$ and $PC_3 = \{d \oplus e \leftarrow a\}$. PC_1 only depends on PC_2 , PC_2 has no dependencies, while PC_3 depends only on PC_1 . Proceeding according to the topological order of the program components, we start by evaluating PC_2 and obtain the partial answer set $\{(c, 0.7)\}$. We feed this partial answer set into the next component PC_1 . This program is normal and hence requires no minimality check associated to disjunctive programs. We obtain the partial answer set $\{(a, 1), (b, 0.5), (c, 0.7)\}$. The last component PC_3 is disjunctive, but it is also SRCF w.r.t. its atoms, and hence we can perform shifting to obtain the normal program $\{d \leftarrow a \otimes \text{not } e, e \leftarrow a \otimes \text{not } d\}$, which again can be evaluated without using minimality checks.

3.6 Experimental benchmark

In this section, we experimentally evaluate the effectiveness of the proposed method. We used `clingo` from the Potassco project [10] as the underlying ASP solver for finding k -answer sets, and the `Coin-OR Cbc`¹ solver as the MIP program solver for minimality checking. The implementation is available online on [8]².

For this benchmark, we used two problems: (1) the fuzzy graph colorability as given in the introduction, and (2) the fuzzy set covering problem, which is a generalization of the classical set covering problem, defined as follows: A fuzzy set F is defined as a function $F : U \rightarrow [0, 1]$, where U is the universe of discourse, and $F(u)$ for $u \in U$ is the degree of membership of u in F . A fuzzy subset S of F is a fuzzy set such that $S(u) \leq F(u), \forall u \in U$. Given a fuzzy set F and a collection of subsets $C = \{S_1, \dots, S_n\}$ of F , the problem asks whether we can find a fuzzy sub-collection of C , such that every member of F is covered *sufficiently* by the subsets selected from C , and that the degree to which a subset S_i is selected is below a given threshold. We encode the problem in FASP as follows: the fuzzy set F is given by a set of facts of the form $f(x) \leftarrow \bar{a}$, the subsets S_i given by facts of the form $subset(s_i)$ and their membership degrees by $member(s_i, x) \leftarrow \bar{b}$. The maximum degree to which a subset S_i can be selected is denoted by a constraint $\bar{c} \leftarrow in(s_i)$. The following FASP program encodes the problem goal and constraints:

$$\begin{aligned} in(S) \oplus out(S) &\leftarrow subset(S) \\ covered(X) &\leftarrow (in(s_1) \otimes member(s_1, X)) \oplus \dots \oplus (in(s_n) \otimes member(s_n, X)) \\ \bar{0} &\leftarrow f(X) \otimes \mathbf{not} covered(X) \end{aligned}$$

For both benchmark problems, instances are generated randomly with no attempt to produce “hard” instances. Constant truth values for fuzzy facts (e.g., for edge weights) are drawn randomly from the set \mathbb{Q}_{10} . Two types of instances are considered: (1) where no saturation rules are present, which means that the program is an SRCF program, and (2) where the saturation rules are added randomly with a 0.1 probability for each $b(x)$ and $w(x)$ atoms (in the graph coloring problem instances) and each $in(x)$ atoms (in the set covering problem instances). Since fuzzy answer set evaluation using finite-valued translation such as the one used in [8] cannot, in principle, be used to prove inconsistency, we opted to generate only instances that are known to be satisfiable.

To be able to see the advantage of applying our approach, we run the solver on all instances both with and without employing SRCF detection and shifting to reduce to normal programs. When SRCF detection is not employed, a minimality check has to be performed to verify that the answer sets obtained in any disjunctive

¹<https://projects.coin-or.org/Cbc>

²<https://github.com/mushthofa/ffasp>

Table 3.1 Values in the cells indicate average execution times (over ten instances) in seconds for the non-timed-out executions. Cells labeled with '(TO)' indicates that all executions of corresponding instances exceeded time/memory limit

problem saturation method	fuzzy graph coloring					fuzzy set cover				
	no		yes			no		yes		
	δ	σ	δ	σ		δ	σ	δ	σ	
1	$n = 20$	2.9	1.7	2.7	1.8	$n = 10$	7.9	5.2	8.2	7.9
2	$n = 30$	6.6	3.8	6.1	3.8	$n = 15$	13.4	6.5	17.3	17.1
3	$n = 40$	10.6	5.7	10.7	6.1	$n = 20$	18.2	9.6	17.4	17.3
4	$n = 50$	19.8	11.5	23.0	11.0	$n = 25$	29.8	13.4	30.1	29.9
5	$n = 60$	34.8	17.7	36.0	20.4	$n = 30$	71.4	17.6	71.4	70.6
6	$n = 70$	53.4	25.4	55.3	28.2	$n = 35$	(TO)	22.3	(TO)	(TO)
7	$n = 80$	74.8	33.9	76.1	41.1	$n = 40$	(TO)	27.8	(TO)	(TO)
δ = no shifting, σ = with shifting applied										

component of the program are indeed minimal. Thus, our experiment will be useful to see the effectiveness of the proposed reduction over the baseline method of computing answer sets and checking for minimality.

The experiment was conducted on a Macbook with OS X version 10.8.5 running on Intel Core i5 2.4 GHz with 4 GB of memory. Execution time for each instance is limited to 2 minutes, while memory usage is limited to 1 GB. Table 3.1 presents the results of the experiment. Each value is an average over ten repeats.

From the result, we can see that when SRCF detection and shifting are used, execution times are generally lower than when only minimality checks are used, even when saturation rules are present. This is especially true for the instances of the fuzzy graph coloring problem. The use of program decomposition/modularity analysis to separate program components that are SRCF from those that are non-SRCF can be beneficial since this means we can isolate the need for minimality checks to only those non-SRCF components, while the rest can be evaluated as normal programs after performing the shifting operation. For the set covering problem, we see no significant improvement in the running time when using the shifting operator for instances with saturation, one of the reasons being that the instances are such that minimality checks are needed regardless of what method is used (due to the fact that most components are non-SRCF). However, for the non-saturated instances, again we still see a clear advantage of using SRCF detection and shifting.

References

- [1] Davy Van Nieuwenborgh, Martine De Cock, and Dirk Vermeir. *Fuzzy answer set programming*. In Proceedings of the 10th European Conference on Logics in Artificial Intelligence, pages 359–372. 2006.
- [2] Marjon Blondeel, Steven Schockaert, Dirk Vermeir, and Martine De Cock. *Fuzzy answer set programming: An introduction*. In Soft Computing: State of the Art Theory and Novel Applications, pages 209–222. Springer, 2013.
- [3] Chitta Baral. *Knowledge representation, reasoning and declarative problem solving*. Cambridge University Press, 2003.
- [4] Vladimir Lifschitz. *What Is Answer Set Programming?*. In AAAI, volume 8, pages 1594–1597, 2008.
- [5] *Special issue on Answer Set Programming*. AI Magazine, 37(3), 2016.
- [6] Mario Alviano and Rafael Peñaloza. *Fuzzy answer sets approximations*. Theory and Practice of Logic Programming, 13(4-5):753–767, 2013.
- [7] Mario Alviano and Rafael Peñaloza. *Fuzzy answer set computation via satisfiability modulo theories*. Theory and Practice of Logic Programming, 15:588–603, 7 2015.
- [8] Mushthofa Mushthofa, Steven Schockaert, and Martine De Cock. *A finite-valued solver for disjunctive fuzzy answer set programs*. In Proceedings of European Conference in Artificial Intelligence 2014, pages 645–650, 2014.
- [9] Mushthofa Mushthofa, Steven Schockaert, and Martine De Cock. *Solving Disjunctive Fuzzy Answer Set Programs*. In Proceedings of the 13th International Conference on Logic Programming and Non-monotonic Reasoning, pages 453–466, 2015.
- [10] Martin Gebser, Benjamin Kaufmann, Roland Kaminski, Max Ostrowski, Torsten Schaub, and Marius Schneider. *Potassco: The Potsdam answer set solving collection*. AI Communications, 24(2):107–124, 2011.
- [11] Nicola Leone, Gerald Pfeifer, Wolfgang Faber, Thomas Eiter, Georg Gottlob, Simona Perri, and Francesco Scarcello. *The DLV system for knowledge representation and reasoning*. ACM Transactions on Computational Logic (TOCL), 7(3):499–562, 2006.
- [12] James P Delgrande, Torsten Grote, and Aaron Hunter. *A general approach to the verification of cryptographic protocols using answer set programming*. In Proceedings of the 10th International Conference in Logic Programming and Nonmonotonic Reasoning (LPNMR 2009), pages 355–367, 2009.

- [13] Esra Erdem, Vladimir Lifschitz, and Martin Wong. *Wire routing and satisfiability planning*. Computational LogicCL 2000, pages 822–836, 2000.
- [14] Giovanni Grasso, Nicola Leone, Marco Manna, and Francesco Ricca. *ASP at Work: Spin-off and Applications of the DLV System*. Logic programming, knowledge representation, and nonmonotonic reasoning, 6565:432–451, 2011.
- [15] Martin Gebser, Arne König, Torsten Schaub, Sven Thiele, and Philippe Veber. *The BioASP library: ASP solutions for systems biology*. In Tools with Artificial Intelligence (ICTAI), 2010 22nd IEEE International Conference on, volume 1, pages 383–389. IEEE, 2010.
- [16] Mushthofa Mushthofa, Gustavo Torres, Yves Van de Peer, Kathleen Marchal, and Martine De Cock. *ASP-G: an ASP-based method for finding attractors in genetic regulatory networks*. Bioinformatics, 30(21):3086, 2014.
- [17] Steve Dworschak, Susanne Grell, Victoria J Nikiforova, Torsten Schaub, and Joachim Selbig. *Modeling biological networks by action languages via answer set programming*. Constraints, 13(1-2):21–65, 2008.
- [18] Esra Erdem. *Theory and Applications of Answer Set Programming*. PhD thesis, 2002. The University of Texas at Austin.
- [19] Marjon Blondeel, Steven Schockaert, Dirk Vermeir, and Martine De Cock. *Complexity of fuzzy answer set programming under Łukasiewicz semantics*. International Journal of Approximate Reasoning, 55(9):1971–2003, 2014.
- [20] Marjon Blondeel, Steven Schockaert, Dirk Vermeir, and Martine De Cock. *Fuzzy Answer Set Programming: An Introduction*. In Ronald R. Yager, Ali M. Abbasov, Marek Z. Reformat, and Shahnaz N. Shahbazova, editors, Soft Computing: State of the Art Theory and Novel Applications, volume 291 of *Studies in Fuzziness and Soft Computing*, pages 209–222. Springer Berlin Heidelberg, 2013.
- [21] Carlos Viegas Damásio and Luís Moniz Pereira. *Antitonic Logic Programs*. In Proceedings of the 6th International Conference on Logic Programming and Nonmonotonic Reasoning, pages 379–392. 2001.
- [22] Jeroen Janssen, Steven Schockaert, Dirk Vermeir, and Martine De Cock. *General fuzzy answer set programs*. In Fuzzy Logic and Applications, pages 352–359. Springer, 2009.
- [23] Stefano Aguzzoli and Agata Ciabattoni. *Finiteness in infinite-valued Łukasiewicz logic*. Journal of Logic, Language and Information, 9(1):5–29, 2000.

- [24] Steven Schockaert, Jeroen Janssen, and Dirk Vermeir. *Satisfiability Checking in Łukasiewicz Logic as Finite Constraint Satisfaction*. Journal of Automated Reasoning, 49(4):493–550, 2012.
- [25] Rachel Ben-Eliyahu and Rina Dechter. *Propositional semantics for disjunctive logic programs*. Annals of Mathematics and Artificial intelligence, 12(1-2):53–87, 1994.
- [26] Jürgen Dix, Georg Gottlob, and Wiktor Marek. *Reducing disjunctive to non-disjunctive semantics by shift-operations*. Fundamenta Informaticae, 28(1):87–100, 1996.
- [27] Robert G Jeroslow. *Logic-based decision support: mixed integer model formulation*. Elsevier, 1989.
- [28] Alexander Schrijver. *Theory of linear and integer programming*. John Wiley & Sons, 1998.
- [29] R. Hähnle. *Proof theory of many-valued logic-linear optimization-logic design: connections and interactions*. Soft Computing, 1(3):107–119, 1997.
- [30] Vladimir Lifschitz and Hudson Turner. *Splitting a logic program*. In Proceedings of the 11th International Conference on Logic Programming, pages 23–37, 1994.
- [31] Kenneth A Ross. *Modular stratification and magic sets for Datalog programs with negation*. In Proceedings of the 9th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, pages 161–171, 1990.
- [32] Thomas Eiter, Georg Gottlob, and Heikki Mannila. *Disjunctive datalog*. ACM Transactions on Database Systems, 22(3):364–418, 1997.
- [33] Emilia Oikarinen and Tomi Janhunen. *Achieving Compositionality of the Stable Model Semantics for Smodels Programs*. Theory and Practice of Logic Programming, 8(5-6):717–761, 2008.
- [34] Thomas Eiter, Giovambattista Ianni, Roman Schindlauer, and Hans Tompits. *DLVHEX: A prover for semantic-web reasoning under the answer-set semantics*. In Proceedings of IEEE/WIC/ACM International Conference on Web Intelligence, 2006, pages 1073–1074, 2006.
- [35] Thomas Eiter, Wolfgang Faber, and Mushthofa Mushthofa. *Space Efficient Evaluation of ASP Programs with Bounded Predicate Arities*. In 24th AAAI Conference on Artificial Intelligence, pages 303–308, 2010.

4

Modelling multi-valued networks using FASP

4.1 Introduction

In biological systems, genes are known to interact with each other in a complex and dynamic way. Briefly, each gene's activation state can influence the activation states of other genes, either positively or negatively. These interactions can be modelled using a graph structure, which is usually called a Gene Regulatory Network (GRN). It determines the patterns of activation states of the genes, which in turn affects the phenotypic behavior of the system.

As explained in Chapter 2, one of the most important concepts in modelling the dynamics of GRNs are the so-called *attractors*, which are the sets of states to which the system converges. An attractor usually corresponds to the observed characteristics/phenotypes of the biological system [1]. For example, the attractors of a GRN usually correspond to the expression patterns of the genes in the network for specific types of cells [2, 3]. In studying the dynamics of such networks it is therefore of importance to be able to identify their attractors.

In systems biology, one of the most popular approaches to formalise a GRN is to use a so-called Boolean Network (BN) [4–6]. Boolean networks represent genes as nodes that can take on Boolean values (intuitively representing the activation levels of the genes), while interactions between the genes are represented as Boolean functions that determine the value of each node at a certain time, depending on the current values of the other genes. The state transitions of a GRN

and their attractors can be readily represented using such a formalism.

There have been numerous works about computational tools to simulate the dynamics of Boolean networks and to compute their attractors, mostly using logic-based techniques such as Binary Decision Diagrams (BDDs) or Boolean SAT solvers [7–12]. More recently, Answer Set Programming (ASP) has become a particularly interesting framework for modelling GRNs and Boolean networks [13–16].

ASP is a popular declarative programming paradigm which allows for an easy and intuitive encoding of many combinatorial search and optimisation problems [17, 18]. The availability of fast and efficient solvers for ASP, such as `clasp` [19] and `DLV` [20], allows for the application of ASP in various fields [21, 22]. Despite its flexibility and expressive power, however, ASP lacks the ability to directly encode problems in continuous domains.

Having only two levels of activation is sometimes not always enough to fully understand the dynamics of real biological systems. For example, in [3, 23–26], examples of systems are given whose dynamics can only be modelled by considering more than two activation levels. One classic example is the *lac operon* regulatory system, which is a set of genes that controls the production of the proteins needed to metabolise lactose in enteric bacterias, such as *Escherichia coli* (see e.g., [27]). In this case, it has been shown that one of the key attractors cannot be characterized using a Boolean encoding (because of the so-called “leaky-expression”). Despite the importance of multi-valued activation levels for modelling gene regulatory networks, only limited progress has been made on developing simulation tools that can support them. To the best of our knowledge, only one tool has been developed that supports multi-valued activation levels [24].

In this chapter, we describe our proposed method on the use of Fuzzy Answer Set Programming (FASP) [28] as a computational framework to simulate the dynamics of multi-valued regulatory networks. As explained in Chapter 3, FASP is a form of declarative programming that extends ASP by allowing graded truth values in atomic propositions and using fuzzy logic connectives to aggregate these truth values. Recent work on the implementation of a FASP solver, such as [29–33], has opened the door to the application of FASP for solving real-world applications. Other frameworks dealing with the extension of ASP, or more generally, logic programming into the fuzzy domains have been proposed in the literature, e.g., [34–39]. While we have specifically chosen to use the FASP framework and the corresponding solver from [32], other multi-valued extensions of ASP might also be suitable for the purpose of modelling the dynamics of multi-valued regulatory networks.

Here, we propose an encoding of the dynamics of multi-valued biological interaction networks that can be executed/solved using the FASP solver proposed in [32], and we prove the correctness of this encoding. We then perform an exten-

sive benchmark test using synthetic networks as well as real biological networks found in the literature to show the efficiency and applicability of this method. The results indicate that the method is efficient for the size of the networks typically used in the Boolean/discrete modelling of regulatory networks (up to around a few dozen genes in the network).

The remainder of this chapter is structured as follows: We first give a brief overview of related works in Section 4.2. We then formally define the multi-valued networks and present our FASP-based encoding in Section 4.3. In Subsection 4.3.1, we describe a general fuzzy logic framework for encoding general aspects of multi-valued networks. Subsequently, we describe the specific encoding needed to represent the dynamics and to compute the steady states and attractors of the network in Subsection 4.3.2 and 4.3.3. Section 4.4 describes the FASPG tool that implements the proposed method, as well as providing an automatic encoding for the network. Section 4.5 contains the experiments we conducted to test the feasibility and efficiency of the proposed method, while Section 4.6 provides a conclusion.

4.2 Related Work

Since they were introduced by Kauffman [4], Boolean networks have gained considerable popularity as a simple but powerful modelling technique in systems biology. Boolean networks have been used to describe the dynamics of regulatory networks in cases where we have reasonably good knowledge about the regulatory relationship between the genes, and where the activation levels of genes can be simply represented as “on” and “off”. In such cases, the dynamics of the network, and especially the attractors, usually correspond to some biologically relevant phenotype, e.g. a cell type. For instance, in [40, 41] and more recently, [3] and [42], the flower development in *Arabidopsis thaliana* was modeled using a Boolean network, in which the network attractors corresponded to stable gene expression levels during the different stages of flower development. In [43], Li et al. used a Boolean network model and its steady states to describe the different stages of the yeast cell cycle, where the stages of the cycle correspond to the strong attractors of the network. Kaufman et al. [5] explained the various states of the immune system with Boolean network models. Similarly, the regulatory networks involved in the various parts of the development of *Drosophila melanogaster* were studied using Boolean networks in [44], [45] and [46].

Although Boolean networks provide a useful simplification to study the dynamics of gene regulatory networks, using only two values to represent the activation may cause one to miss important characterizations of GRNs that have attractors containing “intermediate” levels of expressions of the genes (see e.g., [3, 23–25, 47]). In [24], an extension of Boolean networks into multi-valued net-

works in which each node is allowed to have k levels of activation (where $k \geq 2$) is considered. Using the so-called 1-hot encoding, these multi-valued networks are reduced into a representation which allows techniques already used in Boolean networks, such as Binary Decision Diagrams (BDD), to be applied. However, the use of an encoding scheme such as 1-hot encoding can make the representation quite cumbersome, especially for large values of k , since it requires us to explicitly encode the logical operators for all combinations of truth values. As we will show, the use of FASP can overcome this problem by using fuzzy logic connectives.

Several computational tools have been developed to compute attractors in Boolean network models. In [7], Garg et al. developed **genYsis**, which uses techniques involving BDDs to compute attractors. Ay et al. [10] used state-space pruning and randomized state-space traversal methods to improve the scalability of the attractor computation. Dubrova et al. [11] used a Boolean Satisfiability (SAT) solver, which was shown to be more efficient, both in terms of computation time and space requirements, compared to the BDD-based approach. Zheng et al. [12] developed **geneFatt** based on the Reduced Order BDD (ROBDD) data structure, which further improves the efficiency of the attractor computation. Berntenis et al. [9] considered the enumeration of attractors of larger networks by restricting the enumeration of possible states to only the relevant subsets. More recently, [14] used Answer Set Programming (ASP) to model the computation of attractors in a Boolean network. However, these methods were designed to compute the dynamics of Boolean networks, i.e., where the nodes can take only two possible values. In this Section, we describe our extension to our previous work in [14], by using Fuzzy Answer Set Programming (FASP) to allow the computation of the dynamics of multi-valued networks.

4.3 Multi-valued Networks

4.3.1 Modelling multi-valued networks using FASP

Models of multi-valued biological interaction networks are typically specified through a set of input-output relationships for each node, detailing the values each node takes, given the combinations of the values of the regulators, i.e. nodes that affect it. We formalize this idea, using the concept of a multi-valued network defined as follows.

Definition 2 (Multi-valued network, network state). *A multi-valued network is a tuple $G = \langle X, F, k \rangle$ where $X = \langle x_1, \dots, x_n \rangle$ is a tuple of multi-valued variables denoting the nodes of the network, $F = \langle f_1, \dots, f_n \rangle$ is a tuple of update functions, and $k \geq 1$ is a parameter describing the number of activation levels for all the*

nodes. Specifically, for each node $x \in X^1$, we allow $k + 1$ activation levels, i.e., the value for each x is taken from the set $\mathbb{Q}_k = \{0, \frac{1}{k}, \dots, \frac{k-1}{k}, 1\}$. A network state is then defined as an assignment $V : X \rightarrow \mathbb{Q}_k$. Furthermore, each function $f_i \in F$ satisfies $f_i : \mathbb{Q}_k^n \rightarrow \mathbb{Q}_k$ and is defined using the Łukasiewicz connectives $\otimes, \oplus, \vee, \wedge$, and \neg , instead of the Boolean connectives.

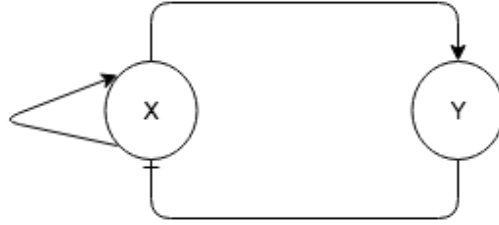
From this, we naturally extend the definitions of state transition, update scheme as well as attractor. Note that the definition of the Hamming distance function Δ in (1) can also be applied to the multi-valued network states.

Definition 3 (State transition). *The tuple F of functions defines the state mapping function $f : S \rightarrow S$ as follows: the state $f(v)$ for a state v is the state $w \equiv \langle f_1(v), \dots, f_n(v) \rangle$. The state transition of a multi-valued network is a relation $\hookrightarrow : S \rightarrow S$ whose definition is determined by the type of the update scheme that the network has. The notion of synchronous and asynchronous update scheme in multi-valued networks is defined similarly to the one in Boolean networks.*

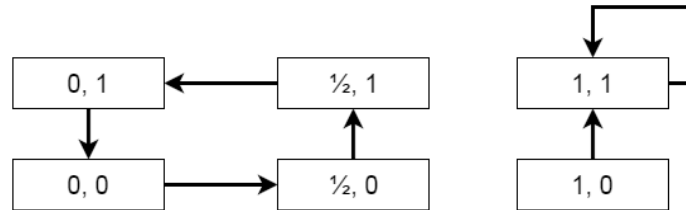
In the literature (e.g., [3, 48]), the values each node can take are usually given as integers, ranging from $0, 1, \dots, k$. Due to the fact that our model is expressed in fuzzy logic, we need to map these values into the $[0, 1]$ range, which can simply be done by mapping each value v to $\frac{v}{k}$. Furthermore, the ranges of possible values often differ from node to node (e.g., the network in [48] has one node with two levels, and one node with three levels). For such cases, we choose k based on the node with the largest range of values, and we map the values of any node having $l < k$ possible values into the values of an l -sized subset of \mathbb{Q}_k (while preserving order), as illustrated in Example 5. This does not affect the behaviour of the modeled system. In fact, in real biological networks encountered in the literature, we mostly see the situation where some nodes have exactly k levels, whereas the rest have only two possible values. In such a case, we can map the values of the two-valued nodes to the set $\{0, 1\}$.

Example 8. *As a running example, we take the network describing the production of mucus in *Pseudomonas aeruginosa* described in [48]. There are two nodes in the network, namely x and y , with x having three possible values: 0, 1 or 2, and y having only two values: 0 or 1. Therefore, to model the network in our fuzzy logical representation, we set $k = 2$, and map the values of x into $\{0, \frac{1}{2}, 1\}$, while keeping the values of y as they are. The node x is negatively-regulated by node y and positively by itself, while y is positively-regulated by x . The network structure is shown in Figure 4.1. The input-output relationships between the two nodes, as given in [48], are shown in Table 4.1. Based on the regulatory relationships between the nodes, the state transition graph of this network is as shown in Figure 4.2. From the state transition graph, we can clearly see that the network*

¹When it is more convenient, we will abuse the notation for X and treat it as a set.

Figure 4.1 Diagram for the network of *P. aeruginosa***Table 4.1** Regulatory relationship in the *P. aeruginosa* mucus development network

No.	$x(t)$	$y(t)$	$x(t+1)$	$y(t+1)$
1	0	0	$\frac{1}{2}$	0
2	0	1	0	0
3	$\frac{1}{2}$	0	$\frac{1}{2}$	1
4	$\frac{1}{2}$	1	0	1
5	1	0	1	1
6	1	1	1	1

Figure 4.2 State transition graph for the network of *P. aeruginosa* using the synchronous update

has two attractors: one is a steady state, namely $\langle 1, 1 \rangle$, and the other is a cyclic attractor of size 4.

Below, we extend the idea of using ASP to model the dynamics of biological networks as used in [49] and [14] by allowing a multi-valued activation level in each node. However, instead of using ASP in a *meta-level* approach to describe the dynamics of the network, as in [14, 15], we propose to directly encode the interaction between nodes using FASP rules, which allows for a simpler and more efficient implementation. As shown in [16], a direct encoding of the interaction between nodes in a Boolean network is enough to characterize fixed-size attractors. The same holds for multi-valued networks with FASP under an appropriate many-

valued logic semantics.

4.3.2 Finding steady states

We first tackle the problem of finding the single state attractors – also called steady states – of a multi-valued network. Recall that the steady states are identical for the synchronous and asynchronous update schemes.

Let $G = \langle X, F, k \rangle$ be a multi-valued network. First, for every node $x \in X$ in the network, we consider two fuzzy propositional atoms p_x and n_x , and write the following FASP rules:

$$\begin{aligned} p_x \oplus n_x &\leftarrow \bar{1} \\ \bar{0} &\leftarrow p_x \otimes n_x \end{aligned}$$

Intuitively, these two rules generate “guesses” for the values of p_x and n_x such that $p_x + n_x = 1$. Define $GS(G)$ as the set of all such rules. If x is a node that only takes Boolean values, we can add the following rule (usually called the *saturation rule*):

$$p_x \leftarrow p_x \oplus p_x$$

This rule forces the atom p_x to take only Boolean values in any answer set of the program.

We then encode the interaction between nodes by creating a rule for every node x_i , where the head of the rule is a propositional atom p'_{x_i} associated with the node, while the body corresponds to the direct translation of the fuzzy logic function for the update rule of x_i , replacing the occurrences of the negation symbol \neg with FASP’s default negation **not**. Formally, let f_i be the update function of a node x_i of the network. The corresponding FASP *node update rule* of that node, denoted by $NU(f_i)$ is a FASP rule defined as follows:

$$p'_{x_i} \leftarrow BU(f_i)$$

where $BU(f_i)$ is the *body of the node update rule*, which is a FASP expression defined recursively as follows:

- $BU(f_i) = val$ if $f_i \equiv val$ and $val \in [0, 1]$
- $BU(f_i) = p_{x_i}$ if $f_i \equiv x_i$ for a node x_i
- $BU(f_i) = BU(exp_1) \circ BU(exp_2)$ if $f_i \equiv exp_1 \circ exp_2$ for some expressions exp_1, exp_2 and $\circ \in \{\oplus, \otimes, \vee, \wedge\}$
- $BU(f_i) = \mathbf{not} p_x$ if $f_i \equiv \neg x$

Define $NU(G)$ as the set of rules created in this step, i.e., $NU(G) = \{NU(f_i) \mid 1 \leq i \leq n\}$. Intuitively, the atom p'_{x_i} holds the activation value of the node x_i after the update function has been applied. To drive the FASP program to find a steady-state, we enforce the condition that the activation level of each node is the same after the update. This can be done by using the following rules $CS(i)$ for each node x_i

$$\begin{aligned}\bar{0} &\leftarrow p_{x_i} \otimes \mathbf{not} p'_{x_i} \\ \bar{0} &\leftarrow p'_{x_i} \otimes \mathbf{not} p_{x_i}\end{aligned}$$

Define $CS(G)$ as the set of all *constraining* rules, i.e. $CS(G) = \{CS(i) \mid 1 \leq i \leq n\}$. The example below illustrates the construction process of the FASP program $P(G) = GU(G) \cup NU(G) \cup CS(G)$ for the multi-valued network introduced in Example 8.

Example 9. Consider the network of *P. aeruginosa* given in Example 8. Since the network consists of two nodes, x and y , the initial guessing rules for the nodes' values can be written as

$$\begin{aligned}x \oplus n_x &\leftarrow \bar{1} \\ \bar{0} &\leftarrow x \otimes n_x \\ x \oplus n_y &\leftarrow \bar{1} \\ \bar{0} &\leftarrow y \otimes n_y\end{aligned}$$

Since we need y to be Boolean, we add the following rule:

$$y \leftarrow y \oplus y$$

The regulatory relationships between the nodes x and y in the network (as given by Table 4.1) can be captured by the following update functions expressed in Łukasiewicz formulas:

$$\begin{aligned}f_1(x, y) &= ((x \vee \tfrac{1}{2}) \otimes \neg y) \oplus z \\ z &= (x \otimes \tfrac{1}{2}) \oplus (x \otimes \tfrac{1}{2}) \\ f_2(x, y) &= x \oplus x\end{aligned}$$

where z is an auxiliary variable.² We thus construct the following FASP rules to represent the update on each node.

$$\begin{aligned}x' &\leftarrow ((x \vee \tfrac{1}{2}) \otimes \mathbf{not} y) \oplus z \\ z &\leftarrow (x \otimes \tfrac{1}{2}) \oplus (x \otimes \tfrac{1}{2}) \\ y' &\leftarrow x \oplus x\end{aligned}$$

²The variable z is an auxiliary variable only intended to allow us to present a more concise expression here.

Finally, we add the following constraints to find only steady-states:

$$\begin{aligned}\bar{0} &\leftarrow x' \otimes \text{not } x \\ \bar{0} &\leftarrow x \otimes \text{not } x' \\ \bar{0} &\leftarrow y' \otimes \text{not } y \\ \bar{0} &\leftarrow y \otimes \text{not } y'\end{aligned}$$

It can be verified that the resulting program has exactly one 2-answer set, which contains $\{(x, 1), (y, 1)\}$, corresponding to the only steady state $\langle 1, 1 \rangle$ of the network.

The previous example also illustrates the fact that we need to translate the regulatory relationships between multi-valued activation levels into FASP rules. In practice, it may not always be easy to perform this translation manually. As we will explain in Section 4.4, in practice this step can be performed automatically using the tool we wrote.

Next we show that the correspondence between steady states of the multi-valued network G and k -answer sets of the FASP program $P(G) = GU(G) \cup NU(G) \cup CS(G)$ holds in general.

Proposition 1. *The program $P(G) = GU(G) \cup NU(G) \cup CS(G)$ captures all the steady states of the multi-valued network G , i.e., for every k -answer set I of $P(G)$, the state S s.t. $S(x) = I(p_x)$ for every $x \in X$ is a steady state of G , and for every steady state S of G , there is a corresponding k -answer set I of G s.t. $S(x) = I(p_x)$ for every $x \in X$.*

Proof. First, it can be easily seen that in any answer set I of $P(G)$, we have that $I(p'_x) = I(p_x)$, due the rules in $CS(G)$. Suppose that S is a steady-state of the multi-valued network G . By definition, we have that $f_i(X) = S(x_i)$ for every $x_i \in X$. We will show that the interpretation I s.t. $I(p_x) = S(x)$ and $I(n_x) = 1 - S(x)$ for every $x \in X$ is a k -answer set of the program $P(G)$. First, by the definition of $GU(G)$, it is clear that I is a model of $GU(G)$. For every rule r in $NU(G)$ corresponding to the update function f_i , from the fact that $I(p_x) = I(p'_x) = S(x)$ for every $x \in X$, it can be shown that the recursive definition of $BU(f_i)$ entails that $I(\text{Body}(r)) = f_i(X)$. Since we have $f_i(X) = S(x_i)$, we also have that $I(\text{Body}(r)) = S(x_i)$. This means that $I(\text{Head}(r)) = I(p'_{x_i}) = S(x_i) = I(\text{Body}(r))$, which means that I is also a model of the rule r . Consequently, I is a model of $NU(G)$, and thus also of $P(G) = GU(G) \cup NU(G)$. It is easy to see that I is a minimal k -model of $GU(G)$, since any k -model $J < I$ will violate at least one rule in $GU(G)$.

Conversely, if we have a k -answer set I of $P(G)$, we can show that the state S s.t. $S(x) = I(p_x)$ for every $x \in X$ is a steady state of the network. It is sufficient to show that $f_i(X) = S(x_i) = I(p_x) = I(p'_x)$ for every $x_i \in X$. Since

I is a model of the rule $NU(f_i)$, we have that $I(p'_{x_i}) \geq I(\text{Body}(NU(f_i))) = I(BU(f_i))$. From the definition of $BU(f_i)$ it can be shown that $I(BU(f_i)) = f_i(X)$. Hence we have that $I(p'_{x_i}) \geq f_i(X)$. Suppose that $I(p'_{x_i}) > f_i(X)$, for some $x_i \in X$. Consider the k -interpretation J such that $J(p'_a) = I(p'_a)$ for every $a \in X$ s.t. $a \neq x_i$, and $J(p'_{x_i}) = f_i(X)$. We have that $J < I$, and it can be seen that J is also a k -model of $P(G)$ (since it satisfies all the rules in $P(G)$), contradicting the minimality of I . Hence, we must have that $I(p'_{x_i}) = f_i(X)$ for every $x_i \in X$. \square

4.3.3 Finding fixed-size cyclic attractors

It is clear that the approach from Section 4.3.2 is not suitable for finding cyclic attractors, since the proposed encoding does not represent different values of each node at different update times. Recall that we can have either the synchronous or the asynchronous update schemes for our networks, and that using different update schemes on the same network can result in different sets of attractors. We need to explicitly take into account the time dimension to distinguish between different update schemes, and thus compute the appropriate sets of attractors.

Taking into account the time dimension can be achieved by adding a parameter t , representing time, to each of the fuzzy propositional atoms p_x and n_x . This time parameter can be limited up to a certain maximum value, say s , if we are interested in only finding cyclic attractors of size up to s . This can be done simply by adding facts that assert the truth of a predicate called $time(t)$ for $t = 0, 1, \dots, s$.

The initial guessing rules $GU_0(G)$ are now written as

$$\begin{aligned} p_x(0) \oplus n_x(0) &\leftarrow \bar{1} \\ \bar{0} &\leftarrow p_x(0) \otimes n_x(0) \end{aligned}$$

where the parameter 0 encodes the fact that we are guessing at the initial time point $t = 0$. We then define a new encoding of the node update rule that incorporates a time parameter t . In order to do this, we first introduce the so called *time-dependent body of a node update rule*, defined as follows:

- $TBU(f_i, t) = val$ if $f_i(x_i) \equiv val$ and $val \in [0, 1]$
- $TBU(f_i, t) = p_x(t)$ if $f_i(x_i) \equiv x$ for a node x
- $TBU(f_i, t) = TBU(exp_1, t) \circ TBU(exp_2, t)$ if $f_i(x_i) \equiv exp_1 \circ exp_2$ for some expressions exp_1, exp_2 and $\circ \in \{\oplus, \otimes, \vee, \wedge\}$
- $TBU(f_i, t) = \text{not } p_x(t)$ if $f_i(x_i) \equiv \neg x$

We then define the *time-dependent node update rules* TNU , that perform the update to the values in each node, as follows:

$$p_{x_i}(t+1) \leftarrow time(t) \otimes TBU(f_i, t)$$

For the synchronous case, this is enough to encode the fact that at each time step t , each node's value is updated using the update function defined for the node.

For the asynchronous update scheme, recall that even though a state can have multiple successor states (due to the non-deterministic choice of which node is updated), only states that have a single possible successor can be part of an attractor. Thus, at any time step, we need to ensure that there is only one possible successor state of the current state. This can be done by checking that there is exactly one node that gets a new value during the updates, since if no nodes get a new value, then the state would be a steady state, while if more than one node gets updated, then there will be multiple successors to the current state.

This can be done by first adding the following set of rules for each node $x \in X$:

$$\begin{aligned} d_x &\leftarrow p_x(t+1) \otimes \mathbf{not} p_x(t) \\ d_x &\leftarrow p_x(t) \otimes \mathbf{not} p_x(t+1) \\ d_x &\leftarrow d_x \oplus d_x \end{aligned}$$

which intuitively derives the atom d_x if the node x gets a new value during the update. We then add a constraint

$$\bar{0} \leftarrow d_x \otimes d_y$$

for every pair of nodes x and y . This forces that there is at most one node having a new value during the update. Finally, using

$$\begin{aligned} at_least_one &\leftarrow (d_{x_1} \vee \dots \vee d_{x_n}) \\ \bar{0} &\leftarrow \mathbf{not} at_least_one \end{aligned}$$

ensures that there is exactly one node that receives a new value during the update.

We can now define the required condition to find cyclic attractors, independent of the update scheme. The following set of rules and constraints can be used to find cyclic attractors up to size s . First, add the following rules for all k , $1 \leq k \leq s$ and all $x_i \in X$:

$$\begin{aligned} a_k &\leftarrow p_{x_i}(0) \otimes \mathbf{not} p_{x_i}(k) \\ a_k &\leftarrow p_{x_i}(k) \otimes \mathbf{not} p_{x_i}(0) \end{aligned}$$

These rules ensure that a_k is false iff the value of $p_x(0)$ equals to $p_x(k)$ for all $x \in X$, which means that there is a cyclic attractor of size k (or of size an integer divisor of k). Then add the following constraint:

$$\bar{0} \leftarrow a_1 \bar{\wedge} a_2 \dots \bar{\wedge} a_s$$

which forces at least one of the a_k 's to be false, say a_l , which means that there is a cyclic attractor of size l (or a divisor of l). The example below illustrates the FASP program construction process for the network from Example 8.

Example 10. Consider again the network in Example 8, and consider the task of finding the cyclic attractors of size 4 under the synchronous update. Denote this network as G . The initial guessing rules $GU_0(G)$ are:

$$\begin{aligned} x(0) \oplus n_x(0) &\leftarrow \bar{1} \\ y(0) \oplus n_y(0) &\leftarrow \bar{1} \\ \bar{0} &\leftarrow x(0) \otimes n_x(0) \\ \bar{0} &\leftarrow y(0) \otimes n_y(0) \end{aligned}$$

Furthermore, since we need to allow node y to be 0 or 1 only, we add a constraint:

$$y(T) \leftarrow y(T) \oplus y(T)$$

The node updates $TNU(G)$ can be represented using the following rules

$$\begin{aligned} x(T+1) &\leftarrow time(T) \otimes ((x(T) \vee \tfrac{1}{2}) \otimes \text{not } y(T)) \oplus z(T) \\ z(T) &\leftarrow (x(T) \otimes \tfrac{1}{2}) \oplus (x(T) \otimes \tfrac{1}{2}) \\ y(T+1) &\leftarrow time(T) \otimes (x(T) \oplus x(T)) \end{aligned}$$

To find synchronous cyclic attractors up to size 4, we add the following for all $i = 1, \dots, 4$:

$$\begin{aligned} a_i &\leftarrow x(0) \otimes \text{not } x(i) \\ a_i &\leftarrow x(i) \otimes \text{not } x(0) \\ a_i &\leftarrow y(0) \otimes \text{not } y(i) \\ a_i &\leftarrow y(i) \otimes \text{not } y(0) \\ \bar{0} &\leftarrow a_1 \bar{\wedge} a_2 \bar{\wedge} a_3 \bar{\wedge} a_4 \end{aligned}$$

One can check that the resulting program has exactly five 2-answer sets. One of these answer sets encodes the static transitions of the steady-state $\langle 1, 1 \rangle$, by having the same values for $x(0), \dots, x(4)$ and $y(0), \dots, y(4)$. The other four answer sets encode the cyclic attractor $\langle 0, 0 \rangle \hookrightarrow \langle \frac{1}{2}, 0 \rangle \hookrightarrow \langle \frac{1}{2}, 1 \rangle \hookrightarrow \langle 0, 1 \rangle \hookrightarrow \langle 0, 0 \rangle$, with each answer set encoding the different initial conditions.

Recall that the example network does not have any cyclic attractor of size > 1 (as explained in Figure 3) for the asynchronous update. In this case, we need to

add the following rules and constraints:

$$\begin{aligned}
 d_x &\leftarrow p_x(t+1) \otimes \text{not } p_x(t) \\
 d_x &\leftarrow p_x(t) \otimes \text{not } p_x(t+1) \\
 d_x &\leftarrow d_x \oplus d_x \\
 d_y &\leftarrow p_y(t+1) \otimes \text{not } p_y(t) \\
 d_y &\leftarrow p_y(t) \otimes \text{not } p_y(t+1) \\
 d_y &\leftarrow d_y \oplus d_y \\
 \bar{0} &\leftarrow d_x \otimes d_y \\
 at_least_one &\leftarrow d_x \vee d_y \\
 \bar{0} &\leftarrow \text{not } at_least_one
 \end{aligned}$$

We can see that the states $\langle 0, 0 \rangle$, $\langle 0, 1 \rangle$, and $\langle 1, 0 \rangle$ will be eliminated from the search immediately, since they have multiple successor states, as shown in Figure 3.

4.4 Automatic encoding of network descriptions

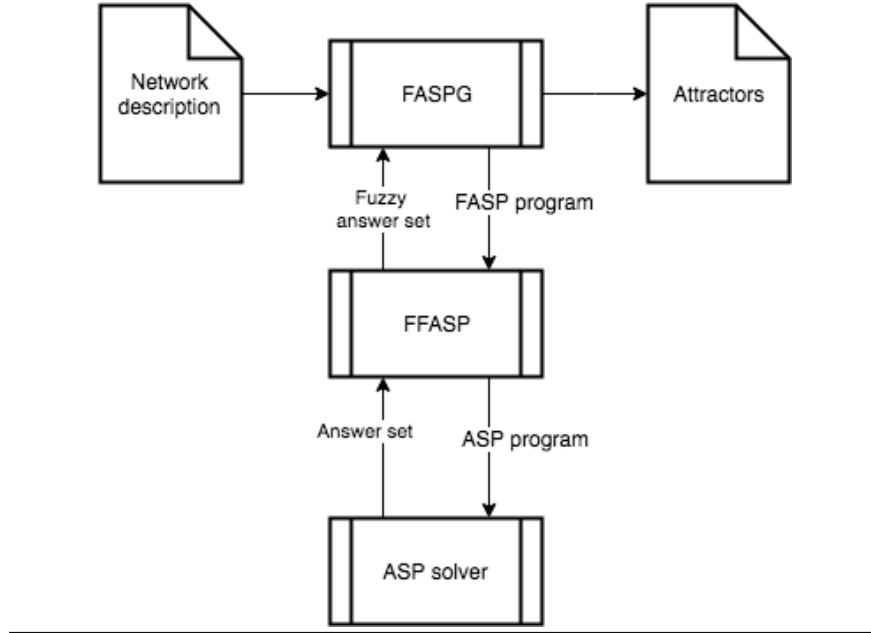
Biological networks with multiple activation levels are often specified in terms of the regulatory relationships between their nodes (e.g., in [3, 48]). Such relationships are basically a set of input-output specification for every node, consisting of every possible combination of values of every node regulating it. To generate the required FASP program for computing the attractors, we need to represent these relationships in the form of fuzzy logic formulas under Łukasiewicz semantics, such as the ones given in Example 9. It is not always straightforward for a human expert to find a suitable formula that fits a certain input-output relationship specification. We therefore provide a tool, called FASPG³, that performs this task automatically, and then invokes a FASP solver to compute the attractors of the GRN. Figure 4.3 shows the work flow of FASPG.

The input for FASPG is the description of a network, consisting of:

- The number of nodes, n
- The number of activation levels each node has, k
- An input-output specification for every node (described below)

An input-output specification of a node is a set of assignments for that node, given all possible combinations of the nodes regulating it. For example, consider a node x regulated by m nodes, y_1, \dots, y_m . Then, the input-output specification for x is

³FASPG is available at <http://github.com/mushthofa/faspg>

Figure 4.3 FASPG work flow

a table of k^m rows, each row consisting of a possible combination of the values of the y_i 's and a corresponding value for x .

Given such an input-output specification for a node, FASPG automatically constructs a correct set of Łukasiewicz logic formulas that evaluates to the required value for the node, following the construction process outlined in Proposition 2.

Proposition 2. *Suppose we are given that x has value v whenever each y_i has the value v_i , $i = 1, \dots, m$. Consider the program $F(x, v)$ consisting, for each i , of the following rules:*

$$\begin{aligned}
 p_i &\leftarrow y_i \otimes \overline{1 - v_i} \\
 p_i &\leftarrow p_i \oplus p_i \\
 q_i &\leftarrow \text{not } y_i \otimes \overline{v_i} \\
 q_i &\leftarrow q_i \oplus q_i \\
 c_i &\leftarrow \text{not } p_i \otimes \text{not } q_i
 \end{aligned}$$

and the single rule

$$x \leftarrow c_1 \otimes \dots \otimes c_m \otimes \overline{v}$$

It holds that in any answer set I of $F(x, v)$, $I(x) = v$ whenever $I(y_i) = v_i$ for every $i = 1, \dots, m$.

Proof. Intuitively, the atoms p_i and q_i are Boolean atoms signifying the condition of whether the value of y_i is $> v_i$ and $< v_i$, respectively. Therefore, the atom c_i , which is only true when both p_i and q_i are false, encodes the condition when the value of y_i is exactly v_i . The last rule of $F(x, v)$ then assigns the value of v to x , given that all c_i 's are true. \square

Such an encoding is applied to every row in the input-output relationship table, and then used in the program encoding for the computation of the attractor. Note that this encoding is not the only possible one we can come up with, nor is it necessarily the most efficient one, but as the experiments below will show, it is efficient enough for real-world networks.

After obtaining the encoding for the regulatory relationships, FASPG writes the remaining program encoding for the appropriate problem, and then submits it to the FASP solver FFASP⁴ [31, 32], which in turn performs the translation to ASP and calls the ASP solver CLINGO [19]. The attractors are then deduced from the resulting answer sets by FASPG.

4.5 Benchmark and Experiments

In the literature, little work has been done so far on computing attractors of multi-valued networks obtained from biological knowledge, due to the lack of appropriate tools to perform analysis on multi-valued networks. Our work is aimed to address this issue. In order to show the applicability of our approach, we collected several multi-valued networks obtained from the known biological networks in the literature. We run our approach on these networks and verify the expected results. Furthermore, to test the scalability of our approach, we also applied it to randomly generated synthetic networks and measure the time and memory requirements. All experiments were run on a machine with an 2.5GHz Intel Xeon CPU and a maximum of 15 GB of allowed memory consumption.

4.5.1 Experiments on real networks

To evaluate the correctness and efficiency of our method, we have tested it on a number of biological network models obtained from the literature. Table 4.2 represents the summary of the data collected. In each of these networks, each node is either Boolean-valued, or three-valued (represented as either the values 0, 1 and 2 or 'low', 'medium' and 'high' in the papers originally describing these multi-valued networks), except for the *D. melanogaster* segmentation network which uses a four-valued logical model. In encoding the regulatory relationships between the nodes in the network, we assign values from \mathbb{Q}_k to any k -valued nodes.

⁴FFASP is available at <http://github.com/mushthofa/ffasp>

Consequently, in these network models, we only consider attractors reached from the set of states where the Boolean-valued nodes are assigned either 0 or 1, and 3-valued and 4-valued nodes are assigned values from \mathbb{Q}_3 and \mathbb{Q}_4 , respectively. To generate all the possible relevant states, we add a saturation rule as described in Section 4.3 to each Boolean node x . For each of these models, the steady-states are computed, and compared to the ones reported in their respective reference(s).

For the *A. thaliana* flowering network, the network update functions are listed in [3] as name-values pairs indicating the input-output pairs of the update function on each node. For the Th cell regulatory network, [51] proposed different versions of the network. For our purpose, we use the logical rules presented in the Equation 2 in that paper, and evaluate them as 3-valued Łukasiewicz functions (i.e., treating \vee and \wedge as \oplus and \otimes , respectively), which is equivalent to the 1-hot encoding used in [2]. For the *D. melanogaster* segmentation network, the network update functions are represented using the notation used in [47]. By ignoring the time-delay parameter of this representation and using the assumption of the basal-expression levels of the genes to be 0 (as also done in [25]), we can faithfully represent each of the update functions given using Łukasiewicz logic formulas.

Table 4.2 shows, for each network, the number of nodes (n), the number of Boolean nodes, the number of possible activation levels (k), the number of steady-states found, and the computation time using our method. We can see that for the largest network ($n = 23$), the computation time is still very manageable (< 5 seconds). Except for the *A. thaliana* flowering network, we have taken advantage of the fact that the source literature already represented the update function as a logical function that can be directly translated into Łukasiewicz logic formulas. This might not always be the case, as shown in the *A. thaliana* network, where the interaction network was given just in the form of input-output pairs between the regulating nodes and the regulated node. In such cases, FASPG relies on the construction process from Proposition 2 to automatically generate the update function. These automatically-generated formulas, despite being correct, might cause the computations to take more time compared to manually crafted ones. The following subsection details an experiment on applying our method to synthetic networks to gain a more realistic picture of the computational requirements when we use FASPG to assist in the encoding of the interaction network.

4.5.2 Experiments on synthetic networks

Due to the limited availability of results about them in the literature, experiments on real biological networks can only paint a small picture on the efficiency of the application of the proposed method. Furthermore, the benchmark test on real networks that we presented in the previous subsection was limited to only the computation of steady states, due to the non-availability of cyclic attractor data

for any of the networks. Additionally, we would like to see the effects of using FASPG's automatic encoding of the interaction network. Below we therefore apply the method on randomly generated networks. These additional experiments are intended to assess the computational resources (in terms of time and space) needed to run the method, given increasing values of n and k . To this end, we generated 5 random networks for each combination of n and k , ranging from $n = 5$ to $n = 50$ with a step of 5, and $k = 1$ to $k = 6$. To generate realistic network topologies, we follow the procedure for generating random scale-free networks as given in [52]. Briefly, during the random network generation, each node is added one by one. At each step, the probability that an existing node is connected to a new node is proportional to its current degree. The directionality of the interactions are then chosen randomly. Furthermore, to limit the computational burden, we restrict the number of incoming regulatory interactions for a node to be within the range of 1 to 5. In each of these regulatory relationships, a set of random input-output relationships are generated (which covers every possible combination of values for the regulators).

For each of these random networks, we solve the following tasks using FASPG:

- Find all steady states of the network.
- Find at least one cyclic attractor with size < 5 using either synchronous or asynchronous updates (or report that there are none).

In each of our runs, we record the running time and the maximum memory usage. We set a time-out of 20 minutes per computation. For every combination of n and k , we run the method on 5 different randomly-generated networks, and we report the average of the running times and memory usages on the 5 networks, unless we observe a time-out or a memory-out in any of the 5 networks, in which case we report it as a failure.

Figure 4.4 and Figure 4.5 show the computation time and memory usage of the algorithm in finding all steady states, respectively. Overall, we notice that the method performs quite well in computing steady states for lower values of k , with the largest instance ($n = 50$) requiring less than 5 minutes, on average, to complete. However, we can clearly see that the bottleneck is in k , and for $k \geq 5$, computation time as well as memory usage increase drastically with larger values of n .

Figure 4.6 and Figure 4.7 show the computation time and memory usage for finding cyclic attractors using synchronous updates. Overall, we see that finding cyclic attractors generally takes more time and memory than finding steady states. The overall trend that k seems to be the bottleneck can still be observed, with even more time-outs. For $k = 1$, no time-outs are observed for the network sizes considered. For larger k , we start to observe more and more time-outs, with $k = 4$ having time-outs for $n > 5$.

Figure 4.4 Running time for computing steady states. Missing nodes indicate failure due to time-outs/memory-outs.

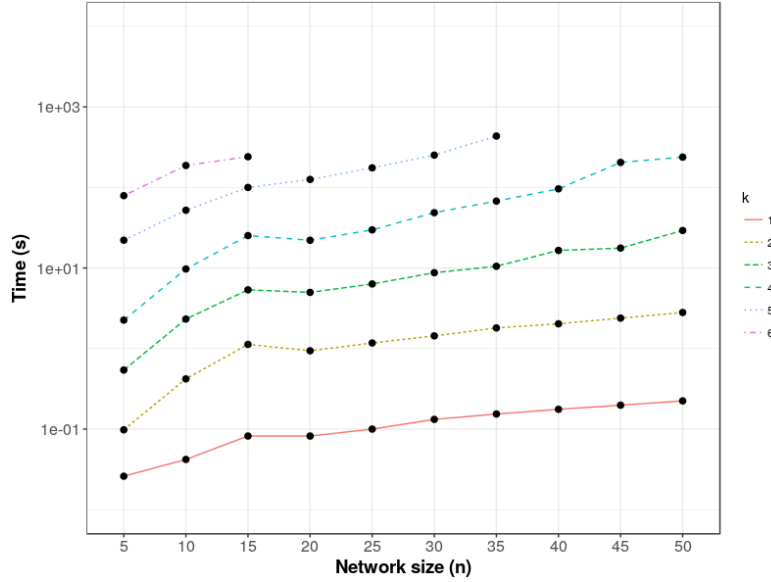


Figure 4.5 Memory usage for computing steady states. Missing nodes indicate failure due to time-outs/memory-outs.

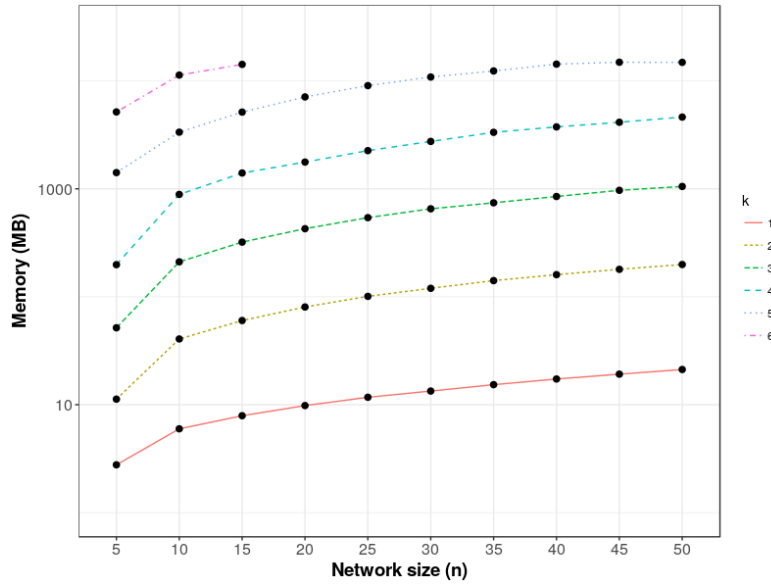


Figure 4.6 Running time for computing synchronous cyclic attractors. Missing nodes indicate failure due to time-outs/memory-outs. The singleton node represents $k = 4$. All instances with $k > 4$ failed due to time-outs/memory-outs.

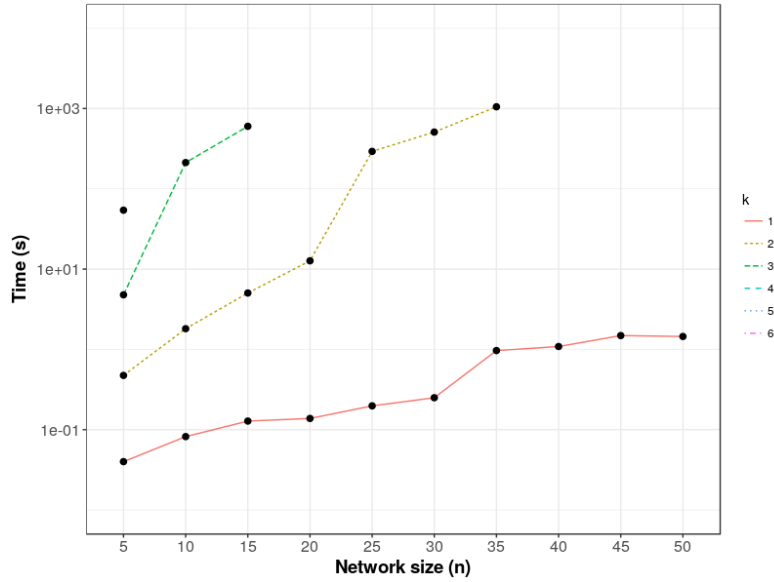


Figure 4.7 Memory usage for computing synchronous cyclic attractors. Missing nodes indicate failure due to time-outs/memory-outs. The singleton node represents $k = 4$. All instances with $k > 4$ failed due to time-outs/memory-outs.

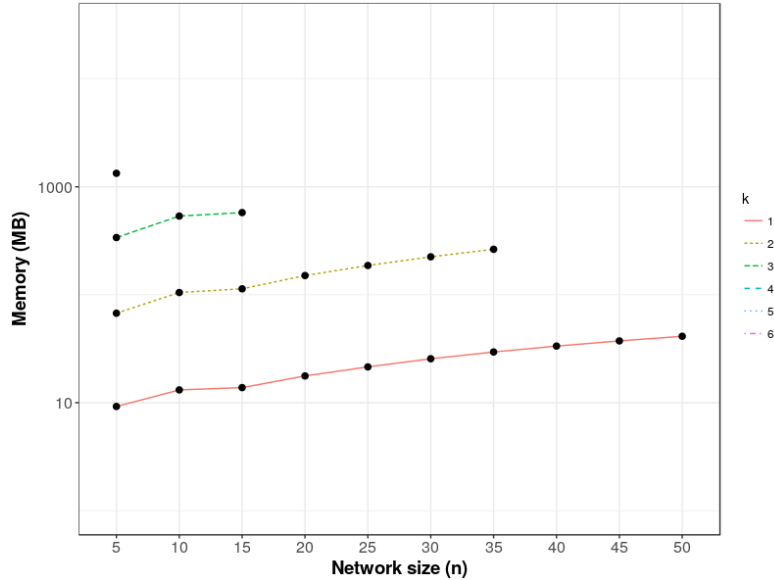


Figure 4.8 Running time for computing asynchronous cyclic attractors. Missing nodes indicate failure due to time-outs/memory-outs.

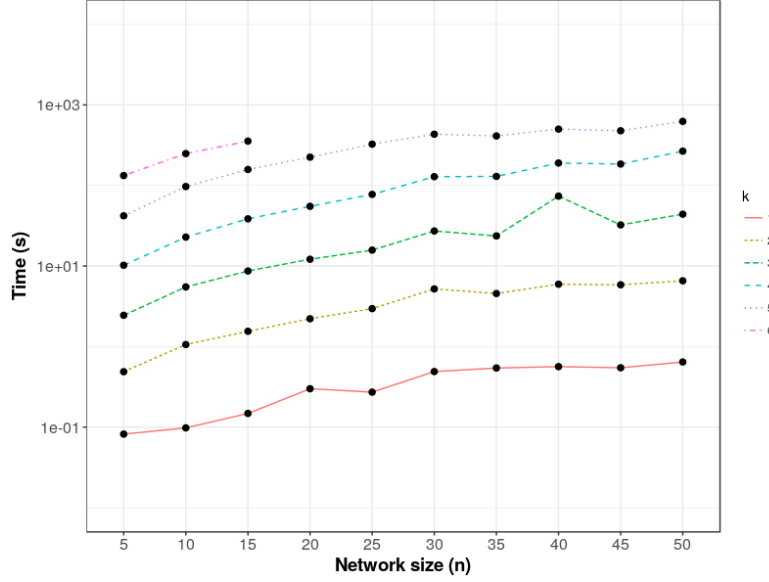
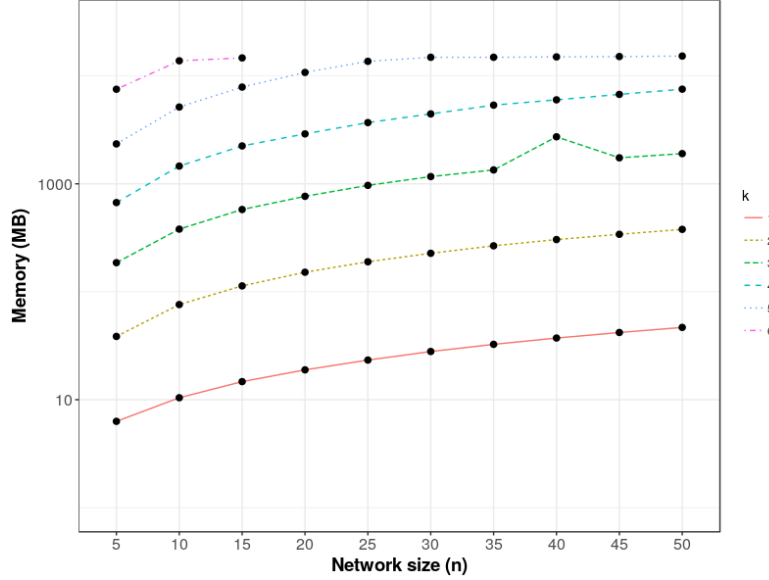


Figure 4.8 and Figure 4.9 show the computation time and memory usage for finding cyclic attractors using asynchronous updates. Here, we notice that the time requirement for finding asynchronous cyclic attractors is, in general, lower than in the synchronous case. This is probably due to the more stringent criteria applied to the dynamics (in which only one possible successor state is allowed) which can be exploited by the solver. No time-outs are observed for the network sizes considered. However, we see a larger memory-usage than for either the steady-states and synchronous cyclic attractors, with larger instances having memory outs. In conclusion, we observe, as expected, that time and memory requirements generally increase exponentially w.r.t the size of the network (n), while the number of possible values in the activation level of the genes (k) serves as an exponential factor. In addition, we observe that computing steady states generally has a lower computational requirements than computing synchronous and asynchronous attractors. Synchronous attractor computation generally requires more computation time than the other two, while memory consumption is generally the biggest bottleneck in asynchronous attractor computation.

Figure 4.9 Memory usage for computing asynchronous cyclic attractors. Missing nodes indicate failure due to time-outs/memory-outs.



4.6 Conclusion

ASP has been successfully applied to model the dynamics of gene regulatory networks in the Boolean setting; see e.g. [14, 15]. In these works, the encoding of the update function is restricted to two specific types (denoted as r^* and r^+ in [14]), due to the particular way that the encoding of the dynamics is written (i.e., encoding the update function at a *meta-level*). In [16], it was suggested that each of the node's update functions of a Boolean network can be directly encoded as a rule in ASP. This allows for a more generic encoding of the network update function. Furthermore, it was shown that the steady states of the network are directly obtainable using the semantics of ASP. To obtain the cyclic attractors, [16] proposes an extension of the ASP semantics which allows to capture cyclic attractors “naturally” as answer sets of the program. Such an extension is not obvious nor easy to develop and implement, however, since it requires the redefinition of the basics of ASP, as well as the reimplementing of currently available solvers. In addition, this method is only geared towards Boolean networks, instead of multi-valued networks.

In this chapter, we have describe our proposal for the method to use FASP, an extension of ASP in the continuous domain, as a convenient language for encoding the dynamics of multi-valued networks. To the best of our knowledge, this is

the first real-world application of FASP that goes beyond small toy examples. We showed the correctness of our encoding, and we evaluated its efficiency for computing the steady-states of real biological networks found in the literature. The experimental results show that the proposed method works quite efficiently, especially for finding the most biologically-relevant type of attractors, which are the steady-states and small-sized attractors. The method incorporates two distinguishing characteristics:

- It allows graded activation levels in the nodes of the networks instead of only “on” and “off”, and
- It allows a more flexible definition of the network update function by encoding the dynamics of the network using a *time* argument. In contrast to the approach used in [14, 49], the use of the time argument and the direct encoding of the network update function allows for a more general relationships between interacting nodes. Additionally, this alleviates the requirement to extend/redefine the theoretical notion of answer sets in logic programming, as is required by the approach used in [16] for encoding the computation of cyclic attractors.

Table 4.2 Benchmark results.

No.	network (and references)	# of nodes	# of Boolean nodes	k	# of steady-states	time (seconds)
1	<i>P. aeruginosa</i> mucus development network [48, 50]	2	1	3	2	0.03
2	<i>A. thaliana</i> flowering network [3]	15	7	3	10	2.87
3	Th cell regulatory network [2, 51]	23	9	3	4	4.3
4	<i>D. melanogaster</i> segmentation network [25]	7	4	4	4	4.2

References

- [1] Stuart A. Kauffman. *The origins of order: Self-organization and selection in evolution*. Oxford university press, 1993.
- [2] Luis Mendoza. *A network model for the control of the differentiation process in Th cells*. Biosystems, 84(2):101–114, 2006.
- [3] Carlos Espinosa-Soto, Pablo Padilla-Longoria, and Elena R Alvarez-Buylla. *A gene regulatory network model for cell-fate determination during Arabidopsis thaliana flower development that is robust and recovers experimental gene expression profiles*. The Plant Cell Online, 16(11):2923–2939, 2004.
- [4] Stuart A Kauffman. *Metabolic stability and epigenesis in randomly constructed genetic nets*. Journal of theoretical biology, 22(3):437–467, 1969.
- [5] Marcelle Kaufman, Jacques Urbain, and René Thomas. *Towards a logical analysis of the immune response*. Journal of theoretical biology, 114(4):527–561, 1985.
- [6] Hidde De Jong. *Modeling and simulation of genetic regulatory systems: a literature review*. Journal of computational biology, 9(1):67–103, 2002.
- [7] Abhishek Garg, Ioannis Xenarios, Luis Mendoza, and Giovanni DeMicheli. *An efficient method for dynamic analysis of gene regulatory networks and in silico gene perturbation experiments*. In Research in Computational Molecular Biology, pages 62–76. Springer, 2007.
- [8] Gustavo Arellano, Julian Argil, Eugenio Azpeitia, Mariana Benitez, Miguel Carrillo, Pedro Gongora, David Rosenblueth, and Elena Alvarez-Buylla. *“Antelope”: a hybrid-logic model checker for branching-time Boolean GRN analysis*. BMC Bioinformatics, 12(1):490, 2011.
- [9] Nikolaos Berntenis and Martin Ebeling. *Detection of attractors of large Boolean networks via exhaustive enumeration of appropriate subspaces of the state space*. BMC Bioinformatics, 14(1):1–10, 2013.
- [10] Ferhat Ay, Fei Xu, and Tamer Kahveci. *Scalable Steady State Analysis of Boolean Biological Regulatory Networks*. PLoS ONE, 4(12):e7992, 12 2009.
- [11] Elena Dubrova and Maxim Teslenko. *A SAT-based algorithm for finding attractors in synchronous Boolean networks*. IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB), 8(5):1393–1399, 2011.
- [12] Desheng Zheng, Guowu Yang, Xiaoyu Li, Zhicai Wang, Feng Liu, and Lei He. *An Efficient Algorithm for Computing Attractors of Synchronous And Asynchronous Boolean Networks*. PloS one, 8(4):e60593, 2013.

- [13] Steve Dworschak, Susanne Grell, Victoria J Nikiforova, Torsten Schaub, and Joachim Selbig. *Modeling biological networks by action languages via answer set programming*. Constraints, 13(1-2):21–65, 2008.
- [14] Mushthofa Mushthofa, Gustavo Torres, Yves Van de Peer, Kathleen Marchal, and Martine De Cock. *ASP-G: an ASP-based method for finding attractors in genetic regulatory networks*. Bioinformatics, 30(21):3086, 2014.
- [15] Timur Fayruzov, Martine De Cock, Chris Cornelis, and Dirk Vermeir. *Modeling protein interaction networks with answer set programming*. In Bioinformatics and Biomedicine, 2009. BIBM’09. IEEE International Conference on, pages 99–104. IEEE, 2009.
- [16] Katsumi Inoue. *Logic programming for Boolean networks*. In Proceedings of the Twenty-Second international joint conference on Artificial Intelligence-Volume Volume Two, pages 924–930. AAAI Press, 2011.
- [17] Vladimir Lifschitz. *What Is Answer Set Programming?*. In AAAI, volume 8, pages 1594–1597, 2008.
- [18] Chitta Baral. *Knowledge representation, reasoning and declarative problem solving*. Cambridge University Press, 2003.
- [19] Martin Gebser, Benjamin Kaufmann, Roland Kaminski, Max Ostrowski, Torsten Schaub, and Marius Schneider. *Potassco: The Potsdam answer set solving collection*. AI Communications, 24(2):107–124, 2011.
- [20] Nicola Leone, Gerald Pfeifer, Wolfgang Faber, Thomas Eiter, Georg Gottlob, Simona Perri, and Francesco Scarcello. *The DLV system for knowledge representation and reasoning*. ACM Transactions on Computational Logic (TOCL), 7(3):499–562, 2006.
- [21] Thomas Eiter, Giovambattista Ianni, and Thomas Krennwallner. *Answer set programming: A primer*. In Reasoning Web. Semantic Technologies for Information Systems, pages 40–110. Springer, 2009.
- [22] Esra Erdem. *Theory and Applications of Answer Set Programming*. PhD thesis, 2002. The University of Texas at Austin.
- [23] Gilles Didier, Elisabeth Remy, and Claudine Chaouiya. *Mapping multivalued onto Boolean dynamics*. Journal of Theoretical Biology, 270(1):177 – 184, 2011.
- [24] Abhishek Garg, Luis Mendoza, Ioannis Xenarios, and Giovanni DeMicheli. *Modeling of multiple valued gene regulatory networks*. In Proceedings of the 29th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBS 2007), pages 1398–1404. IEEE, 2007.

- [25] Lucas Sanchez and Denis Thieffry. *Segmenting the fly embryo:: a logical analysis of the pair-rule cross-regulatory module*. Journal of Theoretical Biology, 224(4):517–537, 2003.
- [26] Alexander Bockmayr and Heike Siebert. *Programming Logics: Essays in Memory of Harald Ganzinger*, chapter Bio-Logics: Logical Analysis of Bioregulatory Networks, pages 19–34. Springer Berlin Heidelberg, 2013.
- [27] Harvey Lodish, Arnold Berk, S Lawrence Zipursky, Paul Matsudaira, David Baltimore, James Darnell, et al. *Molecular cell biology*, volume 5. WH Freeman New York, 2000.
- [28] Davy Van Nieuwenborgh, Martine De Cock, and Dirk Vermeir. *Fuzzy answer set programming*. In Proceedings of the 10th European Conference on Logics in Artificial Intelligence, pages 359–372. 2006.
- [29] Marjon Blondeel, Steven Schockaert, Dirk Vermeir, and Martine De Cock. *Complexity of fuzzy answer set programming under Łukasiewicz semantics*. International Journal of Approximate Reasoning, 55(9):1971–2003, 2014.
- [30] Mario Alviano and Rafael Peñaloza. *Fuzzy answer sets approximations*. Theory and Practice of Logic Programming, 13(4-5):753–767, 2013.
- [31] Mushthofa Mushthofa, Steven Schockaert, and Martine De Cock. *A finite-valued solver for disjunctive fuzzy answer set programs*. In Proceedings of European Conference in Artificial Intelligence 2014, pages 645–650, 2014.
- [32] Mushthofa Mushthofa, Steven Schockaert, and Martine De Cock. *Solving Disjunctive Fuzzy Answer Set Programs*. In Proceedings of the 13th International Conference on Logic Programming and Non-monotonic Reasoning, pages 453–466, 2015.
- [33] Mario Alviano and Rafael Peñaloza. *Fuzzy answer set computation via satisfiability modulo theories*. Theory and Practice of Logic Programming, 15:588–603, 7 2015.
- [34] Peter Vojtáš. *Fuzzy logic programming*. Fuzzy sets and systems, 124(3):361–370, 2001.
- [35] Joohyung Lee and Yi Wang. *Stable Models of Fuzzy Propositional Formulas*. In Proceedings of the 14th European Conference on Logics in Artificial Intelligence, JELIA 2014, page 326. 2014.
- [36] Nicolás Madrid and Manuel Ojeda-Aciego. *Towards a Fuzzy Answer Set Semantics for Residuated Logic Programs*. In Web Intelligence/IAT Workshops, pages 260–264, 2008.

- [37] Carlos Viegas Damásio and Luís Moniz Pereira. *Antitonic Logic Programs*. In Proceedings of the 6th International Conference on Logic Programming and Nonmonotonic Reasoning, pages 379–392. 2001.
- [38] Umberto Straccia. *Annotated answer set programming*. In In: Proceedings of the 11th International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems (IPMU-06, 2006.
- [39] Umberto Straccia. *Managing Uncertainty and Vagueness in Description Logics, Logic Programs and Description Logic Programs*. In Reasoning Web, 4th International Summer School, Tutorial Lectures, volume 5224 of *Lecture Notes in Computer Science*, pages 54–103. Springer Verlag, 2008.
- [40] Luis Mendoza and Elena R Alvarez-Buylla. *Dynamics of the Genetic Regulatory Network for Arabidopsis thaliana Flower Morphogenesis*. Journal of theoretical biology, 193(2):307–319, 1998.
- [41] Luis Mendoza, Denis Thieffry, and Elena R Alvarez-Buylla. *Genetic control of flower morphogenesis in Arabidopsis thaliana: a logical analysis*. Bioinformatics, 15(7):593–606, 1999.
- [42] Yara-Elena Sanchez-Corrales, Elena R Alvarez-Buylla, and Luis Mendoza. *The Arabidopsis thaliana flower organ specification gene regulatory network determines a robust differentiation process*. Journal of theoretical biology, 264(3):971–983, 2010.
- [43] Fangting Li, Tao Long, Ying Lu, Qi Ouyang, and Chao Tang. *The yeast cell-cycle network is robustly designed*. Proceedings of the National Academy of Sciences of the United States of America, 101(14):4781–4786, 2004.
- [44] Lucas Sánchez and Denis Thieffry. *A Logical Analysis of the Drosophila Gap-gene System*. Journal of theoretical Biology, 211(2):115–141, 2001.
- [45] Réka Albert and Hans G Othmer. *The topology of the regulatory interactions predicts the expression pattern of the segment polarity genes in Drosophila melanogaster*. Journal of theoretical biology, 223(1):1–18, 2003.
- [46] Aitor González, Claudine Chaouiya, and Denis Thieffry. *Logical modelling of the role of the Hh pathway in the patterning of the Drosophila wing disc*. Bioinformatics, 24(16):i234–i240, 2008.
- [47] Rene Thomas. *Regulatory networks seen as asynchronous automata: a logical description*. Journal of Theoretical Biology, 153(1):1–23, 1991.
- [48] Janine Guespin-Michel and Marcelle Kaufman. *Positive Feedback Circuits and Adaptive Regulations in Bacteria*. Acta Biotheoretica, 49(4):207–218, 2001.

- [49] Timur Fayruzov, Martine De Cock, Chris Cornelis, and Dirk Vermeir. *Modeling protein interaction networks with answer set programming*. In Bioinformatics and Biomedicine, 2009. IEEE International Conference on, pages 99–104. IEEE, 2009.
- [50] Sabine Peres and Jean-Paul Comet. *Contribution of Computational Tree Logic to Biological Regulatory Networks: Example from Pseudomonas Aeruginosa*. In Proceedings of the First International Workshop on Computational Methods in Systems Biology (CMSB 2003), pages 47–56, 2003.
- [51] Luis Mendoza and Ioannis Xenarios. *A method for the generation of standardized qualitative dynamical systems of regulatory networks*. Theoretical Biology and Medical Modelling, 3(1):13, 2006.
- [52] Réka Albert and Albert-László Barabási. *Statistical mechanics of complex networks*. Reviews of modern physics, 74(1):47, 2002.

5

Network-based predictive modelling for cancer drug response

5.1 Introduction

Over the last few years international efforts have resulted in the profiling of 1000 of cancer genomes of different cancer types (ICGC). Based on these results, hundreds of cancer drivers have been identified. These drivers serve as novel drug targets or as bio markers for prognosis and/or personalized therapy prediction. In current medical practice, panels of actionable drivers have been proposed that are screened for by targeted sequencing in order to decide on a per mutation basis whether or not a patient would benefit from a certain therapy. For a clonal disease such as cancer such single mutation-based decision making has drawbacks. This is because most of the current actionable drivers have been identified by searching in panels of tumor genomes for recurrently occurring mutations that can be associated with the disease. Actionable mutations in clinical screening panels thus correspond to frequently occurring mutations. However, cancer is a clonal disease and therefore not all tumors that have the same sub-type and that would possibly benefit from a certain therapy will carry exactly the same mutation. Previous cohort analyses has shown that molecular sub-types originate by hitting the same driver pathway but that this can occur in many different and often mutually exclusive ways. By screening for the most frequently occurring actionable mutations, patients that carry rare mutations in the same driver pathway that is affected by the frequent actionable mutation will be missed, resulting in a high false negative rate

(patients that would potentially benefit from therapy would not receive therapy). By screening for the most frequent mutations only around 11%-86% (depending on cancer types and study) would receive a therapy [1–3].

In addition, cancer is a complex disease and primary sub-types but also other clinically relevant phenotypes such as drug resistance depend on the aberration of multiple pathways. Clinical practice has indeed shown that many of the patients that carry a so-called positive actionable mutation do not respond to therapy as predicted (false positives). This is mainly because panels of mutations that are screened for in clinical practice only focus on the actionable mutations, but have not been designed to also take into account the genetic context in which the actionable mutation occurs. As a result, for each therapy (e.g. drug) designing a proper screening panel would ideally require designing sets of mutations, rather than single mutations that characterize the full actionable driver pathway and genetic context in which a driver pathway appears actionable. Such a biomarker design thus requires a protocol that pools information from large patient cohorts with well-defined clinical properties to identify recurrently affected pathways or networks.

Network-based approaches, which use a molecular interaction network to steer the driver identification problem are ideally suited for the detection of recurrently mutated pathways and have already been successfully applied to analyze patient cohorts in TCGA/ICGC. However, as clinical information is relatively sparse in TCGA/ICGC and TCGA/ICGC contains mainly primary tumors, these methods have mainly been used for unsupervised driver identification and/or sub-typing. This will result in molecular sub-types that can explain the origin of a disease, but that do not necessarily have clinical implications. In addition, cancer cells have a fast evolving mutational landscape and accumulate mutations that confer resistance to first line therapies and/or result in metastases which might be much more relevant towards predicting optimal therapies than the mutations that explain the molecular origin or tissue type of a cancer. Incorporating quantitative information on clinically relevant phenotypes is thus essential to identify pathways/genetic aberrations that associate with clinically relevant phenotypes but that might be less prominently present than the aberrations determining the molecular origin of the disease.

In this chapter, we described our proposed method of integrating biological network information for feature selection and data transformation for phenotypic prediction, in particular, for the context of cancer drug response prediction. To steer the search for causal genetic aberrations towards those that associate with a phenotype of interest, we modified a diffusion-based network approach to integrate phenotypic information with molecular profiling data in order to identify sub-networks that can explain quantitative differences in phenotypic behavior. As a proof-of-concept, we tested the methodology on recently described panels of cell

lines for which the quantitative response to different drugs was available. For well described drugs, the known drug targets and/or resistance conferring mutations were among the highly prioritized features, irrespective of the cell line and/or tissue type in which they occurred. As an additional validation of the biological relevance of our selected features we demonstrated that the identified features could be used to build a classifier that was able to distinguish between drug responsive and resistant cell lines. These results showed that the network based approach was able to accurately extract from molecular profiling data of tumor panels (patients or their preclinical models) the features that can be used as biomarkers to predict a clinical phenotype, such as drug response.

5.2 Results

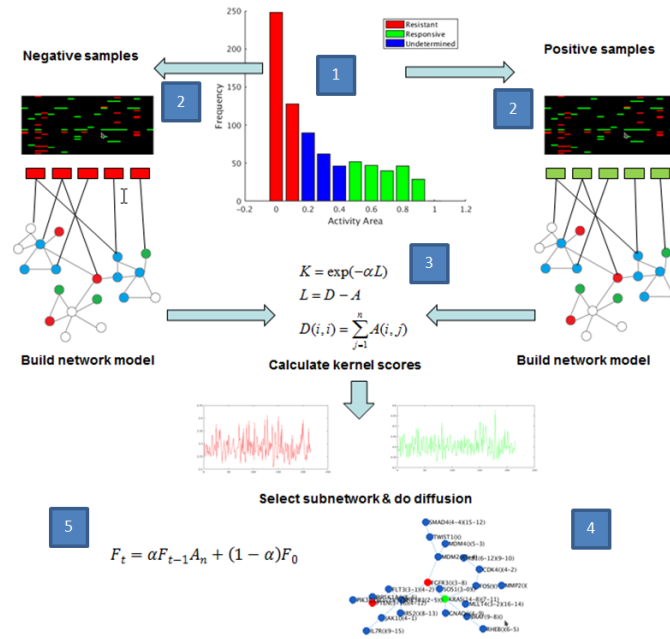
5.2.1 Network-based strategy molecular to identify sub-networks driving a clinical phenotype

Our method aims at identifying molecular sub-networks that are recurrently affected across a cohort of tumor samples and that associate with a phenotype of interest. The overview of the method is given in Figure 5.1. and builds upon a previously published integrative network based method [4]. The input consists of a set of genotyped samples (tumor samples, cell lines) that are both clinically and molecularly phenotyped (e.g. by transcriptome profiling).

Based on their phenotype, samples are subdivided in two groups containing respectively samples with a positive and samples with a negative phenotype. Per group we build a dedicated network model in which both samples and genetic entities are represented by nodes. Genetic entity nodes can be connected to other genetic entity nodes if information on an interaction between the entities is available *a priori* e.g. from public databases such as KEGG [5], Reactome [6] & The Atlas of Cancer Signaling Network (ACSN) [7]. Genetic entity nodes are also connected to status nodes provided a genetic entity was found in an aberrant state in at least one sample. Different types of status nodes exist depending on the type of aberration i.e. reflecting whether a gene contains a copy number variation (CNV), a somatic mutation (MUT), or whether it is differentially expressed. The status of a gene is indicated in a binary way (1 is aberrant status and 0 is normal status). Status nodes are connected to the respective sample nodes in which the aberrant status was observed. To incorporate phenotypic information, connections between sample and status nodes are weighed using the phenotypic strength (i.e., how strongly they respond to the drug) displayed by the sample.

Subsequently the network model is used to assess the similarities between the sample and respectively the status and entity nodes. Similarity scores are based computed based on kernel diffusion as outlined in Verbeke et. al [4]. The respon-

Figure 5.1 Overview of method. 1) A pair of thresholds is chosen to define the resistant and responsive cell lines for a certain drug, 2) molecular data from the set of cell lines on each set are combined and integrated on top of the biological network, 3) kernel scores are computed and averaged for all cell lines within each set, 4) rankings of features are determined by the largest differential scores between the two sets, 5) Train a classifier, and then test using new data with/without diffusion.



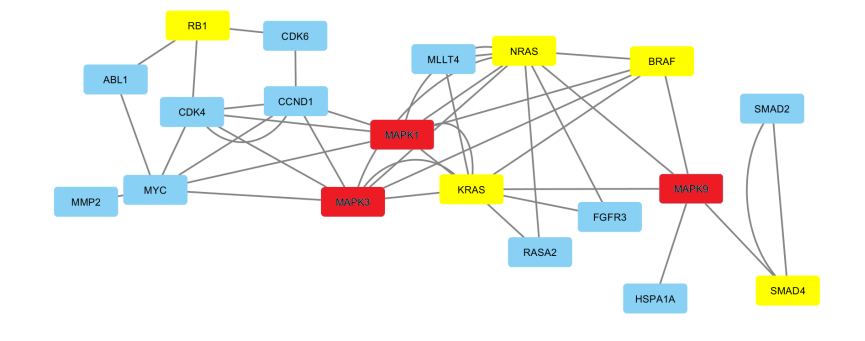
ders and non-responders sets are analyzed separately, hereby assuming that within one set i.e. respectively responders or non-responders the same pathways will be triggered. Molecular aberrations or genes in an individual sample relevant to the phenotype of the sample will through the network based diffusion be reinforced by aberrations occurring in other samples of same set that are located in the same entities or at least in the same local network neighborhood of the affected entity. The similarity scores between sample-entity or between sample-status nodes reflect the importance of respectively an entity or status node in driving the phenotype of the sample. Entity or status nodes that are highly relevant for the responders (i.e. with a high average similarity score for the samples in the responder set) and non-relevant for the non-responders (i.e. with a low average similarity score for the samples in the non-responder set) or vice versa intuitively represent the gene sets (entity nodes) and their genomic/transcriptomic features (status nodes) that are most discriminative for both phenotypes. Therefore, based on the difference in score a node obtained in respectively the responder and non-responder set, a node rank is assigned. Nodes are assigned a positive rank if they were relatively more relevant for the responsive phenotype than for the non-responsive one and a negative rank if the opposite was true (i.e. the sign of the rank is based on the sign of the difference between the average scores of the node-sample similarities obtained in either set, the rank itself is based on the absolute difference in these nodes scores). Status nodes are scored per status type (MUT or NET, see the section on Methods and Data) separately to assess the marginal effect of each aberration type in prioritizing the corresponding genetic entity (to which the status node is connected).

Using these selected nodes, we then train a random forest classifier to classify the two classes: responders and non-responders. We compare the performance of the resulting classifier with a baseline random forest classifier trained using statistically-selected features. As an additional step in our method, we also propose a data transformation approach using a PageRank-like [8] diffusion on the selected sub-networks, and trained another random forest classifier using the transformed data (see Methods and Data for details). The resulting random forest classifiers trained using the network-selected features and the transformed data are then compared with the baseline random forest classifier in terms of classification performance in a cross-validated setting.

5.2.2 Application of the method to drug response prediction

To test our method, we applied it to recently published drug related cell-lines from the Genomics of Drug Sensitivity in Cancer (GDSC) [9]. This study profiled for a large set representative tumor cell-lines, transcriptomic and genetic data as well as the quantitative response to a large number of drugs. Cell-lines were stratified according to their drug response. To avoid disturbing the identification of pathway

Figure 5.2 Genes associated to the top ranking nodes selected for the MEK inhibitor PD-0325901, mapped on the gene-gene interaction network. Blue nodes indicate the genes that were selected as top ranking using the network based data integration based on their status nodes, while yellow nodes indicates the nodes associated to top predictive features. Red nodes were not selected by our method, but are displayed to show the connectivity between the selected nodes.



based features with noisy signals per drug, only the cell lines that show a clear (non) response signal were retained (details in Methods and Data). Applying our network-based integrative strategy on the data allowed prioritizing for each drug the entity and status nodes (MUT, CNV, NET) that associate with a difference in clinical phenotype to the drug. The degree to which each selected node associates to drug response is represented by its network similarity score, computed from the difference in the average normalized score of the nodes between the set of responsive and non-responsive samples. Analyzing quantitative drug response together with the genotype and expression profiling of the individual cell-lines will result, per drug, in pathway features that determine whether or not a cell-line is drug responsive. The major added value of network-based features as compared to features derived by a statistical method (e.g. random forest, ANOVA) is their biological interpretability. Mapping the top-ranking nodes to the corresponding genes in the gene-gene interaction network results per drug in a tightly connected sub-network, confirming that the method indeed enforces the selection of features that consistently trigger the same sub-network. This drug specific sub-network represents the molecular mode of action that results in the observed drug response. Figure 5.2 shows as a representative example the gene sub-network containing the 19 nodes associated with the features selected for PD-0325901, a known MEK-inhibitor. As shown here, the genes associated to these features are closely connected to the known (upstream) members of the MAP kinase pathway, suggesting their role in determining response to the MEK inhibitor.

5.2.3 Validation of the pathway-based features by assessing predictive power

To illustrate the relevance of the pathway-based features extracted by our network-based integration method, we assessed their added value in predicting drug response in a cross validation setting. To this end we trained a random forest fed with the best 30 network-based features and tested its accuracy in correctly predicting drug response of the left out samples (see Materials and Methods). We choose for a classifier that can make use of 30 features as this mimics a currently realistic clinical setting in which targeted screening of a medium-sized small biomarker panel would be used to stratify patients. We compared the performance of this network-feature-based classifier with a baseline i.e. a random forest trained with 30 statistically derived features (details in Methods and Data).

For ease of interpretation, the data set is subdivided into two sets: 1) the data set (156 out of the 214 drugs) in which a clear statistical signal is present that allows a feature extraction and classification performance that is larger than what can be expected at random, and 2) the data set in which such statistical signal is not obviously present (58 out of the 214 drugs). The presence of statistical signal (i.e., correlation between genomic features of the cell lines and the their drug response) was assessed by comparing the predictive performance of a random forest trained on all features with the performance of a random forest trained on the same features derived from a randomized data set (features remain the same but had been randomly-permuted to break the correlation with the drug response, see Methods and Data for details).

For data sets with a clear statistical signal, the random forests trained with network-based features performed better than the random forest fed with statistically derived features in 112 out of the 156 cases, indicating that in those cases, the network-based method does not only allow extracting the correct significant features in most cases but that it tends to also select less spurious features than the baseline random forest. Figure 5.3, 5.4 & 5.5 show the comparison of the accuracy results for these 156 drugs, where we split the plots based on whether the random forest classifier trained using our network-based feature selection could perform better than random (Figure 5.3 & 5.4) or not (Figure 5.5). In Figure 5.3 & 5.4 (total 131 drugs), we can see that the random forest trained using the network-selected features almost always outperformed the baseline random forest (105/131), and in many cases by significantly large margins. In Figure 5.5 (total 25 drugs), the improvement we gained from using the network-selected features are no longer as significant as the previous case, but we can see that there are still some improvements over the baseline method.

For the second subset of data (i.e., where there was no obvious signal correlating the genomic features to the drug response, consisting of 58 drugs), we consider only the case where our network-based feature selection method could

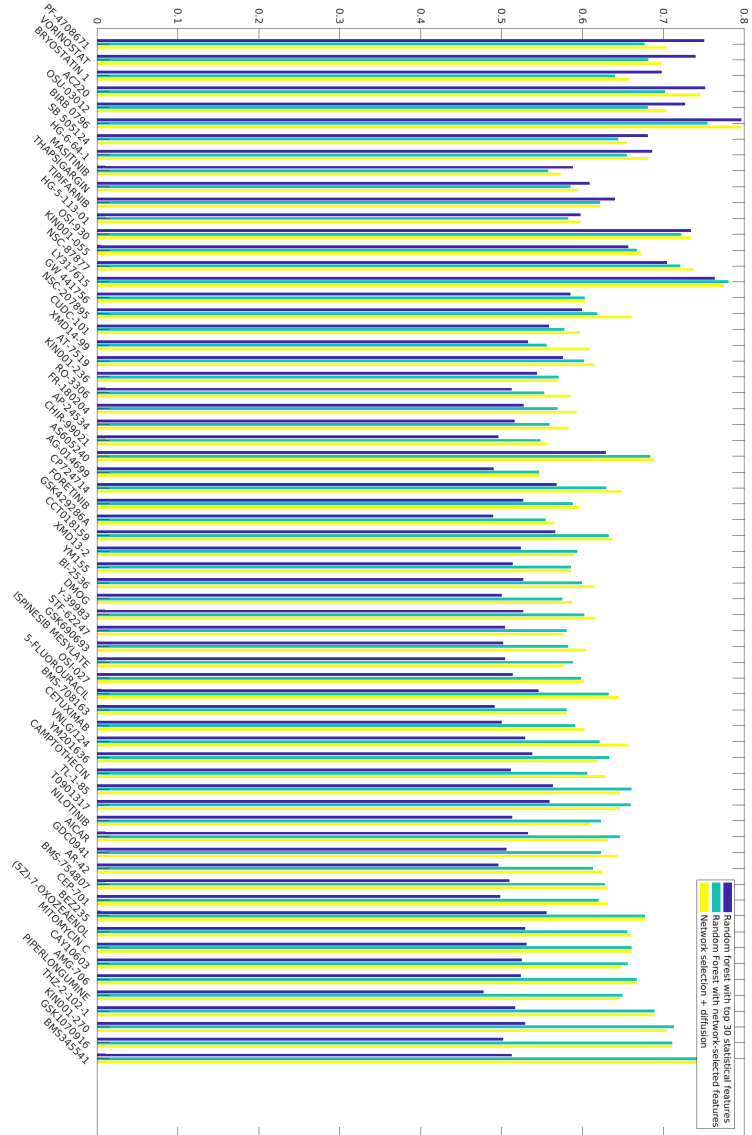


Figure 5.3: Comparison of the performance of random forest classifiers using the statistical feature selection (blue), random forest using network-based feature selection (green) and random forest using diffused data (yellow) in the case where a clear signal was present in the data and where the network feature selection method performed statistically better than random (part I).

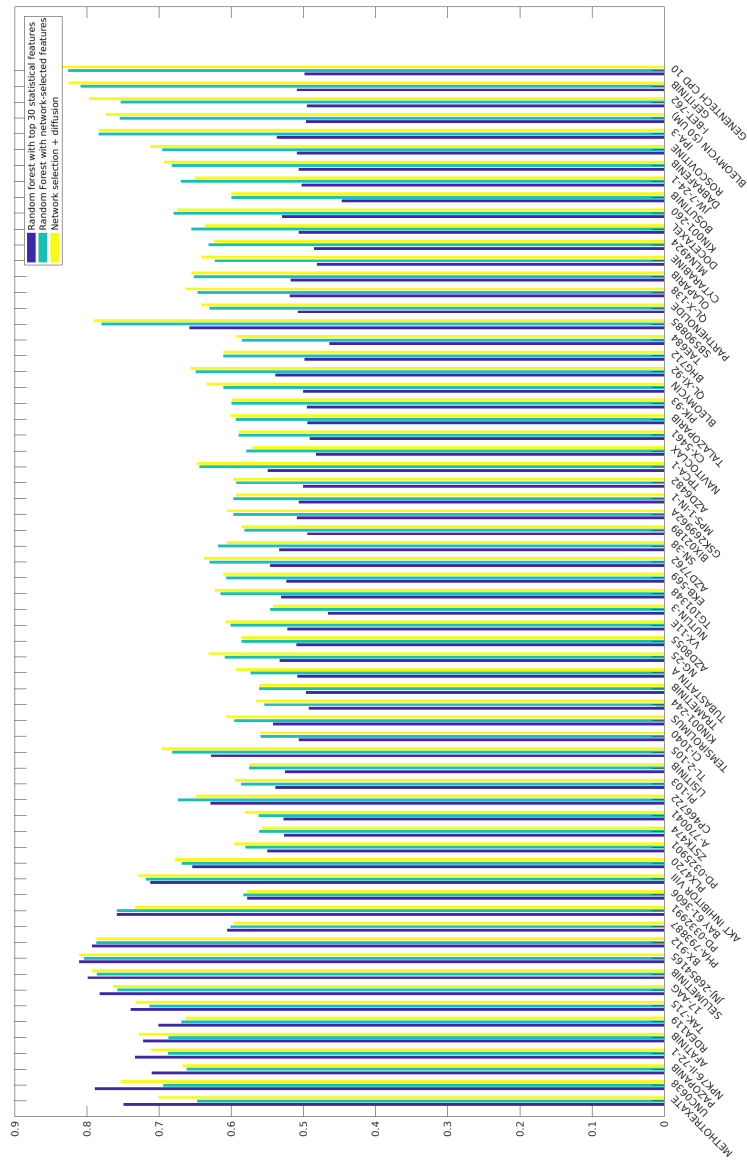


Figure 5.4: Comparison of the performance of random forest classifiers using the statistical feature selection (blue), random forest using network-based feature selection (green) and random forest using diffused data (yellow) in the case where a clear signal was present in the data and where the network feature selection method performed statistically better than random (part 2).

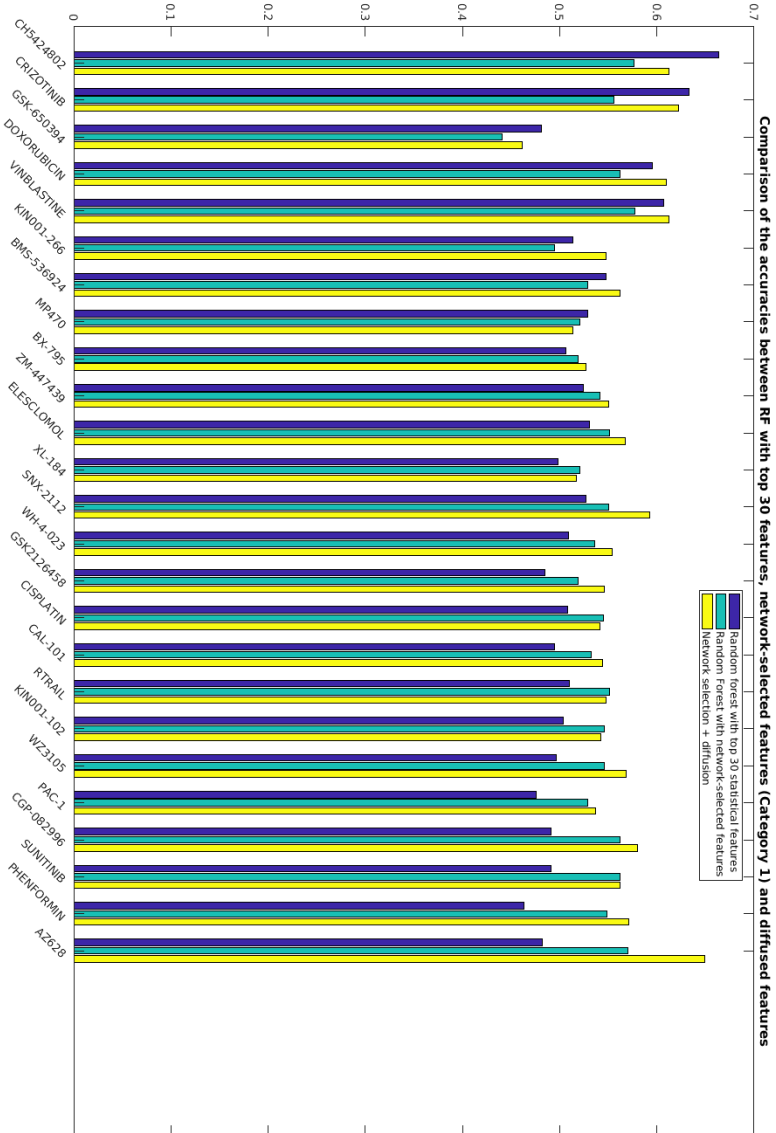


Figure 5.5: Comparison of the performance of random forest classifiers using the statistical feature selection (blue), random forest using network-based feature selection (green) and random forest using diffused data (yellow) in the case where a clear signal was present in the data and where the network feature selection method did not perform statistically better than random.

still perform statistically better than random (Figure 5.6, total 16 drugs). Here, we see that despite the fact that the signal in the data is not very strong, the network-based feature selection managed to improve slightly compared to the baseline. We do not consider the complimentary case (i.e., where our network-based feature selection method also could not perform statistically better than random), since this means that both our proposed method and the baseline method obtained poor performances, which probably indicates the low quality of the data and/or a very weak presence of signal in the data.

The previous results show that our network-feature selection is able to select relevant features and generally reduce the number of selected spuriously-correlated features. Although random forests are generally robust against the selection of spuriously-correlated features, the relative higher number of initial features in combination with the small sample size is likely to result in selecting at least some spurious features when performing a mere statistical selection. Imposing that informative features should belong to the same network neighborhood results is an extra biological constraint that reduces the number of spurious features. This higher robustness towards selecting spurious features is thus an inherent property of using the network prior during feature selection. However, in some cases, we also see that using the network-based selected features, classification performance decreased. The lower performance of the network-feature based classification as compared to the baseline in at least a subset of the cases is to be expected as the network-based method depends on the completeness of the interaction network. Features not connected in the network are harder to identify by the network method than with the random forest based feature extraction.

Comparing the network-feature based classifier with baseline already illustrated that using a network prior allows extracting relevant features. However, given the fact that a random forest is limited in the maximal number of features it can select, combining a network-feature selection with a random forest classifier does not yet extract the full potential of the identified drug specific network. As explained above network-based features by nature also include aberrations that occur infrequently in a cohort and therefore only have marginal explanatory or predictive power for the phenotype of interest (here, drug response). To maximally exploit these less predictive features that occur in an insignificant number of samples, we also tested an alternative strategy in which we performed a data transformation, prior to feeding the data into the network-feature-based classifier (i.e. the random forest that was trained with the network-based features on the non-transformed data). To this end, the read-out of all 30 network features (mutations, CNVs, expression signals) were separately diffused for each test data through the selected drug-specific sub-network selected by our method. This transformation of the original data will allow signals in the selected sub-network that were observed in few samples to be extrapolated to neighboring nodes in the sub-network that

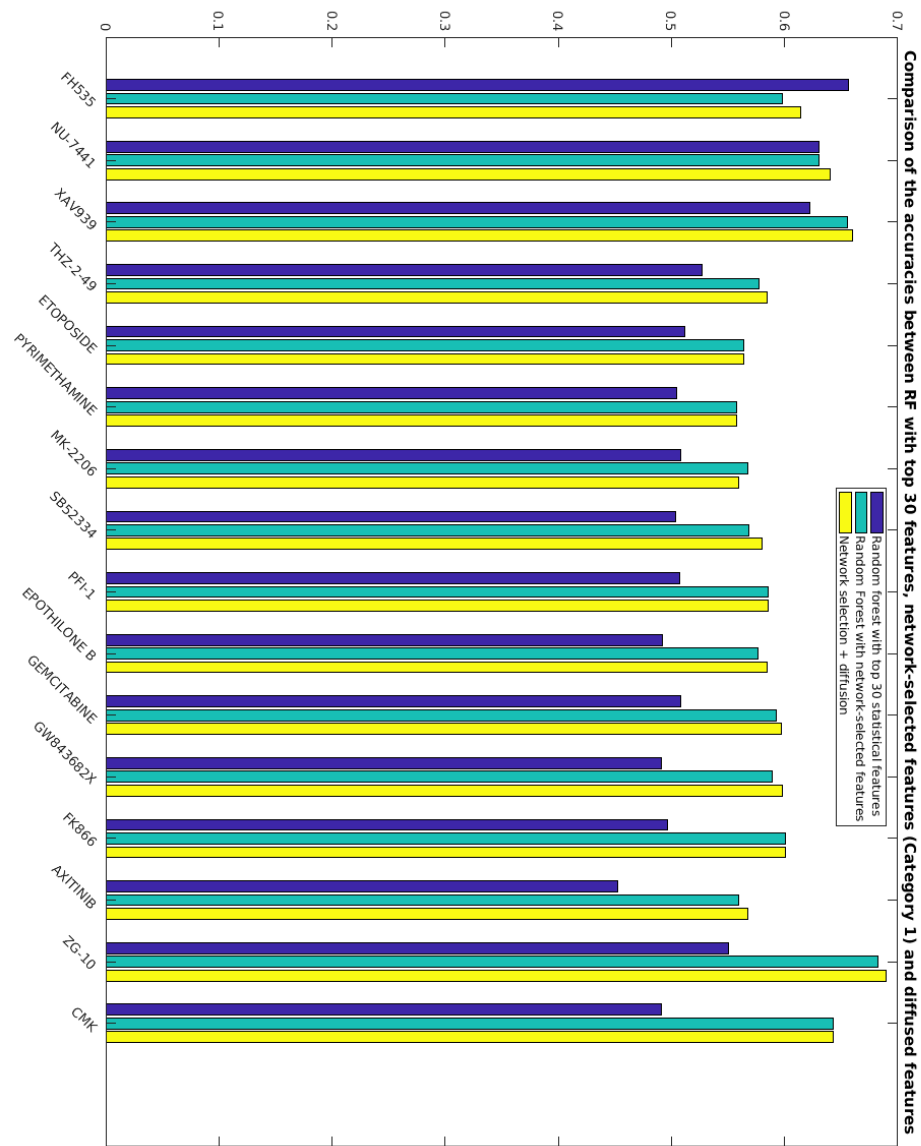


Figure 5.6: Comparison of the performance of random forest classifiers using the statistical feature selection (blue), random forest using network-based feature selection (green) and random forest using diffused data (yellow) in the case where no clear signal was present in the data but where the network feature selection method could still perform statistically better than random.

tend to be more frequently mutated and thus carry a more pronounced statistical signal. This transformation allows a frequency driven approach such as a network-feature based random forest to more easily recover these samples too. Comparing the results of this diffusion-based classification strategy with baseline and with the network-feature based random forest classifier applied to the non-transformed left out data shows the added value of exploiting the information contained in the full drug specific sub-network.

For the random forest classifier that uses network-based data transformation (using network-based diffusion, see Methods and Data), classification performances are generally higher than the baseline method and the method using the network-selected features. For the first subset of the data, network-based data transformation resulted a better classification performance than the baseline method on 126 out of 156 drugs, and better than the method using the network-selected features on 103 of 156 drugs (although the improvement in this case is generally quite small, Figure 5.3 and Figure 5.4). For the second subset of the data (58 drugs), network-based data transformation quite often gives a classification performance that is at least as high as using network-based feature selection (42 out of 58 drugs), although, similarly as in the previous case, any improvements over using the network-based feature selection are very marginal. This general observation that the improvement obtained using the network-based data transformation method is marginal compared to just using the the network-based feature selection method is probably due to the fact that the number of samples with rare mutations is too small compared to the bulk of the sample set, giving only advantages on very rare cases. However, in practice, such an approach could still be useful for potentially being able to suggest alternative treatment for cancer patients that have rare mutations.

5.2.4 Biological Relevance of the pathway-based features

To assess per drug the biological relevance of the top-ranked genes and their corresponding features, we assessed to what extent known drug-feature associations could be recovered by our method. Table 1 shows for 18 drugs that are either clinically approved or under a clinical trial, the features, known to be predictive for response (+) or non-response (-) of the cell lines carrying them. We compared to what extent the top scoring features selected by our network-based feature selection method and the most predictive features obtained after training the random forest with the network-based features used as input corresponded with those known drug response features. As is shown in Table 1, all known features were ranked highly by our network-based feature selection method and/or selected as the most prominent features after training the classifier. This shows that from the thousands of features used as input, our network based feature extraction method

Figure 5.7 List of known drug-feature associations, and the rankings of the features obtained using our network-based feature ranking method

Drug Name	Target	Pathway	Feature	Sign	Network score ranking	Predictor Ranking
AC220	FLT3	RTK signaling	FLT3-MUT	+	6 (MUT)	-
AFATINIB	ERBB2,	EGFR signaling	EGFR-AMP	+	1 (CNV)	9
AMG-706	VEGFR,	RTK signaling	PDGFRA-AMP	+	3(CNV)	-
BOSUTINIB	SRC, ABL,	ABL signaling	BCR-ABL-MUT	+	10 (MUT)	-
DABRAFENIB	BRAF	ERK MAPK signaling	BRAF-MUT	+	1 (MUT)	1
DASATINIB	ABL, SRC,	ABL signaling	BCR-ABL-MUT	+	5 (MUT)	-
GDC0941	PI3K (class	PI3K signaling	PIK3CA-MUT	+	1 (MUT)	1
GEFINITIB	EGFR	EGFR signaling	EGFR-MUT	+	6 (MUT)	11
GEFINITIB	EGFR	EGFR signaling	EGFR-AMP	+	2 (CNV))	2
GSK690693	AKT	PI3K signaling	PIK3CA-MUT	+	4 (MUT)	9
GSK690693	AKT	PI3K signaling	PTEN-MUT	+	2 (MUT)	2
MITOMYCIN	DNA	DNA replication	TP53-MUT	+	1 (NET)	5
NILOTINIB	ABL	ABL signaling	BCR-ABL-MUT	+	2 (MUT)	-
NUTLIN-3	MDM2	p53 pathway	TP53-MUT	-	2 (MUT))	1
OLAPARIB	PARP1,	Genome integrity	EWSR1-FLI1-	+	2 (MUT)	5
PD-0325901	MAP2K1 (MEK1)	ERK MAPK signaling	BRAF-MUT	+	2 (MUT)	1
PD-0325901	MAP2K1 (MEK1)	ERK MAPK signaling	KRAS-MUT	+	3 (MUT)	2
PD-0325901	MAP2K1 (MEK1)	ERK MAPK signaling	KRAS-MUT	+	5 (MUT)	3
PD-0332991	CDK4, CDK6	cell cycle	RB1-DEL	-	1 (CNV)	2
PLX4720	BRAF	ERK MAPK signaling	BRAF-MUT	+	1 (MUT)	1
TRAMETINIB	MAP2K1	ERK MAPK signaling	BRAF-MUT	+	1 (MUT)	1
TRAMETINIB	MAP2K1 (MEK1),	ERK MAPK signaling	KRAS-MUT	+	3 (MUT)	2
TRAMETINIB	MAP2K1 (MEK1),	ERK MAPK signaling	NRAS-MUT	+	5 (MUT)	3

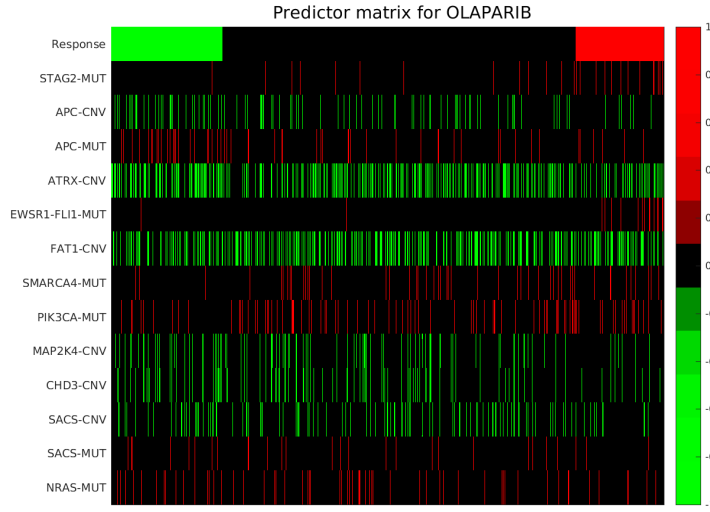
is able to prioritize the biologically relevant ones.

To assess how our selected features compare to those that were obtained by respectively ANOVA and LOBICO (a logical predictive model described in [10]) on the same data set, we compared the top selected features after random forest training to the features obtained using ANOVA or LOBICO. ANOVA finds on average 2.9 responses associated features per drug of which on average 0.8 features per drug were also retrieved by our method whereas LOBICO detected on average 3.7 predictive features per drug of which on average 0.8 features overlapped with those found by our method. In the comparison mentioned above we focused on features that were prioritized because of (1) their univariate association to response, as the network score ranking selects the most differentially scored nodes between the resistant and responsive set, (2) because of its predictive power, because the features were ranked and selected using their predictive importance in the random forest model. As these criteria are reminiscent of what univariate methods or predictive methods do, an overlap in features selected by respectively our method and ANOVO/LOBICO is to be expected. However part of the feature score of our network-based features is derived from its degree of connectedness

with other selected features, because the network scores calculation tends to give higher scores to connected nodes. Thus, our method represents a compromise between individual importance of features and their relationships within the biological network. Because of the latter network-component of the score, our network based method is able to also prioritize features that by themselves are only weakly-associated to response and/or have low predictive power (as is illustrated by the low predictive value), but that are relevant to the response because they trigger together with other features the same causal molecular network. In addition, rather than associating every single aberration separately to the phenotype as is done with classical regression methods, aberrations are collapsed on genes which are then associated to the phenotype through the network. Using the network thus provides an intuitive way to integrate different data sources: if different types of aberrations occur in the same gene that on their own occur too rarely to have any predictive power collapsing them on genes increases their association signal. This is an additional factor that contributes to the complementarity between our network-based method and the more classical association techniques. An example is for instance the PTEN mutation & deletion which were by our network feature selection method both found to be associated to TGX 221 response (PTEN-MUT is third in the mutation ranking, and PTEN-CNV is second in the CNV ranking) but for which in the original publication only the mutation was prioritized (by the ANOVA model). As PTEN is a known tumor suppressor gene that negatively regulates the PI3K-Akt pathway, mutations in PTEN have a well-known association with response to PI3K inhibitors, including TGX 221 [11–13]. The association between TGX 221 response and the loss of PTEN was recently confirmed [14].

As mentioned above, the main advantage of our integrative network-based feature extraction methods is that it also allows identifying features that are infrequent in the sample cohort but that associate with the phenotype through their connectivity with other more prominently present features. Such features are harder to identify with methods that rely on mutational recurrence such as LOBICO [10] or standard ANOVA or a mere random forest. We illustrate the biological relevance of some of these novel features with a few representative examples. We focused mainly on drugs for 1) which the identified features were not identified in the original publication by ANOVA or LOBICO, 2) for which our network-feature based classifier outperformed baseline. These are typically drugs for which predictive network-based features could be detected that do not occur frequent in the cohort (sometimes in a few samples only). A first example is Olaparib, an FDA-approved PARP inhibitor. Figure 5.8 shows the selected features extracted for Olaparib. Features were ranked according to their predictive power in the random forest trained with network-based features. Based on our analysis of the GDSC data we prioritized as one of the most predictive positive predictors for Olaparib response the

Figure 5.8 Top-ranking features selected using our method for OLAPARIB. Rows represent the selected features, whereas column represent cell lines, sorted from the most non-responsive to most responsive. Mutations and amplifications are shown as 1s, while deletions as -1s.



EWSR1-FLI1 fusion, which was also recovered by previous studies using ANOVA and LOBICO and which is a well-known biomarker for positive response to Olaparib [15].

However, next to EWSR1-FLI1 fusion we detected several features that occur much less abundantly in the sample set than the EWSR1-FLI1 fusions. For instance mutations in STAG2 and PIK3CA were both prioritized as features predictive for positive Olaparib response [16]. STAG2 involved in chromatid cohesion could, in analogy to other genes that cause sensitivity to PARP inhibition affect replication fork integrity and therefore homologous recombination repair. The involvement of STAG2 mutation cells in driving sensitivity to Olaparib was confirmed by a recent study in Glioblastoma [17]. Related to PIK3CA it has previously been shown that PI3K inhibition can lead to DNA damage, downregulation of BRCA1/2, gain in poly-ADP-ribosylation, and subsequent sensitization to PARP inhibition [18]. As predictors of a negative response to Olaparib, we prioritized APC deletions and mutations. We did not find any direct link in literature between Olaparib and APC mutations. However, APC plays a role in DNA repair, particularly in base excision repair (BER) pathway. In normal cells that have DNA damage above the threshold limit, the APC level increases and blocks BER leading to apoptosis. It is thus tempting to assume that in the absence of APC, cells can es-

cape apoptosis despite having a high load of DNA damage and subsequently might have a higher chance to become tumorigenic. Interestingly, APC was also prioritized as a negative response predictor for Camptothecin another drug interfering with DNA damage pathways. Camptothecin is an inhibitor of topoisomerase 1 inducing irreversible double-strand breaks which normally should induce apoptosis, unless APC mediated apoptosis is impaired.

5.3 Discussion

In this study we presented a network-based method to select molecular features associated with drug response in the context of cancer treatment. The method is motivated by the assumption that because of the evolutionary characteristics of cancer, two different cell lines with different molecular aberrations can exhibit a similar phenotype (e.g., drug response to the same drug) if their mutations affect the same pathway. By exploiting prior information in the form of molecular networks, this information on pathways that are recurrently affected across samples and associate with the phenotype of interest (i.e. drug response) is taken into account. This use of network-based prior information allows recovering rarely mutated features that have a low association signal/and or predictive power on their own. In addition using a network as a scaffold to perform the analysis provides a straightforward and intuitive way to cope with multiple data sets in an integrative way. By considering these aberrations in the context of a network (network neighborhood) relevant associations between infrequently occurring molecular features and drug response will be up-weighted because of their location in the network neighborhood of more prominently associating features. As a result our method can recover novel drug associated features on the GDSC [9] data set that were not yet described in the original study. The relevance and added value of network-based features was proven by assessing their predictive power in a cross validation setting. Using a network-based feature selection prior to training a random forest classifier increased the performance as compared to a classifier that used all features as input, especially for drugs in which the overall classification performance was low. In those setting no obvious frequent features are present as is testified by the low performance of the random forest trained on all features, and the added value of using a network to “up-weight” the rare features when they occur in each other’s network neighborhood becomes more pronounced. An additional added value of the network-based features is their interpretability. Whereas a random forest classifier often also results in a good predictive model, the extracted features only have a statistical interpretation. The network-based features in contrast provide insight into the molecular mechanisms that drive a drug response. This interpretability is, for instance, clearly illustrated by the fact that the same type of features were selected for drugs with similar modes of action e.g. the selection for

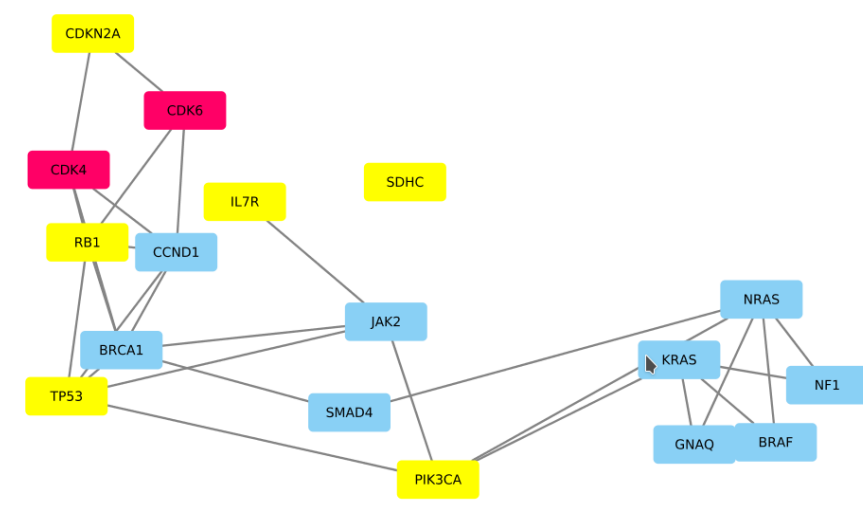
APC as a negative predictor for both Olaparib and Camptothecin.

Another example is the association of APC mutation to response to YM155, a BIRC5 (survivin) inhibitor. Previous studies have shown the regulatory role of APC to survivin, and the role of APC mutation in the development of cancer [14, 19]. Another study has confirmed relevance of APC mutation to YM155 response [20]. Yet another interesting example is the association between the response to MK 2206, and AKT inhibitor with PIK3CA mutations. PIK3CA and AKT are important actors involved in the PI3K-Akt pathway and recent studies have shown a clear association between the response to MK 2206 and PIK3CA mutations [21].

Yet another advantage of using our network-based approach is that ability to prioritize gene(s) among several genes co-amplified/deleted in a certain region. In the GDSC data set, for example, amplification and deletion are measured for regions that may span several different genes. Due to the fact that the genes in a single region are represented by one measurement, any association between the region and the drug response cannot distinguish between which genes in the region are actually more likely to be biologically-relevant in the association. Accordingly, all associations between copy number variations presented in [9] using either ANOVA or LOBICO are always at the region-level, and cannot select which of them are likely to have the causative association. Our network-based approach solves the problem of selecting the biologically relevant genes in the region by integrating the underlying biological network, which implicitly models our knowledge about biological processes/pathways that involve these genes, and that might be relevant in determining drug response. The network-derived scores that are computed per gene will then reflect the degree to which the genes are associated to the response, taking into account the network structure.

An example of this is our result in the feature prioritization for response to PD-0322991, a cyclin D kinase 4/6 inhibitor (Figure 5.9). It has been known that CDKN2A deletion and RB1 mutational status are correlated with response to PD-0322991 in several types of cancer, e.g., luminal ER+ breast cancer cells [32]. Despite the fact that CDKN2A deletion in the GDSC data set is always co-measured with BNC2, JAK2 and PSIP1, CDKN2A deletion were ranked first in both our network score ranking (CNVP) as well as predictor ranking, while none of these other 3 genes were selected, indicating that our method correctly prioritizes the biologically relevant feature among the genes in the same region. We can partially explain this result by examining the underlying sub-network structure connecting the top scoring genes, as depicted in Figure 5.9. From this network structure, we can conclude that CDKN2A receives a high score (1st in NET ranking) not only because it correlates to response, but also because it is closely connected to other high scoring nodes. In fact, we can see that it is also closely-connected to the nodes representing the other 4 features that are also highly-ranked, both by the network scores as well as by the random forest predictor importance measure: RB1 dele-

Figure 5.9 The sub-network selected by our feature selection method for cyclin-inhibitor drug PD-0332991. Yellow nodes are the nodes which are also with high predictive values according to the random forest classifier, while red nodes are the drug's targets.



tion, RB1 mutation, IL7R deletion and TP53 mutation. Coincidentally, all of these nodes are closely-connected as well to the drug targets: CDK4 and CDK6.

5.4 Methods and Data

5.4.1 Cell lines and data processing

Data were collected from the repository of the Genomics of Drug Sensitivity in Cancer (GDSC) [9]. Molecular data from the GDSC project was obtained from the binary Cancer Functional Events (CFEs) matrix for the Pan-Cancer analysis (file PANCAN_simple_MOBEM.rdata.txt), while gene expression was derived from the RMA-normalized expression matrix (file Cell_line_RMA_proc_basalExp.txt). Drug screening profiles were obtained from the AUC column of the fitted dose response (v17). Based on the molecular profiling data obtained from GDSC, we first calculate, for each tissue type, the average gene expression values. Then, per gene and per drug data set, a discrete value representing the level of differential expression of the gene in each of the cell lines w.r.t. the calculated tissue average were then derived. Cell lines are assigned to tissues as defined in the TISSUE2 field of the GDSC_Cells table. Values of 1, 0 or -1 represent whether a gene is respectively highly-expressed, averagely-expressed or lowly-expressed, as compared to the tissue average. Assigning 1 (or -1) to a gene is based on the

following criteria: (1) the gene's expression is higher (or lower) than 90% of that of the other cell lines of the same tissue, (2) the tissue average of the gene's expression is at least 5 and (3) the tissue standard deviation of the gene's expression is at least 2. These discrete differential gene expression levels are used instead of the continuously-valued RMA-normalized gene expressions, for both the network score computation and the random forest training. Per drug, we determined a pair of thresholds ($th_1 < th_2$) to distinguish between cell lines with a clear response signal (cell lines for which the Activity Area $AA > th_2$) and cell lines that clearly did not respond (cell lines for which $AA < th_1$). The thresholds per drug was determined as follows:

- The lower threshold th_1 is defined as the minimum between 0.1 and the 15% quantile of the whole response range for the drug.
- The upper threshold th_2 is defined as the maximum between 0.1 and the 85% quantile of the whole response range for the drug.
- If there are less than 30 samples for either the negative ($AA < th_1$) or the positive ($AA > th_2$), then the drug is discarded (no further analysis will be done).

Cell lines with an AA between th_1 and th_2 were not retained for network-based analysis and for training random forests, in order to avoid disturbing the training data with noisy signals. The molecular interaction network was obtained by combining interactions retrieved from 299 KEGG human pathways [5], from the Atlas of Cancer Signaling Network (ACSN) [7], and from the CCSB Human Interactome project (HI-II-14) [22]. Protein complexes and protein fusions (e.g., BCR-ABL), mostly occurring in KEGG pathways and in the ACSN components, are represented as a separate node connected to their constituent protein nodes. After some pre-filtering and removal of non-connected nodes, the resulting network after combining the three data set consists of around 4800 nodes and 35000 interactions. Dummy nodes were created to represent gene/protein fusions, (e.g., BCR-ABL) which are then connected to nodes representing the fused genes (e.g., BCR and ABL).

5.4.2 Network model

An overview of the network model is given in Figure 5.1. We describe how the network model is constructed below. Given a set of samples with molecular data (MUT, CNV, EXP), and a prior interaction network given by the adjacency matrix A , we perform the following steps:

- Create a node for every sample (called the sample node)

- Create a node representing every mutation, copy number variation and every (non-zero valued) differential gene expression (called the status node) on each gene node in the interaction network, and connect them to their corresponding gene in the interaction network. If an event (say, mutation) for a certain gene is never observed in any sample, then the corresponding the status node is not needed.
- Connect each sample node to their corresponding status nodes representing the occurrences of each feature in the cell lines. Each connection from a sample node i to any data type node is weighted with the value of

$$W_i = \frac{AA_i - \min_{j \in S}(AA_j)}{\max_{j \in S}(AA_j) - \min_{j \in S}(AA_j)}$$

where AA_i is the Activity Area of the cell line in the sample node i if S is the responsive set, and $AA_i = 1 - AA$ if S is the non-responsive set.

5.4.3 Calculating the similarity scores

Subsequently, for each set (resistant and responsive), we perform the following: Similarity scores were based on a Laplacian Exponential Diffusion kernel [23, 24], which has shown to be useful in some context of network based cancer driver identification:

$$\begin{aligned} K &= \exp -\alpha L \\ L &= D - A \\ D(i, i) &= \sum_{j=1}^n A(i, j) \end{aligned}$$

where A is the adjacency matrix of the global network, D is the associated diagonal degree matrix for A , and L is called the Laplacian matrix of the network. The free parameter α was chosen to be equal to 0.001 by trials. The resulting K is a matrix of scores with the same dimensions as A . Kernel normalization was performed as described in [4]. From the kernel score matrix K obtained for a set S , we derive 3 types of scores: mutation scores (Ms), copy number variation scores (Cs) and network entity scores (Es) each corresponding to a specific node type: status nodes (Ms , Cs) or network entity nodes (Es). MS is computed by calculating the average of kernel scores between the sample nodes and the mutation nodes over all sample nodes in set S (i.e., Ms is a vector of length k , where k is the number of mutation node in the set S). Similarly, Cs scores are calculated using the average scores between the sample nodes to their corresponding data type nodes.

5.4.4 Assigning node ranks

For each node type the scores (respectively Ms , Cs and Es) obtained in respectively the responsive and the non-responsive set will be compared to identify differentially scoring nodes. However scores cannot directly be compared between both two sets as the score distributions might differ per set. This is because the values of the kernel scores are affected by the number of available sample nodes in each set. Therefore, scores obtained from either set were quantile-normalized prior to the comparison [25]. Nodes were per node type ranked based on their quantile-normalized scores (respectively Ms , Cs , and Es). From the mutation scores MP and MN corresponding to respectively the responsive and non-responsive sets, the vector is computed. We then select 10 top-ranking positive and negative features based on the scores in M , C and E . Expression scores, copy number variation and entity scores were analyzed similarly. In total, this resulted in 3 different ranking/selections: mutations (MUT), copy number variations (CNV), and the network ranking scores (NET), for a total of 30 features selected. For the MUT and CNV rankings, selected features are immediately used in the next steps (training and classification), whereas the rankings from the entity nodes (NET) is interpreted to mean that both the mutation and the copy number variation features of those nodes (if available) are relevant and selected.

5.4.5 Training and testing the classification model

Using the selected features (i.e., we then train a 50-tree random forest model trained using the AdaBoost bagging method [26]. Each tree was limited to a maximum 5 node splits and a minimum of 10 samples in each leaf node. Before testing the trained random forest model on a new data, we can either perform (or not) a data normalization step using a network diffusion method. In this case, we use a PageRank-like [8] network diffusion over the interaction network on each data type (MUT, CNV) on the features selected by our previously-described network-based feature selection method. Such an approach has been used before in the context of cancer data analysis, e.g., in [27, 28]. In short, we map the molecular profile on the selected features of a sample onto the network, and then diffuse it using the following iterative equation until it converges:

$$F_{t+1} = \alpha F_t A_n + (1 - \alpha) F_0$$

where F_0 is the original profile, A_n is the degree-normalized adjacency matrix, and α is a free-parameter. The value for the parameter α were determined by using a cross-validation parameter tuning independently in each drug (i.e., the optimal values may differ between drugs).

For presentation purposes, we split the cross-validation results into two subsets: 1) the subset (156 out of the 214 drugs) in which a clear statistical signal is

present that allows a feature extraction and classification performance that is larger than what can be expected at random and 2) the data set in which such statistical signal is not obviously present (58 out of the 214 drugs). The criteria by which we determined these two classes is by assessing whether the a random forest classifier trained using the original data performed statistically better than a random forest trained using randomly shuffled data(at $p\text{-value} < 0.05$).

This process is illustrated in Figure 5.10. First, we generate the random background distribution of classification performances by randomly shuffling the features and then train a random forest classifier on the shuffled data, and repeat the process 100 times. We then compute the classification performances of random forest classifiers trained using all data (red dashed line), using top 30 statistically significant features (blue dashed line), using network-based feature selection, either with the network-based data transformation (yellow dashed line) or without (green dashed line). A drug falls into the first subset if the right-tailed p -value of the classification performance of the random forest using all features is less than 0.05. Otherwise, the drug falls into the second subset of the data. We then compare the performances of the other three random forest classifiers to produce the bar plots in Figure 5.3, 5.4 and 5.6.

5.4.6 Implementation of the method

Implementation of the method and as well as the cross-validation procedure are available online at <http://github.com/mushthofa/> in Matlab for easy reproduction. All genomic, transcriptomic and drug response data can be obtained from the GDSC project [9] webpage <https://www.cancerrxgene.org/>.

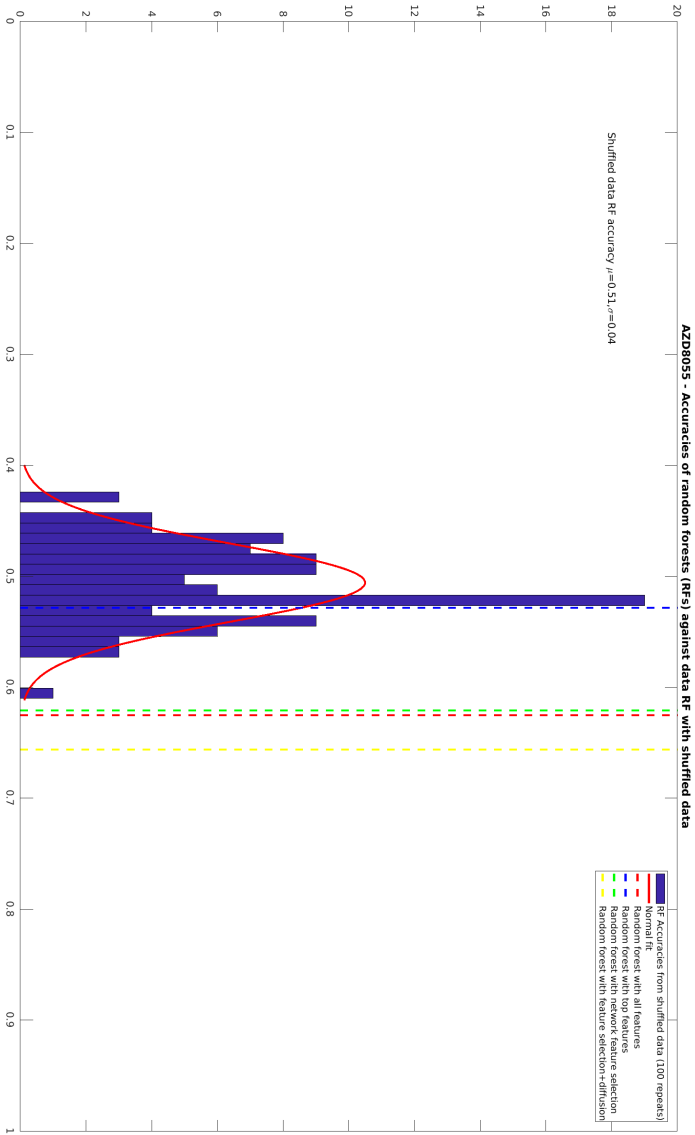


Figure 5.10: Classification performances of random forest using all features (red), top 30 statistically-selected features (blue), network-selected features (green) and using network-based data transformation/diffusion (yellow), compared to the background distribution of random forest performances using randomly shuffled features on drug AZD8055.

References

- [1] T Hedley Carr, Robert McEwen, Brian Dougherty, Justin H Johnson, Jonathan R Dry, Zhongwu Lai, Zara Ghazoui, Naomi M Laing, Darren R Hodgson, Francisco Cruzalegui, et al. *Defining actionable mutations for oncology therapeutic development*. Nature Reviews Cancer, 16(5):319, 2016.
- [2] Chan Shen, Funda Meric-Bernstam, Xiaoping Su, John Mendelsohn, and Sharon Giordano. *Prevalence of actionable mutations and copy number alterations and the price of a genomic testing panel*. Oncotarget, 7(44):71686, 2016.
- [3] Genevieve M Boland, Sarina A Piha-Paul, Vivek Subbiah, Mark Routbort, Shelley M Herbrich, Keith Baggerly, Keyur P Patel, Lauren Brusco, Chacha Horombe, Aung Naing, et al. *Clinical next generation sequencing to identify actionable aberrations in a phase I program*. Oncotarget, 6(24):20099, 2015.
- [4] Lieven PC Verbeke, Jimmy Van den Eynden, Ana Carolina Fierro, Piet Demeester, Jan Fostier, and Kathleen Marchal. *Pathway relevance ranking for tumor samples through network-based data integration*. PloS one, 10(7):e0133503, 2015.
- [5] Minoru Kanehisa and Susumu Goto. *KEGG: kyoto encyclopedia of genes and genomes*. Nucleic acids research, 28(1):27–30, 2000.
- [6] G Joshi-Tope, Marc Gillespie, Imre Vastrik, Peter D’Eustachio, Esther Schmidt, Bernard de Bono, Bijay Jassal, GR Gopinath, GR Wu, Lisa Matthews, et al. *Reactome: a knowledgebase of biological pathways*. Nucleic acids research, 33(suppl_1):D428–D432, 2005.
- [7] I Kuperstein, E Bonnet, HA Nguyen, D Cohen, E Viara, L Grieco, S Fourquet, L Calzone, C Russo, M Kondratova, et al. *Atlas of Cancer Signalling Network: a systems biology resource for integrative analysis of cancer data with Google Maps*. Oncogenesis, 4(7):e160, 2015.
- [8] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. *The PageRank citation ranking: Bringing order to the web*. Technical report, Stanford InfoLab, 1999.
- [9] Francesco Iorio, Theo A Knijnenburg, Daniel J Vis, Graham R Bignell, Michael P Menden, Michael Schubert, Nanne Aben, Emanuel Gonçalves, Syd Barthorpe, Howard Lightfoot, et al. *A landscape of pharmacogenomic interactions in cancer*. Cell, 166(3):740–754, 2016.

- [10] Theo A. Knijnenburg, Gunnar W. Klau, Francesco Iorio, Mathew J. Garnett, Ultan McDermott, Ilya Shmulevich, and Lodewyk F. A. Wessels. *Logic models to predict continuous outputs based on binary inputs with an application to personalized cancer therapy*. Scientific Reports, 6:36812+, November 2016.
- [11] Chenchen Feng, Yang Sun, Guanxiong Ding, Zhong Wu, Haowen Jiang, Lujia Wang, Qiang Ding, and Hui Wen. *PI3K β inhibitor TGX221 selectively inhibits renal cell carcinoma cells with both VHL and SETD2 mutations and links multiple pathways*. Scientific reports, 5:9465, 2015.
- [12] Peter R Shepherd and William A Denny. *Beta-testing of PI3-kinase inhibitors: is beta better?* Cancer discovery, 2(5):393–394, 2012.
- [13] Hongbo Chen, Lin Mei, Lanzhen Zhou, Xiaomeng Shen, Caiping Guo, Yi Zheng, Huijun Zhu, Yongqiang Zhu, and Laiqiang Huang. *PTEN restoration and PIK3CB knockdown synergistically suppress glioblastoma growth in vitro and in xenografts*. Journal of neuro-oncology, 104(1):155–167, 2011.
- [14] Tao Zhang, Tomas Otevrel, Zhenqiang Gao, Zhiping Gao, Sandra M Ehrlich, Jeremy Z Fields, and Bruce M Boman. *Evidence that APC regulates survivin expression: a possible mechanism contributing to the stem cell origin of colon cancer*. Cancer research, 61(24):8664–8667, 2001.
- [15] J Chad Brenner, Felix Y Feng, Sumin Han, Sonam Patel, Siddharth V Goyal, Laura M Bou-Maroun, Meilan Liu, Robert J Lonigro, John R Prensner, Scott A Tomlins, et al. *PARP-1 inhibition as a targeted strategy to treat Ewing’s sarcoma*. Cancer research, pages canres–3648, 2012.
- [16] Louis Chesler, Chris Schlieve, David D Goldenberg, Anna Kenney, Grace Kim, Alex McMillan, Katherine K Matthay, David Rowitch, and William A Weiss. *Inhibition of phosphatidylinositol 3-kinase destabilizes Mycn protein and blocks malignant progression in neuroblastoma*. Cancer research, 66(16):8139–8146, 2006.
- [17] Melanie L Bailey, Nigel J O’Neil, Derek M van Pel, David A Solomon, Todd Waldman, and Philip Hieter. *Glioblastoma cells containing mutations in the cohesin component STAG2 are sensitive to PARP inhibition*. Molecular cancer therapeutics, 13(3):724–732, 2014.
- [18] Yasir H Ibrahim, Celina García-García, Violeta Serra, Lei He, Kristine Torres-Lockhart, Aleix Prat, Pilar Anton, Patricia Cozar, Marta Guzmán, Judit Grueso, et al. *PI3K inhibition impairs BRCA1/2 expression and sensitizes BRCA-proficient triple-negative breast cancer to PARP inhibition*. Cancer discovery, 2(11):1036–1047, 2012.

- [19] Tao Zhang, Jeremy Z Fields, Lynn Opdenaker, Tomas Otevrel, Emi Masuda, Juan P Palazzo, Gerald A Isenberg, Scott D Goldstein, Marc Brand, and Bruce M Boman. *Survivin-induced Aurora-B kinase activation: A mechanism by which APC mutations contribute to increased mitoses during colon cancer development*. The American journal of pathology, 177(6):2816–2826, 2010.
- [20] Omid Fotouhi, Hanna Kjellin, Catharina Larsson, Jamileh Hashemi, Jorge Barriuso, C Christofer Juhlin, Ming Lu, Anders Höög, Laura G Pastroán, Angela Lamarca, et al. *Proteomics suggests a role for APC-survivin in response to somatostatin analog treatment of neuroendocrine tumors*. The Journal of Clinical Endocrinology & Metabolism, 101(10):3616–3627, 2016.
- [21] Nikia A Laurie, Stacy L Donovan, Chie-Schin Shih, Jiakun Zhang, Nicholas Mills, Christine Fuller, Amina Teunisse, Suzanne Lam, Yolande Ramos, Adithi Mohan, et al. *Inactivation of the p53 pathway in retinoblastoma*. Nature, 444(7115):61, 2006.
- [22] Thomas Rolland, Murat Taşan, Benoit Charlotiaux, Samuel J Pevzner, Quan Zhong, Nidhi Sahni, Song Yi, Irma Lemmens, Celia Fontanillo, Roberto Mosca, et al. *A proteome-scale map of the human interactome network*. Cell, 159(5):1212–1226, 2014.
- [23] François Fouss, Kevin Francoisse, Luh Yen, Alain Pirotte, and Marco Saerens. *An experimental investigation of kernels on graphs for collaborative recommendation and semisupervised classification*. Neural networks, 31:53–72, 2012.
- [24] Risi Imre Kondor and John Lafferty. *Diffusion kernels on graphs and other discrete input spaces*. In ICML, volume 2, pages 315–322, 2002.
- [25] Benjamin M Bolstad, Rafael A Irizarry, Magnus Åstrand, and Terence P. Speed. *A comparison of normalization methods for high density oligonucleotide array data based on variance and bias*. Bioinformatics, 19(2):185–193, 2003.
- [26] Yoav Freund and Robert E Schapire. *A decision-theoretic generalization of on-line learning and an application to boosting*. Journal of computer and system sciences, 55(1):119–139, 1997.
- [27] Matan Hofree, John P Shen, Hannah Carter, Andrew Gross, and Trey Ideker. *Network-based stratification of tumor mutations*. Nature methods, 10(11):1108, 2013.

-
- [28] Thanh Le Van, Matthijs van Leeuwen, Ana Carolina Fierro, Dries De Maeyer, Jimmy Van den Eynden, Lieven Verbeke, Luc De Raedt, Kathleen Marchal, and Siegfried Nijssen. *Simultaneous discovery of cancer subtypes and subtype features by molecular data integration*. *Bioinformatics*, 32(17):i445–i454, 2016.

6

Conclusion

This thesis is the culmination of 4 years of research within the context of the application of computer science to computational biology. The work in first part of the thesis answers the first problem statement as described in Chapter 1. First, we started by considering one of the commonly used logical/discrete modelling frameworks for understanding the behaviour of gene regulatory networks, the so-called Boolean network. In Chapter 2, we investigated the use of Answer Set Programming (ASP) as the framework used for implementing the simulation of Boolean networks in a flexible and generic way. In this context, we developed ASP-G, a framework in ASP to simulate the trajectory of the states of the network, as well as compute its attractors. We tested the correctness of the tool implemented, and also assessed its computational resource requirements and limitations, one of which is the limitation of having only a binary level of activations (“on” or “off”).

We therefore set out to lift this limitation by investigating the use of Fuzzy ASP as the underlying framework. Since FASP is not yet as mature as ASP w.r.t. in terms of solver implementation, we investigated ways to develop a full-fledged solver for FASP that is capable of executing the encoding needed to represent the simulation of multi-valued networks. In Chapter 3, we described our proposal for the method to evaluate FASP programs by using a translation scheme into ASP, as well as our implementation for this method. We performed a benchmark assessment for our FASP solver and noted its limitations. The solver works correctly and with reasonable improvements in performance compared to an existing solver, while also managing to solve a larger class of programs.

We then proceeded to describe our encoding of multi-valued networks in FASP,

allowing the simulation of GRNs with multi-level activation values and the computation of their attractors in a declarative and generic way. We implemented this method in a new tool called FASPG, and assessed its correctness and performance, both on real biological networks as well as synthetic networks, with the take-away message being that the method performs reasonably well for the size of the networks generally considered for these types of problems (around 30-50 genes), especially for the computation of steady-states attractors, but for other types of problems, performance still needs to be improved.

The second part of the thesis proposes a solution to the second problem statement described in Chapter 1. Here, we turned our attention to the problem of incorporating biological network information in a predictive model that is aimed at increasing the robustness of the model against noise, small sample size in the data and also selecting relevant predictive features (“biomarkers”) associated to the observed phenotype. In particular, we focused on the problem of determining the features relevant for predicting cancer drug response. We built upon an existing network-based omics data integration method, adapted it to our particular goal (feature selection and prediction), applied it to an existing public dataset of cancer cell lines’ drug response. The resulting selected features often form a tight sub-network that readily hints at the mode of action of the particular drug. We also proposed a data transformation technique that takes advantage of this selected sub-network in order to increase prediction performance, which was indeed observed in many cases.

6.1 Future Perspective

In this section, we highlight some of the limitations and possible further improvements that can be pursued for further research, as well as some outlook on the importance and role that the methods we proposed in this thesis might hold in the future. With regards to the discrete/logical modelling of gene regulatory network tools we developed, some aspects can still be improved. For example, w.r.t. the computational performance, FASPG does not yet perform very well for computing cyclic attractors in the synchronized update scheme. This might be due to the nature of synchronizing updates, which exerts stronger constraints on the model, which in turn typically means that a combinatorial search on the solutions can require more steps than on problems with less strong constraints. This can be done by, e.g., investigating ways to prune the search space for special conditions, or using a more compact and efficient representation for the encoding. Another aspect that might be improved is the user-friendliness of the system. In particular, the input-output might be made more user friendly by providing e.g., a graphical/web-based interface that can cater to a wider user-base, and at the same time, graphical outputs on the simulations of the network might provide better visualizations that can aid in

finding insights about the behaviour of the biological system.

With respect to our feature selection method, we recognize that the model does not currently take into account two types of information about the network: direction and weights. Directionality is a natural component in many biological networks: a gene may regulate another gene, but the other gene may not regulate the former, indicating a one-directional flow of information/causality. A model that includes such directionality information may be able to perform better than our current method. Unfortunately, using the graph kernel diffusion method described in our method, directionality information in the graph is discarded, and we could not find yet an appropriate graph kernel diffusion method that can make use of the directionality within the context of our method. We also have not used any weighted connections between the gene nodes in the underlying interaction network (except between the sample node to gene entity nodes to simulate the strength of drug response), despite the fact that graph kernel methods can easily incorporate weights in the connections, and that many sources of biological information can be used to derive such weights (e.g., confidence values of the interactions typically available in biological datasets). However, as of now, we could not find a consistent weighting scheme across the whole network and different types of drug that will result in an acceptable performance. We therefore settled on using a fixed weight in all the nodes. Finally, since one of the goals of the feature selection method is to improve prediction accuracy, a more tight integration with a classification/prediction model would be ideal. Presently, we simply used the feature selection method to filter the top-ranking features, and which can then be fed into a classification/regression model, e.g., random forests. As such, the training step of the models can discard the important features due to the insignificant predictive power of the features (which may be due to small sample size and/or noise). The data transformation method using diffusion we performed is a step towards this direction, however since it is still performed before the training step itself, important features may still be discarded. A classification model that can natively take into account the network information and the ranking of these features on its training step may well perform better than a standard one such as the random forests we used here.

In a longer term, it is worth investigating how we can combine the two approaches investigated in this theses. Logic-based formalisms have been considered to be used for predictive modelling within the context of computational biology. For example, methods such as LOBICO [1] attempt to formalize the associations between biological predictors (e.g., mutations / CNVs) and phenotypic behavior (e.g., drug response). The use of a logical representation can allow for a more flexible conditions of associations between predictors and the predicted behavior, compared to classical classifier such as random forests. This can further be improved by the use of FASP as an expressive fuzzy-logic-based computational framework

that can increase the flexibility of model, e.g., by allowing multiple-levels of gene expressions or copy number alterations, as well as multiple-levels of response, into the model. Another advantage of using an expressive logic-based formalisms to model associations between biological predictors and phenotypic behaviors is that the extracted model can be more “transparent” and easily understandable (for biologists/medical scientists), which could be an important characteristics of a predictive model within the context of biology. In the future, predictive models that are not only perform well in terms of predictive accuracy, but also capable of working as a “white-box”, easily understandable by non-computer-scientists, may play an important role in biology/biomedical sciences, since they can also facilitate an easy “sanity-check” by biologists/medical scientists to see if the model can derive already known associations.

References

- [1] Theo A. Knijnenburg, Gunnar W. Klau, Francesco Iorio, Mathew J. Garnett, Ultan McDermott, Ilya Shmulevich, and Lodewyk F. A. Wessels. *Logic models to predict continuous outputs based on binary inputs with an application to personalized cancer therapy*. Scientific Reports, 6:36812+, November 2016.



List of Publications

Papers in international journals

1. Mushthofa Mushthofa, Gustavo Torres, Yves Van de Peer, Kathleen Marchal, and Martine De Cock. ASP-G: an ASP-based method for finding attractors in genetic regulatory networks. *Bioinformatics*, 30(21):3086, 2014
2. Mushthofa Mushthofa, Steven Schockaert, Ling-Hong Hung, Kathleen Marchal, and Martine De Cock. Modeling multi-valued biological interaction networks using fuzzy answer set programming. *Fuzzy Sets and Systems*, 2018

Papers presented in international conferences & published in proceedings

1. Mushthofa Mushthofa, Steven Schockaert, and Martine De Cock. A finite-valued solver for disjunctive fuzzy answer set programs. In *Proceedings of European Conference in Artificial Intelligence 2014*, pages 645–650, 2014
2. Mushthofa Mushthofa, Steven Schockaert, and Martine De Cock. Solving disjunctive fuzzy answer set programs. In *Proceedings of the 13th International Conference on Logic Programming and Non-monotonic Reasoning*, pages 453–466, 2015

3. Mushthofa Mushthofa, Steven Schockaert, and Martine De Cock. Computing attractors of multi-valued gene regulatory networks using fuzzy answer set programming. In *Proceedings of the 2016 IEEE World Congress on Computational Intelligence*, 2016

Posters presented in international conferences

1. Mushthofa Mushthofa, Lieven Verbeke, and Kathleen Marchal. Network-based method for feature selection and prediction of cancer drug response. In *Poster session of the 2017 Conference on Intelligent Systems for Molecular Biology*, 2017

Book chapter

1. Mushthofa Mushthofa, Steven Schockaert, and Martine De Cock. Fuzzy answer set programming: from theory to practice. In *Beyond Traditional Probabilistic Data Processing Techniques: Interval, Fuzzy, etc. Methods and Their Applications*. Springer, 2018