Analysis and visualization of signal execution in network-driven biological processes

By

Oscar Orlando Ortega Sandoval

Dissertation
Submitted to the Faculty of the
Graduate School of Vanderbilt University
in partial fulfillment of the requirements
for the degree of

DOCTOR OF PHYLOSOPHY
in

Chemical and Physical Biology
June 30, 2021
Nashville, Tennessee

Approved:

To the memory of the 6402 innocent Colombians

ACKNOWLEDGEMENTS

Throughout my Ph.D. journey, I have received support and help from many people. I would first like to thank my advisor, Dr. Carlos F. Lopez, who guided me during my PhD studies and provided me with all the tools necessary to be successful. Dr. Lopez's knowledge and insights were invaluable for developing the methods and analysis described in this thesis. I also want to thank Dr. Lopez for his patience and wisdom during the times when it seemed like I was not moving forward. I would also like to thank my PhD committee members, Dr. Alissa Weaver, Dr. Vivian Gama, Dr. Christopher Fonnesbeck. and Dr. William Holmes, for giving me different perspectives about my projects, providing insights about the biology and the models I worked with, and helping me identify potential pitfalls in my research. I want to thank Dr. Gama for her generosity and for letting me work in her lab to gain experience doing experiments.

I would also like to thank my lab mates as they were always there to help me. I would like to express my sincere gratitude to Blake Wilson, Leonard Harris, and Alex Lubbock for teaching me modeling, coding, and writing skills.

I would like to thank the Vanderbilt International Scholar Program that provided me funding and a community of international students at the beginning of my graduate studies. I would also like to thank the Department of Biochemistry at Vanderbilt for funding my research. I would like to thank the Office of Biomedical Research Education & Training for all the training and information that they provided me, which helped me get multiple job offers.

I would also like to thank my girlfriend Andrea Cuentas and our cat Pola for all of our adventures, for supporting and motivating me, and for being a source of happiness.

Finally, I want to thank my parents, Orlando Ortega and Nancy Sandoval, and my siblings Diego Ortega, Felipe Ortega, and Laura Ortega for their unconditional support, inspiration, and motivation to always move forward.

TABLE OF CONTENTS

DEDICATION .... **ii**

ACKNOWLEDGEMENTS .... **iii**

LIST OF TABLES .... **vii**

LIST OF FIGURES .... **viii**

Chapters

1. Introduction **1**

1.1. Introduction to ODE models and calibration 2
1.2. Defining a cost function for calibration 4
1.3. Types of calibration and quantification of parameter uncertainty 5
1.4. Introduction to dynamic flux analysis 6
1.5. Introduction to network visualization 7
1.6. Reproducibility of systems biology models 9
1.7. Organization 10

2. Development and analysis of the JNK3 Activation Reaction Model (JARM) **11**

2.1. Introduction: Biological background and model scope 11
2.2. Materials and methods 16
2.2.1. Experimental data 16
2.2.2. Model implementation and calibration
2.3. Results 21

3. PyDyNo: a tool to analyze parameter uncertainty in biochemical models **25**

3.1. Summary 25
3.2. Introduction 26

3.3. Materials and methods 28
3.3.1. Development on the python ecosystem 28
3.3.2. Workflow to obtain a discretized representation of network dynamics 28
3.3.3. Obtaining dominant subnetworks 29
3.3.4. Obtaining modes of signal execution 34
3.3.5. Sequence analysis 35
3.4. Results and discussion 35

4. Probability-based mechanisms in biological networks with parameter uncertainty 39
4.1. Summary 39
4.2. Introduction 39
4.3. Materials and methods 41
4.3.1. Mathematical models of apoptosis 42
4.3.2. Experimental data 42
4.3.3. Bayesian inference and parameter calibration 42
4.3.4. Analysis of signal execution 42
4.4. Results 43
4.4.1. Bayesian parameter optimization yields indistinguishable protein concentration dynamics 43
4.4.2. A discretized flux-based analysis of signal execution in networks 46
4.4.3. Key execution modes emerge despite parameter uncertainty 49
4.4.4. Signal execution modes respond differently to eCaspase perturbation 53
4.4.5. Reducing execution mode uncertainty through parameter measurements 54
4.4.6. Modes of signal execution in a detailed apoptosis models with increased parameter uncertainty 56
4.5. Discussion 65
4.5.1. Availability 66

5. Interactive multiresolution visualization of cellular network processes 67
5.1. Summary 67

5.2. Introduction 67
5.3. Materials and methods 69
5.3.1. Overview 69
5.3.2. Input data 70
5.3.3. Output data 70
5.3.4. PyViPR main visualization functions 70
5.3.5. Model calibration 71
5.3.6. Parameter selection for analysis 72
5.4. Results 72
5.4.1. PyViPR overview 72
5.4.2. Design choices for PyViPR 74
5.4.3. Network creation from multiple model components 75
5.4.4. Dynamic visualization in PyViPR 78
5.4.5. Exploration of a biological process with PyViPR: apoptosis execution 79
5.4.6. Multiresolution visualization and exploration of EARM 80
5.4.7. Parameter sets fit experimental data but yield different network dynamics 85
5.5. Discussion 91
5.5.1. Limitations of the study 93

6. Discussion and future directions 94
6.1. Advances in model reproducibility 94
6.2. Advances in model visualization 88
6.3. Advances in the quantification of parameter uncertainty 88

References 100

LIST OF TABLES

Table **Page**

2.1 JARM species and their corresponding initial conditions. The first u in JNK3 corresponds to the threonine site, and the second u is the tyrosine site............................13
2.2 Reactions defined in JARM. The left column shows the bidirectional and catalytic reactions that are generated from PySB rules. The right column shows the kinetic parameters involved in each reaction. Experimentally measured kinetic constants are highlighted in red. 15
5.1 Summary of the Biological Functions Enclosed in Each Community. 84

LIST OF FIGURES

Figure **Page**

2.1 Network of all possible interactions between MKK4, MKK7, JNK3, and arrestin-3 .. 12

2.2 Simulated trajectories of p(Thr)JNK3, p(Thr)JNK3 and doubly phosphorylated JNK3 calibrated to reproduce the experimental data 19

2.3 Posterior probability distributions for calibrated JARM kinetic parameters…......... 20

2.4 Top panel: corresponds to the time-dependent concentration changes of active (doubly phosphorylated) JNK3, pTyr-JNK3 bound to arrestin-3 and pThr-JNK3 bound to arrestin-3. Center panel: represents the catalysis reaction rate values of singly phosphorylated JNK3. Bottom panel: shows the catalysis reaction rate values of active (doubly phosphorylated) JNK3 22

2.5 Top panel: Simulated model trajectories of doubly phosphorylated JNK3. The red and black lines show the production of ppJNK3 when arrestin-3 is present and absent, respectively. Bottom panel: Simulated effect of varying arrestin-3 concentration in the system. 24

3.1 Algorithm to obtain dynamic fingerprints from a simulation 30

3.2 Modes of signal execution in JARM..37

3.3 Amounts of activated JNK3 are markedly different for each execution mode after the 50% MKK7 knockdown...38

4.1 Abridged Extrinsic Apoptosis Reaction (aEARM) network and parameter calibration results...45

4.2 PyDyno workflow. 48

4.3 Modes of signal execution in aEARM...52

4.4 Time of death responses are markedly different for the same perturbation..............55

4.5 Parameter measurements reduce execution mode uncertainty 58

4.6 Modes of signal execution in the full Extrinsic Apoptosis Reaction Network 60

4.7 Upper panel: Time to death distributions in the execution modes 1 and 2 in the “wild type” condition, after a 50% Mcl1 knockdown, and after a 50% knockdown of Bcl2. The boxplot inside the distributions shows the median, first quartile and third quartile of the datasets. Execution modes 1 and 2 show substantial differences in their response to the knockdowns. Lower panel: Time to death distributions in the execution mode 1 after adding a drug that binds XIAP at t=8000 s, and at t=4000, and a drug that binds XIAP and Mcl1 at 4000 s. B) List of the 10 parameters in EARMv2.0 that contribute the most to model prediction. 64
5.1 PyViPR Visual Encodings. (A) Node types used for visualizations as labeled 76
5.2 Multiresolution Visualization of a Reaction Network 81
5.3 Dynamic Visualization of EARM at Different Resolutions 87
5.4 In Silico Knock outs of Bcl-2 Proteins Modulates PARP Cleavage 90

Chapter 1

Introduction

Biological systems have been studied from a reductionist perspective since the early days of molecular biology and have yielded important insights (Morange, 1998). This reductionist approach consists in breaking a large system into different parts and identifying the connections between these parts. All with the assumption that molecule structures and their interactions provide enough explanation to understand the whole system. Apoptosis, an important pathway dysregulated in multiple human diseases (Singh et al., 2019), and JNK3 activation cascade, a pathway that can trigger apoptosis through p53 (Dhanasekaran & Reddy, 2008), are two examples of biological processes that have been studied through the reductionist approach leading to the identification of key protein-protein interactions. However, it has been increasingly clear that only knowing the interaction network of biological processes is not enough to understand and modulate these processes (Barzel & Barabási, 2013). This has led to the use of the systems biology approach, where mathematical models are employed to study the complex dynamics that arise during cellular signaling processes.

The systems biology approach consists of studying the interactions between multiple molecular components and their dynamics to understand the structure of a system and the emergent properties that arise as a result of the interactions (Kitano, 2002). According to Kitano, to understand a biological system it is necessary to study four key properties: 1) System structures: the network of protein interactions generated by extensive research in molecular biology, 2) System dynamics: the changes in temporal concentration of biomolecules, which are studied through mathematical equations 3) The control method: the mechanisms used to modulate cell states, and 4) The design method: approaches used to design biological systems with desired properties. In this work, we focused on understanding the dynamics of the apoptosis and JNK3 activation signaling processes and developed novel tools to analyze signal flow through networks and visualize network structures and dynamics.

To study the dynamical properties of the Apoptosis and JNK3 models we used the rule-based modeling framework PySB (Lopez et al., 2013). This framework encodes the protein-protein interactions and their dynamics into a network of ordinary differential equations (ODEs) using the law of mass action kinetics (Voit et al., 2015). These ODEs contain parameters describing the strength of reactions that are often unknown. To make useful predictions with these models it is necessary to calibrate the parameters and ensure that models reproduce prior experimental data. After calibration, parameters have uncertainties and the effect of these uncertainties on signal execution and prediction have not been studied extensively. Additionally, models encoding biological processes are becoming increasingly larger and harder to visualize. Therefore, the overarching goal of this work is to analyze and visualize the Apoptosis and JNK3 models to understand how different signal execution patterns arise due to parameter uncertainty. In the following sections we present an introduction about the central topics involved in this work.

1.1 Introduction to ODE models and calibration

The Human Genome Project and other large-scale projects have provided genomics, transcriptomics, and proteomics data that has enabled the development of mathematical models to study biological processes (Collins & McKusick, 2001; Legrain et al., 2011). The first step to build a model consists in defining the possible interactions between the molecules in a network. This is accomplished by studying the experimental data available from the signaling mechanism of interest as well as exploring databases, e.g. STRING (Szklarczyk et al., 2019), that contain known and predicted protein-protein interactions. These networks can be studied through different mathematical frameworks including Ordinary Differential Equations (ODE), Boolean networks, stochastic equations and petri nets (Bartocci & Lió, 2016; W. Chen et al., 2010; Machado et al., 2011). In this work, we focused on the analysis of mechanistic ODE models which are used to describe the temporal dynamics of protein concentration driven by consumption and production in different interactions. ODE-based models are particularly useful in the analysis of regulatory motifs like feedback mechanisms and cascade motifs that control the relationship between stimuli and responses (Alon, 2006).

In the ODE framework, cellular reactions are described by the law of mass action, which indicates that the rate of a chemical reaction is directly proportional to the concentration of the molecules involved in the reaction (Voit et al., 2015). Thus, for example the dynamics of an enzymatic reaction, E + S <> ES > E + P, is described by the following system of ODEs:

$$\frac{d[E]}{dt} = -\,k_f[E][S] + k_r[ES] + k_{cat}[ES]$$
$$\frac{d[S]}{dt} = -\,k_f[E][S] + k_r[ES]$$
$$\frac{d[ES]}{dt} = k_f[E][S] - k_r[ES] - k_{cat}[ES]$$
$$\frac{d[P]}{dt} = k_{cat}[ES]$$

Where the expression $d[\cdot]/dt$ on the left-hand side represent the rate of change of a molecular species over time and the expressions on the right-hand side describe the interactions that controls the rate of change of a protein. The solution of this systems of differential equations yields the concentrations of the molecules over time. These mechanistic ODE-based models contain two types of parameters: kinetic parameters (in this case k_f, k_r, k_{cat}), which describe the strength of protein-protein interactions, and initial conditions that refer to the initial protein levels at the beginning of a simulation. Initial protein concentrations can be measured experimentally but all initial conditions of a model are rarely measured (Albeck et al., 2008; Neumann et al., 2010). Kinetic parameters are usually not measured due to technical difficulties or they are measured *in vitro* where it is not clear whether these values are similar to values in a crowded intracellular environment (Schreiber et al., 2009). Thus, the majority of parameter values in a model are unknown and must be estimated in order to obtain quantitative insights from biological network-driven processes. Model parameters are inferred by comparing simulated trajectories of molecular species to their corresponding experimental data and adjusting parameter values so that the simulated trajectories reproduce the experimental data as close as possible. In the next section, we will introduce the concept of cost function, which determines how well a simulation from a calibrated parameter vector fits the experimental data.

1.2 Defining a cost function for calibration

To compare the simulation results from a model to the experimental data it is necessary to define a suitable metric. This metric should take into account the simulated data as well as the experimental data and measure how different the trajectories are from each other. One of the most common cost functions is chi-squared which measures the squared differences of the simulated data to the experimental data and normalizes it by the variance obtained from replicated experiments. Chi-squared is defined as follows:

$$\chi^2 = \sum_t \sum_i \frac{1}{2\sigma_i^2(t)} [x_{model}^i(t;\ \Theta) - x_{data}^i(t)]^2$$

Where Θ is the parameter vector used to run a simulation, $x_{model}^i(t;\ \Theta)$ correspond to the simulated trajectory of molecular species i with parameter vector Θ, x_{data}^i is the experimental data from species i, and σ_i^2 is the variance from the experimental data of species i. The index i runs over all species (observables) that have experimental data, and the index t runs over all time points at which the data was measured. Thus, calibrating a mechanistic model corresponds to minimizing this function, that is finding the parameter vector that makes the difference between experimental data and simulated trajectories closest to zero. This optimization can be performed by different algorithms including Particle Swarm Optimization (Kennedy & Eberhart, 1995) and Differential Evolution Adaptive Metropolis (Shockley et al., 2018; Jasper A. Vrugt, 2016). Additionally, χ^2 is also the negative log of the likelihood that the data will be observed for a given set of parameters, assuming that measurement errors at time t have a Gaussian distribution with variance σ_i^2. This opens the possibility of using Bayesian methods to obtain calibrated parameters. Bayesian methods have the benefit that they calculate the uncertainty of the parameters and also provide information about the parameter vectors that are more likely.

1.3 Types of calibration and quantification of parameter uncertainty

In the past, mathematical models used to be calibrated manually (Kholodenko et al., 1999), where kinetic parameter where derived from basic physical-chemical quantities and fine-tuned manually to fit the data and the quality of the fit was assessed visually. Thanks to advances in computation, novel algorithms have been developed. These algorithms can be categorized in deterministic, stochastic, and hybrid approaches (Heinemann & Raue, 2016).

Deterministic approaches consist in harnessing the information provided by the derivative of the cost function to determine the direction in parameter space where the cost function is minimum. Algorithms use this information to define the next parameter vector to sample and iteratively reduce the value of the cost function. There are algorithms like gradient descent that use the first order derivative, but they can easily get stuck at saddle points in a high dimensional parameter space. To address this, algorithms like Newton's method (Meza, 2011) and the Levenberg-Marquardt algorithm (More, 1978) use the curvature information provided by the second order derivative to optimally navigate the parameter space and improve convergence. The disadvantage of deterministic approaches is that optimization runs can get stuck in local minima. To circumvent this problem, multi-start approaches are utilized where optimization algorithms are run multiple times starting from different positions in parameter space. The assumption is that starting from different locations improves the chances of finding a global minimum.

Stochastic approaches evaluate the cost function multiple times and use heuristics to search for the parameter vectors that minimize the cost function (Abdel-Basset et al., 2018). Heuristic methods commonly used in systems biology (Sun et al., 2012) include simulated annealing (Dekkers & Aarts, 1991), genetic algorithms (Deb, 1999), differential evolution (Storn & Price, 1997), and particle swarm optimization (Kennedy & Eberhart, 1995). Stochastic approaches usually do not include gradient information, but novel hybrid methods, e.g. scatter search (Egea et al., 2007), initially explore the parameter space using stochastic algorithms and then switch to gradient-based methods to fine-tune the search locally.

Given that experimental measurements have uncertainties, when models are calibrated these uncertainties are propagated to the fitted parameter values and to the model-simulated predictions, this uncertainty is defined as practical identifiability (Vanlier et al., 2012). Additionally, depending on the structure of the model there can be structural identifiability when model parameters are functionally related and thus parameter values cannot be constrained to reasonable ranges (Raue et al., 2009). Multiple methods exist to quantify parameter uncertainty (Mitra & Hlavacek, 2019a). In addition to quantifying parameter uncertainty, it is important to understand the effect of these uncertainties in signal execution through biomolecular networks. Simulations with different parameter vectors can show different signal executions through the network, and different sensitivities to perturbations in the network. The next section describes a dynamic approach to study the effect of parameter uncertainty on signal execution through a network.

1.4 Introduction to dynamic flux analysis

Studying signal execution in networks is difficult because there are many factors that affect signal execution. Some of these factors include the number and strength of interactions that proteins can have, and the concentration dynamics of biomolecules. To study the functional consequences of molecular polyspecificity, Stites and colleagues combined the measurements of protein abundance and binding affinities with a computational framework and determined that the relative importance of protein-protein interactions is cell line dependent (Stites et al., 2015). To study network dynamics Harush *et al*. analyzed multiple networks with different topologies and used a perturbative formalism to analyze their signal flow at steady state (Harush & Barzel, 2017). They analytically tracked the contribution of all nodes to the flow of information and then derived an equation that relates network structure and its dynamics with flow patterns. Although these studies provided important insights about signal execution, they were developed to study signal flow at steady state conditions, omitting transient dynamics. These dynamics are important as they provide information about signal flow through the network and the timing of perturbations required to better modulate a cell response.

An approach that takes into account the effect of transient dynamics and parameter uncertainty in models of signaling networks is presented in this work. In this method, described in more detail in Chapter 3 and 4, a model is simulated with a calibrated parameter vector and instantaneous reaction rates are calculated for each protein-protein interaction. Then, using an approach inspired in tropical algebra (Noel et al., 2011) and ultradriscretization theory (Kato et al., 2017a), at each simulated time point a dominant subnetwork is obtained from tracking the reactions with highest signal flow. The sequence of dominant networks is defined as a signal execution fingerprint. This discretization step is repeated for each calibrated parameter vector. Then, our method uses the longest common subsequence metric, and clustering algorithms to compare and group execution fingerprints, respectively. Groups with similar execution fingerprints are said to have the same execution mode. Thus, our analysis is one of the first approaches, to the best of our knowledge, that takes into account non-equilibrium dynamics of signaling networks and parameter uncertainty to create a paradigm shift towards a more probabilistic understanding of signal execution in biochemical networks.

In addition to analyses of signal flow under different conditions, visualization techniques can be used to further study cellular signaling processes. These visualizations are useful as they help researchers to perform exploratory analysis of the model topology, detect patterns that arise from reaction dynamics, and generate hypotheses about the importance and regulation of biomolecules in a network. In the next section, we discuss visualizations of network-driven biological processes.

1.5 Introduction to network visualization

It has become increasingly clear that biological processes are the result of complex interactions of cellular molecules including proteins, RNA, DNA, and metabolites (Barabási & Oltvai, 2004). These complex interactions are represented in a network, where nodes correspond to molecules involved in a pathway and edges represent interactions between molecules. Edges can be undirected when there is not enough information about the direction and nature of the interaction between the nodes and directed when there is information about information flow (transcription factor to gene it regulates) or material flow (substrate to a product). The dynamics of these network-driven biological processes are studied using mathematical models to extract mechanistic insights about the regulation of these biological processes (Aldridge et al., 2006). As new interactions and crosstalk proteins are discovered and integrated into mathematical models, these have become larger and difficult to identify relevant structures and patterns of signal execution. In this scenario, visualization tools present one effective way to explore network processes, acquire conceptual insights about signal-execution mechanisms, and quickly generate mechanistic hypotheses about dynamic regulation.

Tools to visualize network-driven biological processes that are encoded in mathematical models can be classified in two groups, static and dynamic. Static representations include molecular species network (Bergmann et al., 2017), species-reactions network (Schaff et al., 2016), contact map (Boutillier et al., 2018; Cheng et al., 2014; L. A. Harris et al., 2016), rules defined in the model (Boutillier et al., 2018; Cheng et al., 2014), rules network (Danos et al., 2012; Smith et al., 2012), and atom-rule graphs (Sekar et al., 2017). Dynamic visualizations integrate information about temporal changes obtained from a simulation into networks, and facilitate the understanding of causality, feedback mechanisms, or oscillations that occur during biological processes. Current approaches to represent model dynamics include: Copasi (Bergmann et al., 2017) and the Kappa Dynamic Influence Network (Forbes et al., 2017). In Copasi, a tool only compatible for SBML models (Hucka et al., 2003), species concentrations from simulation results are encoded qualitatively in the size of the box around the nodes of a graph and no information is included about reaction rate values and concentrations. The Kappa Dynamic Influence

Network, a web service tool for Kappa models (Boutillier et al., 2018), provides a node-link diagram that displays the temporal influence that each rule has on other rules. Overall, these tools have provided important insights about biological networks. However, there is still an unmet need to effectively visualize increasingly large models and to quantitatively and intuitively display reactions fluxes throughout networks to identify dynamic patterns of signal execution.

A tool that enables the visualization of large networks and their dynamics is presented in this work. This tool, described in more detail in Chapter 5, is a Python package that provides interactive visualizations within the Jupyter Notebook web application (Kluyver et al., 2016). We used Jupyter notebooks because it is a platform for literate programming (Knuth, 2001) that enables researchers to define both code and documentation at the same time whilst developing a workflow for model definition, visualization, and analysis. Our tool has a two-fold functionality: I) it facilitates exploratory analysis of the network and dynamics of biochemical models to generate novel hypotheses, and II) it promotes the reproducibility and dissemination of model analysis pipelines. Thus, this tool accelerates the process of model analysis to generate novel insights about the mechanisms of cell behavior.

1.6 Reproducibility of systems biology models

Reproducibility of published results is essential for the advancement and credibility of science. However, multiple studies have demonstrated that a large percentage of studies published in scientific journals are not reproducible (Baker & Penny, 2016). In a computational field like systems biology, it would be expected that most published models would be reproducible, but a recent survey identified that ~50% of the models considered are not reproducible (Tiwari et al., 2021). Thus, in this work we deposited all code used for analysis and algorithms in GitHub (https://github.com/LoLab-VU) and used Binder (Jupyter et al., 2018) to enable researchers to reproduce our analysis on mybinder.org servers. Also, all tools developed here can be easily integrated into Jupyter Notebooks (Kluyver et al., 2016), which are files with code and documentation that are easy to share.

1.7 Organization

In Chapter 2, we introduce a model, calibration and analysis of the JNK3 activation cascade. In Chapter 3, we introduce Pydyno, which is a software tool to study signal flow through biochemical networks under different conditions. We include details about the implementation, sequence analysis and visualization methods. In Chapter 4, we present a detailed analysis of the Apoptosis pathway using Pydyno. We show how model calibration yields multiple parameter vectors that fit the experimental data and how these parameter uncertainties affect signal execution. In Chapter 5, we present PyViPR, a tool that combines community detection algorithms and simulation results to visualize large models and their dynamics. In Chapter 6, we discuss the results of this work and make conclusion remarks.

Chapter 2

Development and analysis of the JNK3 activation Reaction Model (JARM)

2.1. Introduction: Biological background and model scope

JNK3 is a mitogen-activated protein kinase (MAPK) involved in different physiological processes like apoptosis and cell proliferation when it is activated, and its upregulation has been implicated in neurodegenerative diseases (Yoon et al., 2012). The cascade to activate JNK3 starts from ASK1 (MAP3K kinase), which then activates MKK4/7 (MAP2K kinases) by phosphorylating them, and in turn MKK4/7 activate JNK3 by phosphorylating the tyrosine and threonine sites, respectively (Keshet & Seger, 2010). Recent studies showed that arrestin-3 is a scaffold that facilitates the activation of JNK3 (Zhan et al., 2013). JNK3 is phosphorylated by MKK4 and MKK7 in solution and in the arrestin-3 scaffold, which results in a variety of protein complexes and reactions. However, the mechanisms by which the scaffold improves the activation rate of JNK3 are poorly understood. Thus, it is important to analyze the interactions and dynamics of arrestin-3, MKK4/7 and JNK3 to gain insights about the mechanisms that control JNK3 activation.

We developed the JNK3 Activation Reaction Model (JARMv1.0) to understand the systems-level regulation of JNK3 activation by analyzing the dynamics and reaction rates of the interactions between arrestin-3, MKK4/7 and JNK3. In JARM, arrestin-3 have two binding sites, one where MKK4 or MKK7 binds, and the other where JNK3 binds. JNK3 have a tyrosine site that can be phosphorylated by MKK4 and a threonine site that can be phosphorylated by MKK7. Also, MKK4/7 can phosphorylate JNK3 in the presence or absence of the arrestin-3 scaffold. In the arrestin-3 scaffold, after MKK4 phosphorylates its respective site, it can be replaced by MKK7, and vice versa. JNK3 is activated when it has been phosphorylated both in the tyrosine and threonine sites. A diagram of all model interactions is shown in Figure 2.1, and a summary of species and reactions generated by PySB are shown in Table 2.1 and Table 2.2, respectively.

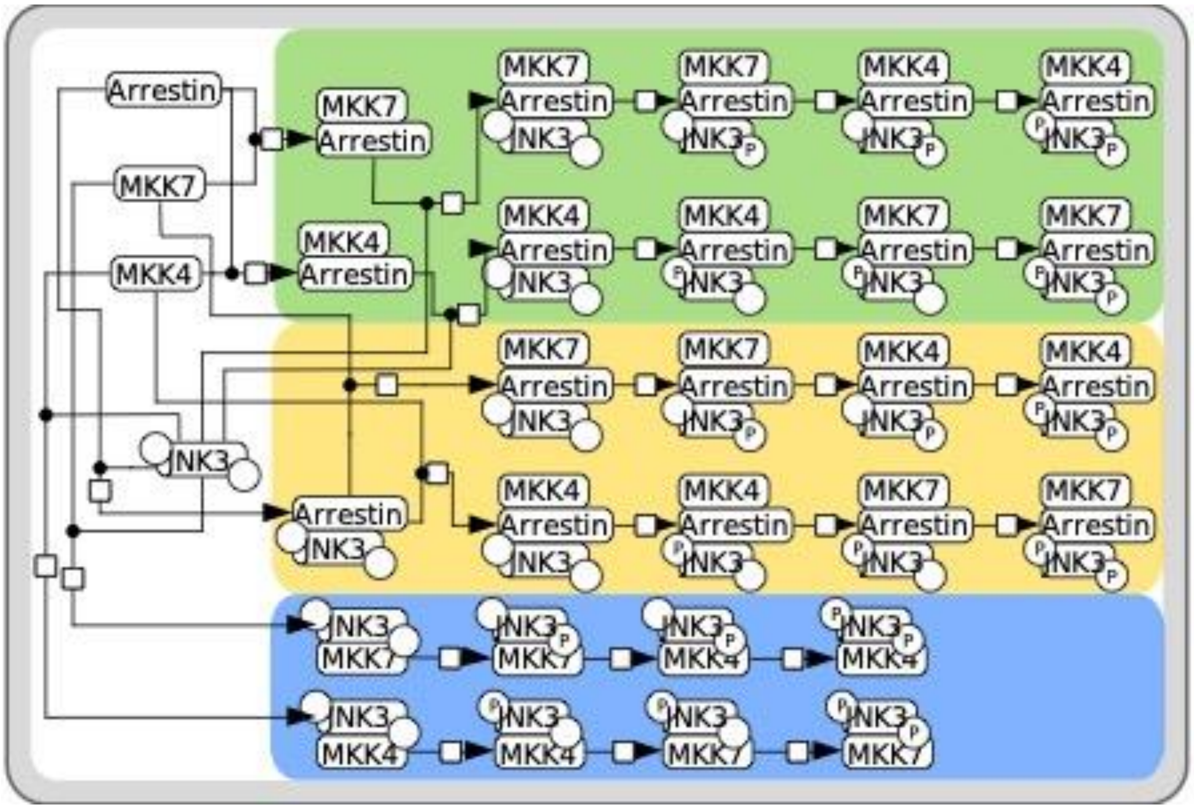


Figure 2.1 Network of all possible interactions between MKK4, MKK7, JNK3, and arrestin-3. Nodes represent the different complexes formed by the interacting species, and circles in the JNK3 node show the phosphorylation state of the tyrosine (left) and threonine (right) sites. The different orders in which JNK3 can be phosphorylated are highlighted by color: arrestin-3:MKK4/7 formation before JNK3 binding (green); arrestin-3:JNK3 complex formation before MKK4/7 binding (yellow); and MKK4/7 binding JNK3 in the absence of arrestin-3 (blue). The convention of Kitano et al (Kitano et al., 2005) was followed.

Species	**Starting concentrations (μMolar)**
Arrestin	0, 5
pMKK4	0.05
pMKK7	0.05
uuJNK3	0.59
puJNK3	0
upJNK3	0.0067
Arrestin:pMKK4	0
Arrestin:pMKK7	0
Arrestin:uuJNK3	0
Arrestin:upJNK3	0
Arrestin:puJNK3	0
uuJNK3:pMKK4	0
puJNK3:pMKK4	0
upJNK3:pMKK4	0
uuJNK3:pMKK7	0
upJNK3:pMKK7	0
puJNK3:pMKK7	0
Arrestin:uuJNK3:pMKK4	0
Arrestin:upJNK3:pMKK4	0
Arrestin:puJNK3:pMKK4	0
Arrestin:uuJNK3:pMKK7	0
Arrestin:puJNK3:pMKK7	0
Arrestin:upJNK3:pMKK7	0
ppJNK3:pMKK4	0
ppJNK3:pMKK7	0
Arrestin:ppJNK3:pMKK4	0
Arrestin:ppJNK3:pMKK7	0
ppJNK3	0

Table 2.1 JARM species and their corresponding initial conditions. The first u in JNK3 corresponds to the threonine site, and the second u is the tyrosine site

PySB-generated Reaction	**Rate or Equilibrium Constant**
Arrestin + pMKK4 ↔ Arrestin:pMKK4	KD_pMKK4_Arr = 347 µmolar
Arrestin + pMKK7 ↔ Arrestin:pMKK7	KD_pMKK7_Arr = 13 µmolar
Arrestin + uuJNK3 ↔ Arrestin:uuJNK3	KD_uuJNK3_Arr = 1.4 µmolar
Arrestin + upJNK3 ↔ Arrestin:upJNK3	KD_upJNK3BindArr = 4.2 µmolar
Arrestin + puJNK3 ↔ Arrestin:puJNK3	KD_puJNK3BindArr = 10.5 µmolar
Arrestin:pMKK4 + uuJNK3 ↔ Arrestin:pMKK4:uuJNK3	KD_MKK4_Arr_bind_uuJNK3 = 1.4 µmolar
Arrestin:pMKK4:uuJNK3 → Arrestin:pMKK4:upJNK3	kcat_pMKK4_ArrJNK3
Arrestin:pMKK4:puJNK3 → Arrestin:pMKK4:ppJNK3	kcat_pMKK4_ArrJNK3
Arrestin:pMKK4:upJNK3 ↔ Arrestin:upMKK4 + upJNK3	KD_upJNK3_bind_Arr_MKK4
Arrestin:pMKK4:puJNK3 ↔ Arrestin:puMKK4 + puJNK3	KD_puJNK3_bind_Arr_MKK4
Arrestin:pMKK4:ppJNK3 ↔ Arrestin:ppMKK4 + ppJNK3	KD_ppJNK3_Arr = 220 µmolar
Arrestin:pMKK7 + uuJNK3 ↔ Arrestin:pMKK7:uuJNK3	KD_MKK7_Arr_bind_uuJNK3 = 1.4 µmolar
Arrestin:pMKK7:uuJNK3 → Arrestin:pMKK7:puJNK3	Kcat_pMKK7_ArrJNK3
Arrestin:pMKK7:upJNK3 → Arrestin:pMKK7:ppJNK3	Kcat_pMKK7_ArrJNK3
Arrestin:pMKK7:puJNK3 ↔ Arrestin:puMKK7 + puJNK3	KD_puJNK3_bind_Arr_MKK7
Arrestin:pMKK7:upJNK3 ↔ Arrestin:upMKK47+ puJNK3	KD_upJNK3_bind_Arr_MKK7
Arrestin:pMKK7:ppJNK3 ↔ Arrestin:ppMKK7 + ppJNK3	KD_ppJNK3_Arr = 220 µmolar

Arrestin:uuJNK3 + pMKK4 ↔ Arrestin:uuJNK3:pMKK4	KD_MKK4BindArr_uuJNK3
Arrestin:puJNK3 + pMKK4 ↔ Arrestin:puJNK3:pMKK4	KD_MKK4BindArr_uuJNK3
Arrestin:uuJNK3 + pMKK7 ↔ Arrestin:uuJNK3:pMKK7	KD_MKK7BindArr_JNK3
Arrestin:upJNK3 + pMKK7 ↔ Arrestin:upJNK3:pMKK7	KD_MKK7BindArr_JNK3
Arrestin:pMKK4:upJNK3 + pMKK7 → Arrestin:pMKK7:upJNK3 + pMKK4	keq_pMKK4_to_pMKK7
Arrestin:pMKK7:puJNK3 + pMKK4 → Arrestin:pMKK4:puJNK3 + pMKK7	keq_pMKK7_to_pMKK4
pMKK4 + uuJNK3 ↔ pMKK4:uuJNK3	KD_MKK4_uuJNK3
pMKK4 + puJNK3 ↔ pMKK4:puJNK3	KD_MKK4_puJNK3
pMKK4:uuJNK3 → pMKK4:upJNK3	kcat_pMKK4_ArrJNK3
pMKK4:puJNK3 → pMKK4:ppJNK3	kcat_pMKK4_ArrJNK3
pMKK4:upJNK3 ↔ pMKK4 + upJNK3	KD_pJNK3_MKK4complex
pMKK4:ppJNK3 ↔ pMKK4 + ppJNK3	KD_pJNK3_MKK4complex
pMKK7 + uuJNK3 ↔ pMKK7:uuJNK3	KD_MKK7_uuJNK3
pMKK7 + upJNK3 ↔ pMKK7:upJNK3	KD_MKK7_upJNK3
pMKK7:uuJNK3 → pMKK7:puJNK3	kcat_pMKK7_ArrJNK3
pMKK7:upJNK3 → pMKK7:ppJNK3	kcat_pMKK7_ArrJNK3
pMKK7:puJNK3 ↔ pMKK7 + puJNK3	KD_pJNK3_MKK7complex
pMKK7:ppJNK3 ↔ pMKK7 + ppJNK3	KD_pJNK3_MKK7complex

Table 2.2 Reactions defined in JARM. The left column shows the bidirectional and catalytic reactions that are generated from PySB rules. The right column shows the kinetic parameters involved in each reaction. Experimentally measured kinetic constants are highlighted in red.

2.2. Materials and methods

2.2.1. Experimental data

Our collaborators generated all the key data to study the role of arrestin-3 in the JNK3 activation cascade. The experimental data can be found in reference (N. A. Perry et al., 2019). This data consisted of 54 measurements at different time points of JNK3 phosphorylated in both the threonine and tyrosine sites, JNK3 phosphorylated only in the threonine site, and JNK3 phosphorylated only in the tyrosine site. Each data point was collected in triplicate enabling the calculation of the mean and standard deviation, which were used for model calibration. Additionally, our collaborators measured the affinities of MKK4/4 and JNK3 for arrestin-3, effectively reducing the number of parameters that had to be calibrated in JARMv1.0

2.2.2. Model implementation and calibration

JARMv1.0 was implemented in Python using the PySB framework (Lopez et al., 2013) that enables users to build mathematical models of biochemical systems as Python programs. In all, JARMv1.0 included 28 biochemical species, 60 chemical reactions, and 44 free parameters. When this program is executed, a set of rules is translated to 28 ordinary differential equations (ODEs) using the mass action kinetics formalism. The solution of the ODEs shows how the concentration dynamics of the molecular species change as a result of the interactions in which they are involved. Since the ODEs generated by JARM require association and dissociation rate constants instead of dissociation equilibrium constants (K_D) values, all experimentally determined K_D values were converted into rate parameters using the equation $K_D=k_r/k_f$. For the model calibration, k_f rates were set within the "average enzyme" distribution of specificity constant (Aldridge et al., 2006; Bar-Even et al., 2011; Zheng et al., 2012), whereas the k_r were allowed to change. Out of linear range values at initial time points and higher than theoretical maximum values in the time-course data were set to zero and the theoretical maximum, respectively. We applied scaling normalization to both the time-course data and the simulated trajectories.

Model calibration was performed using the PyDREAM package (Shockley et al., 2018), which is a python implementation of the (Multiple-Try) Differential Evolution Adaptive Metropolis (DREAM$_{(ZS)}$) algorithm (J. A. Vrugt & Ter Braak, 2011). To reduce the number of model parameters to calibrate, we used experimentally measured dissociation constants for the interactions between MKK4/7, JNK3, and arrestin-3 (N. A. Perry et al., 2019), and literature values for the interactions between MKK4/7 and JNK3 alone (Ho et al., 2006). We assumed that the catalytic constants for JNK3 phosphorylation were the same within a scaffold complex and in solution. The model was pre-equilibrated to allow complexes to form before the reaction is initiated. Parameter prior probabilities were specified as uniform distributions with the lower and upper boundary set with the lowest and highest values from the protein–protein kinetic interaction data and structures (PPKIDS) dataset (H. Bai et al., 2011), indicating a lack of knowledge about the likely parameter values. Literature-based values and experimental measures were fixed. The fit of simulated trajectories to experimental data was measured using the sum of the squared differences:

$$\chi^2 = \sum_t \sum_i \frac{1}{2\sigma_{data}^2}[x_{model}^i(t;\Theta) - x_{data}^i(t)]^2$$

Where t is the time span of the simulation and experiments, $\Theta = (\theta_1, \dots, \theta_n)$ are the parameters of the model, x_{model}^i are the simulations of the model under condition i *and* $x_{data}^i\,(t)$ is the experimental data under condition *i*. This function corresponds to the negative log of the likelihood ($-\ln(likelihood(\Theta))$) in the Bayesian framework assuming the measurement errors at time *t* have a normal distribution. Using the Bayes formula, the value of the log posterior distribution, i.e. the probability of a parameter vector given the experimental data, for a particular parameter vector is defined as

$$-\ln(post(\Theta)) \propto -\ln(likelihood(\Theta)) - \ln\,(prior(\Theta))$$

Where $post(\Theta)$ is the posterior probability of parameter vector Θ, and $prior(\Theta)$ are the prior probabilities assigned to each of the parameters in the parameter vector Θ. PyDREAM samples the posterior distribution and finds probable parameter sets that fit the experimental data within the constraints that the network interactions and the network kinetics imposes. To further constraint the model, we added the requirement that thermodynamic cycles present in the interaction network must obey detailed balance/free energy conservation in the likelihood function.

The $\text{DREAM}_{(ZS)}$ algorithm was run using PyDREAM with five chains that started in parameter locations provided by previous calibrations using the Particle Swarm Optimization algorithm (Kennedy & Eberhart, 1995). Each chain sampled 500,000 parameter sets and the first 250,000 samples were discarded as burn-in. All chains were tested for convergence using the Gelman-Rubin criterion. To measure the goodness of fit of the calibrated model to the experimental data we use the χ^2 statistic. A good fit would correspond to a χ^2 value that is approximately equal to the number of experimentally obtained data points. Using the 5000 most likely parameters obtained from the calibration we obtained a χ^2 value of 32.32 (±0.09). Given that we measured 36 experimental points for pTyr-JNK3 and pThr-JNK3 reactions, we conclude that the model shows a reasonable fit to the experimental data.

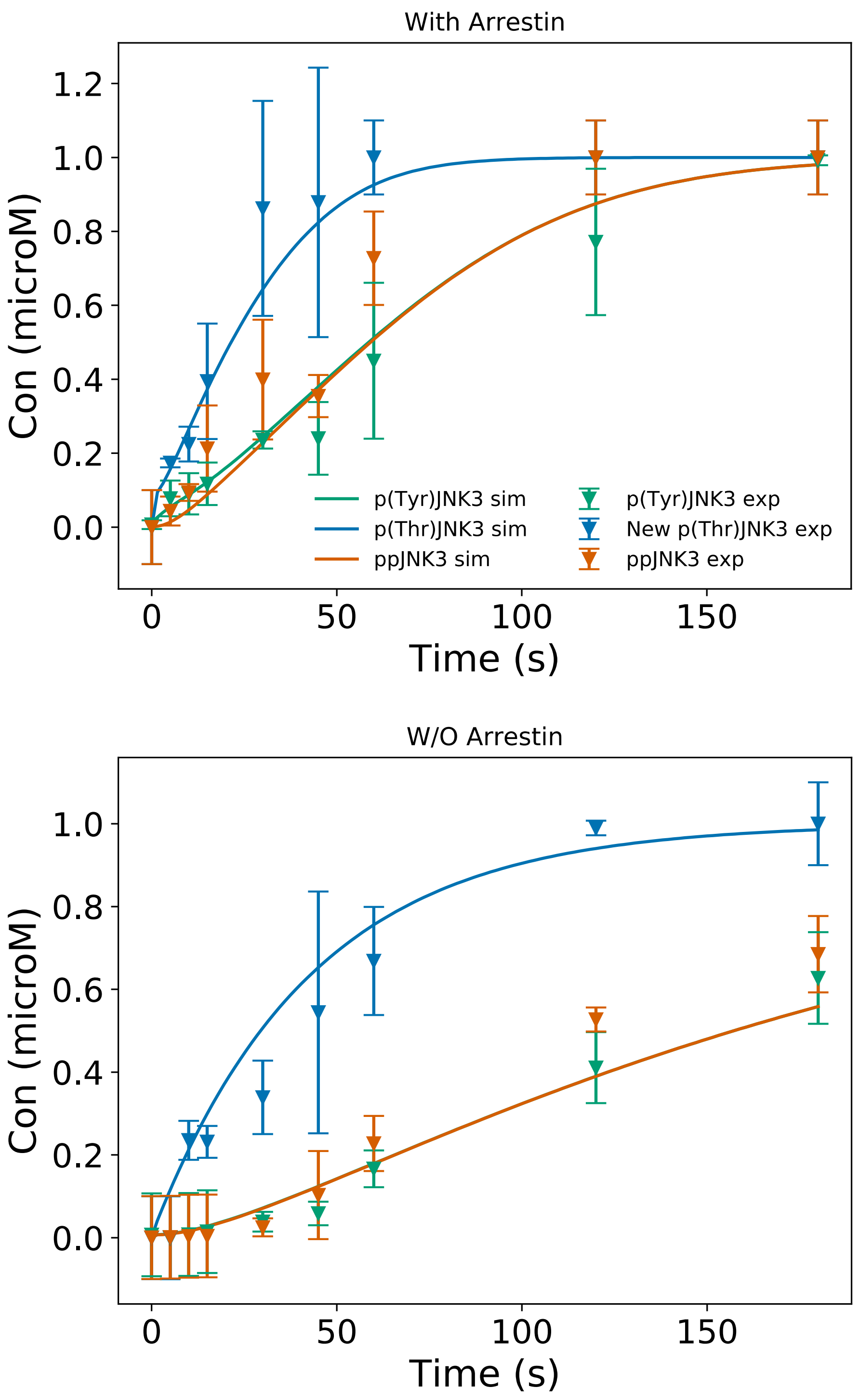


Figure 2.2 Simulated trajectories of p(Thr)JNK3, p(Thr)JNK3 and doubly phosphorylated JNK3 calibrated to reproduce the experimental data. Dots and error bars indicate the mean and standard deviation of the experimental data and solid lines represent the model simulations.

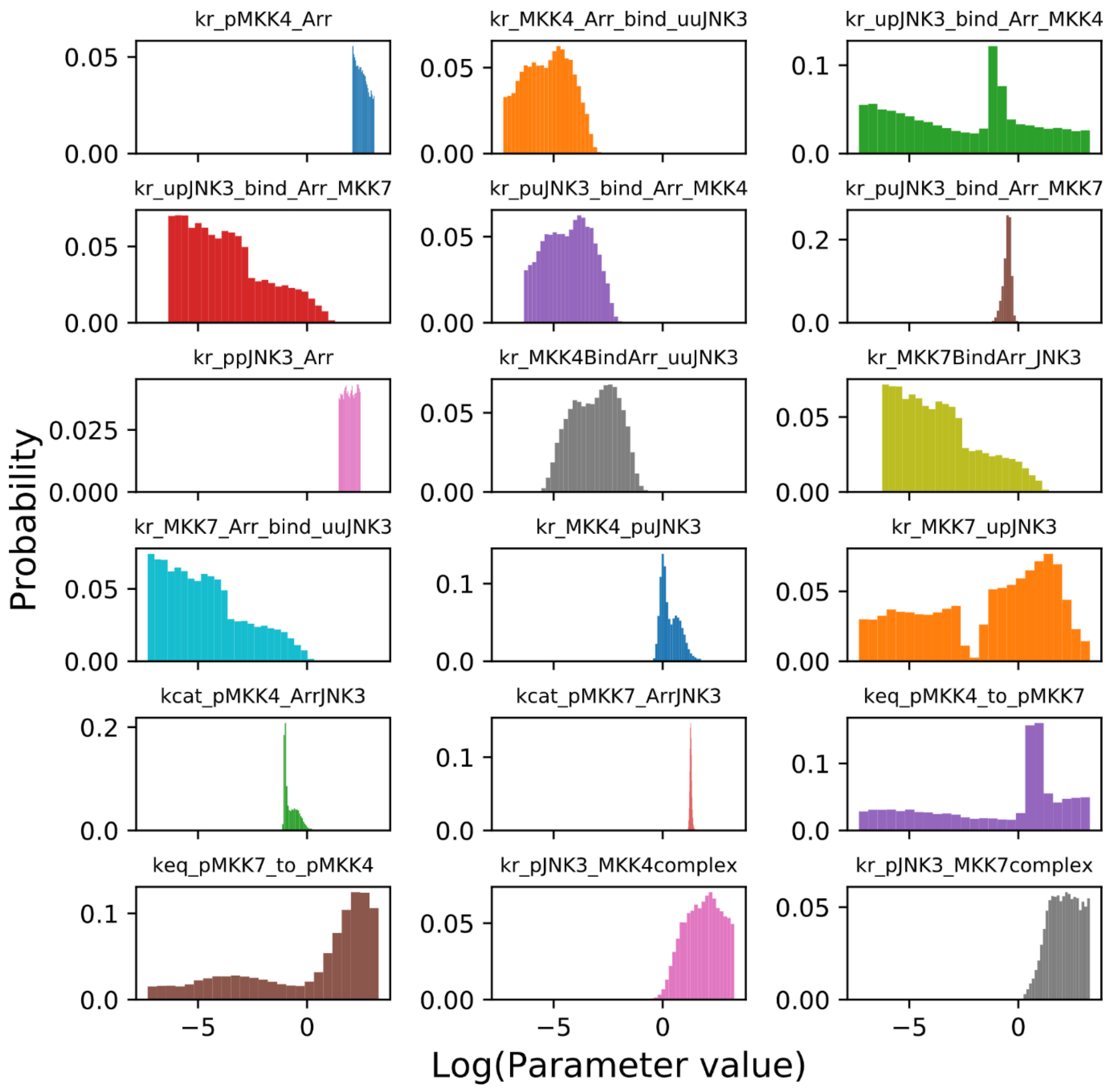
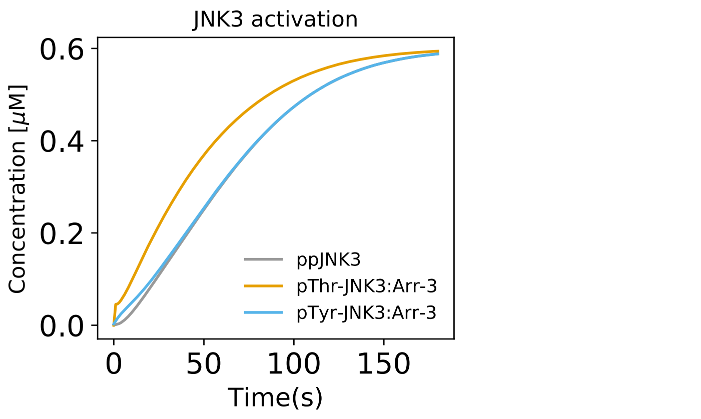


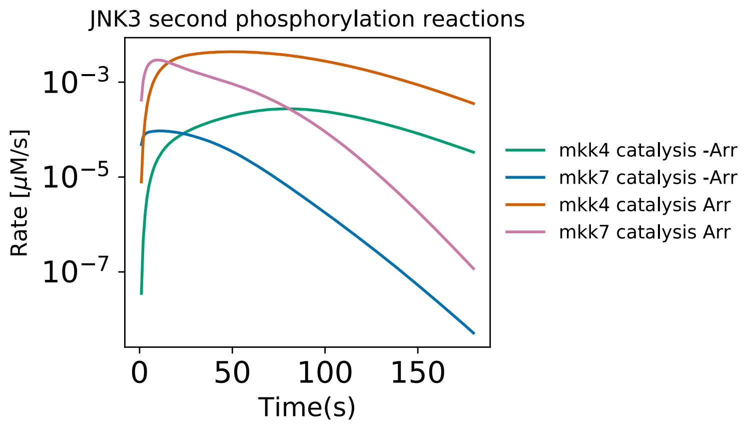
Figure 2.3 Posterior probability distributions for calibrated JARM kinetic parameters. The x-axis corresponds to the base-10 logarithm range for possible values of kinetic parameters in JARM. The y-axis represents the probability of each of the possible parameter values. The resulting distributions provide an idea of the range and likelihood of each parameter value given the experimental data. As shown, some parameters exhibit a broad range of values (~3 orders of magnitude) but the key catalytic parameters for arrestin-3 are well constrained. Parameter values were obtained as described in the calibration section. Parameter names with the prefix ‘kr’, ‘kcat’ and ‘keq’ correspond to the reverse, catalysis (sec^{-1} units) and forward ((uM*sec)$^{-1}$ units) rate constants, respectively.

2.3. Results

To identify a biochemical reaction mechanism that explains the JNK3 activation, we used JARMv1.0, as it encodes the protein interactions relevant to MKK4/7 phosphorylation of JNK3 with and without arrestin-3 (Fig. 2.1). JARMv1.0 was encoded using PySB (Lopez et al., 2013) to enable hypothesis testing as we explored plausible mechanisms. We used an in vitro kinase activation assay to give the model a dynamic point of reference for calibration. This assay characterized the JNK3 phosphorylation time course, in the presence or absence of arrestin-3, by MKK7 and MKK4 (Fig. 2.2, experimental data). The data showed 2.2-fold more phosphorylation of Thr-221 than Tyr-223 at 60 s incubation when arrestin-3 is present. It also showed that ppJNK3 generation in the presence of arrestin-3 displayed a fold change between 1.2 and 1.8 higher than in the absence of arrestin-3. These data were used to calibrate JARMv1.0.

To populate JARMv1.0 parameters, we used our experimentally determined binding constants and values reported in the literature (Zhan et al., 2014). As described in the previous section, remaining parameters were estimated using PyDREAM, a Bayesian parameter inference formalism (Shockley et al., 2018). This yielded parameter probability distributions, constrained by our experimental data, rather than single best-fit values (Figure 2.3). Inspecting these distributions, we observed a few that spanned a narrow parameter range whereas others spanned a broader parameter range. These distributions can be interpreted as an estimate for model sensitivity to parameter variations. From a statistical perspective, JNK3 activation is less sensitive to parameters with broad distributions, and more sensitive to those with a narrow distribution. The calibration results suggest that the system is most sensitive to the catalytic phosphorylation constants of MKK4 and MKK7 (Figure. 2.3). This gave us confidence that experimental measurements provided suitable constraints to explore the dynamics of JNK3 phosphorylation even without direct measurements of other parameters.





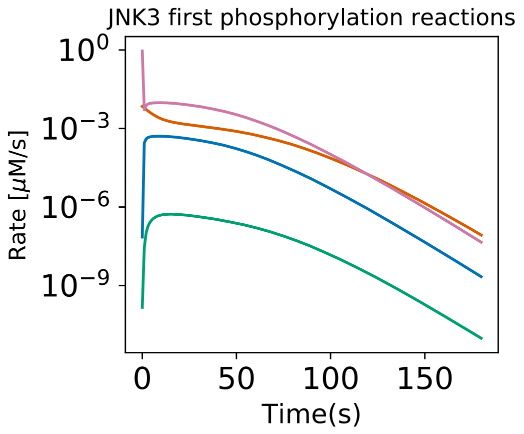
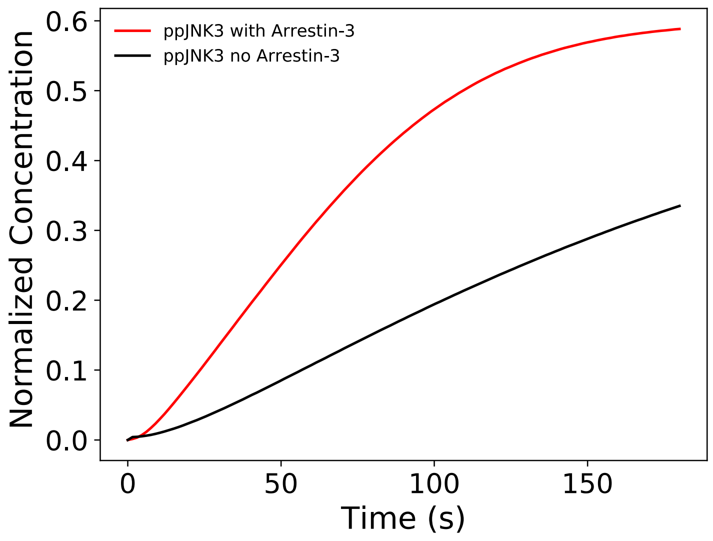


Figure 2.4 Top panel: corresponds to the time-dependent concentration changes of active (doubly phosphorylated) JNK3, pTyr-JNK3 bound to arrestin-3 and pThr-JNK3 bound to arrestin-3. Center panel: represents the catalysis reaction rate values of singly phosphorylated JNK3. Bottom panel: shows the catalysis reaction rate values of active (doubly phosphorylated) JNK3

Calibrated JARMv1.0 showed that the time-dependent concentrations of singly phosphorylated JNK3 in complex (pTyr-JNK3:Arrestin-3, pThr-JNK3:Arrestin-3) or doubly phosphorylated JNK3 (ppJNK3), exhibit nonlinear dynamics that emerge from multiple simultaneous reactions (Figure 2.2). Our Bayesian calibration estimated that the catalytic constant of JNK3 phosphorylation by MKK7 is 2 orders of magnitude faster than the one of JNK3 phosphorylation by MKK4 (Figure 2.3). As shown in Figure 2.4, in the early time points of the simulation MKK7 rapidly produces the first phosphorylation of JNK3 (pThr-JNK3) and, it accumulates in the system over time. Then, MKK4 predominantly produces the second JNK3 phosphorylation via the arrestin-3 scaffold as its reaction rate is up to one order of magnitude faster than all other second phosphorylation events with or without arrestin-3 (Figure 2.4). Taken together, these data suggest that MKK4 phosphorylation is the rate-limiting step in the JNK3 activation pathway, but multiple interactions take place to accomplish this outcome. This is consistent with active MKK4 having a lower affinity for arrestin-3 (Table. 2.2).

As previous studies have shown that less than 50% dual phosphorylation over total JNK expression can lead to a robust physiological response (Khalid et al., 2016; Lei et al., 2002; Muniyappa & Das, 2008) we used JARMv1.0 to study whether arrestin-3 changes the time to reach this threshold. We found that the reactions involving arrestin-3 reach the 50% threshold 2.63 times faster compared with the reaction system without arrestin-3 (Figure. 2.5 Top panel). Finally, we simulated JARMv1.0 with different concentrations of arrestin-3 and identified that the optimal concentration of arrestin-3 for maximal JNK3 phosphorylation is around ~0.49 μM (Figure. 2.5 Bottom panel).

All code used to build the model, fit the model parameters and perform analysis is freely distributed as open-source software and available in the Lopez lab GitHub repository (github.com/LoLab-VU/JARM).



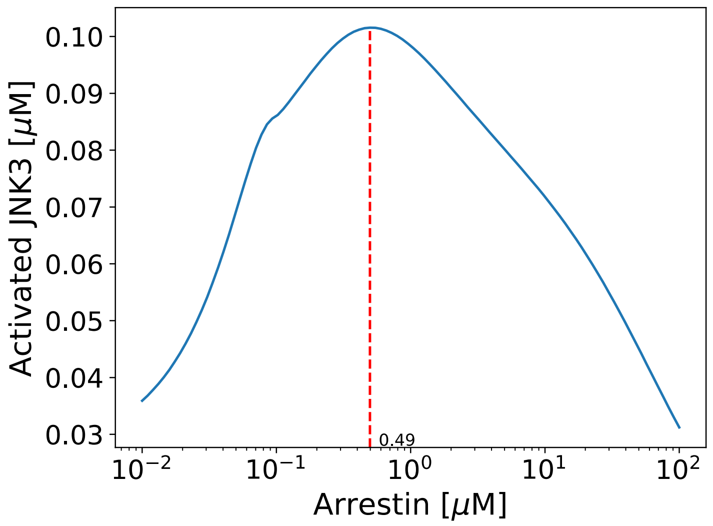


Figure 2.5 Top panel: Simulated model trajectories of doubly phosphorylated JNK3. The red and black lines show the production of ppJNK3 when arrestin-3 is present and absent, respectively. Bottom panel: Simulated effect of varying arrestin-3 concentration in the system.

Chapter 3

PyDyno: a tool to analyze parameter uncertainty in biochemical models

3.1 Summary

The advent of quantitative techniques to probe biomolecular-signaling processes have led to increased use of mathematical models to extract mechanistic insight from complex datasets. These complex mathematical models can yield useful insights about intracellular signal execution but the task to identify key molecular drivers in signal execution, within a complex network, remains a central challenge in quantitative biology. This challenge is compounded by the fact that calibrated models usually have parameter uncertainties that could yield multiple signal execution modes and thus multiple potential drivers in signal execution. Here, we present a novel approach to identify signaling drivers and characterize dynamic signal processes within a network. Our method, PyDyNo, combines physical chemistry, statistical clustering, and tropical algebra formalisms to identify interactions that drive time-dependent behavior in signaling pathways. We use our algorithm to study the effect of parameter uncertainty in the cascade of JNK3 activation. We show that given the parameter uncertainty there are different ways in which the signal can be executed, and we define them as execution modes. These execution modes respond differently, based on the dominant reactions, to the same perturbation to the network. These results can be used along experimental design approaches to identify signal execution mode of a cellular process and target the reactions through which most signal is flowing to modulate a pathway response

3.2 Introduction

Many cellular signaling processes can be represented as complex networks of interconnected biochemical components (Jordan, Landau and Iyengar, 2000). The dynamics of these networks regulate cellular functions by controlling how cells transduce external signal cues into cellular decisions (Purvis and Lahav, 2013). However, there are still fundamental questions about the dynamics of signal execution in biochemical networks, and how changes in concentration or kinetic parameters in a biochemical network could lead to system-wide reconfigurations that can manifest as different cell signaling and fate decisions. Therefore, explaining the dynamic response mechanisms of a network to perturbations and predicting outcomes based on prior knowledge is necessary to accelerate our understanding of cellular signaling dynamics.

To explain the input/output responses of intracellular signaling pathways researchers have appealed to information theoretic approaches (Cheong et al., 2011; Brennan, Cheong and Levchenko, 2012; Levchenko and Nemenman, 2014; Suderman et al., 2017; Shockley et al., 2019). These approaches cast signal transduction in terms of channel capacities, which are the maximum amounts of information that noisy biochemical pathways can transmit from an input stimulus to an output response. This kind of analysis has revealed that the maximum channel capacity of a biochemical process is context dependent and could require the cooperativity of multiple cells to achieve actual information transfer (Suderman et al., 2017). In previous work, we applied information theoretic analysis to the COX-2 reaction network and observed that given an external noise source and a limited set of data, some paths were more or less likely to be utilized depending on the initial levels of signaling molecules (Shockley et al., 2019). Despite the insights provided by information theoretic approaches, little has been done to understand the actual network flux which controls the input/output response. Indeed, the fundamental philosophy of information theoretic approaches is to treat the biochemical network purely in terms of input/output relationships, thereby disregarding the details of the transient dynamics which underpin signal transduction through the network. As such, questions remain about how non-equilibrium dynamics flow through a biochemical network, how the patterns of transient reaction-flux control network response, as well as what dynamic paths are plausible under a given set of conditions.

Reaction-flux based analysis is particularly challenging because many concurrent biochemical interactions take place at any given time in biological networks (Nobeli, Favia and Thornton, 2009) and the reaction rates of these interactions change over time. The quantity of interactions and temporal dynamics makes it difficult to identify meaningful patterns across all the competing biochemical reactions. Ultra-discretization (Kato, Tsujimoto and Zuk, 2017a) and tropical geometry (Noel, Grigoriev and Vakulenko, 2011) methods address these difficulties by mapping continuous non-linear dynamic processes onto piece-wise additive representations where only the dominant terms, i.e protein-protein interactions, are considered. This effectively separates out the most important reactions of the systems dynamics and identifies the conditions in which the important reactions change. Noel and others have demonstrated that such a procedure can also be applied to the dynamics of biochemical networks, however, they principally considered the approach to guide model reduction.

Here, we adapt the notion of dominance used in ultra-discretization and tropical geometry methods to define a dynamic signal execution fingerprint. This fingerprint encodes the temporal patterns of transient reaction flux that dominate signal execution through a biochemical network. Using this approach, we effectively reduce a complex biochemical network down to a subnetwork of dominant protein-protein interactions and thereby identify the dynamic paths relevant for signal propagation in non-equilibrium states. Our approach can be combined with Bayesian inference to rigorously incorporate the effects of parameter uncertainty on the dynamic signal processing mechanisms in biochemical network models. We clustered the signal execution fingerprints generated by the inferred parameter sets and were able to identify conserved modes of dynamic signal transduction that are plausible within the constraints of the experimental data used to train the model. This approach is particularly applicable to biochemical networks which can be represented using physicochemical models that mathematically encode the network of biochemical interactions using mass-action kinetics based differential equations.

3.3 Materials and methods

3.3.1 Development on the python ecosystem.

The Pydyno package was written in Python 3, a powerful objected-oriented programming language that is easy to learn and have a clear syntax (Van Rossum & Drake, 2009). Pydyno utilizes PySB (Lopez et al., 2013) and Tellurium (Choi et al., 2018) to develop and simulate models of network-driven biological processes. These models can be calibrated using Bayesian methods (Mitra et al., 2019; Shockley et al., 2018) to obtain kinetic parameters or initial conditions that make the model reproduce the experimental data. Then, Pydyno employs NetworkX (Hagberg et al., 2008) to obtain a network representation of the simulated models, and Numpy (C. R. Harris et al., 2020) and Sympy (Meurer et al., 2017) to compute the dominant sub-networks as described in section 2.24. Finally, Pydyno leverages SciPy (Virtanen et al., 2020) to calculate a distance matrix that is used with clustering algorithms from Scikit-learn (Pedregosa et al., 2011) to obtain groups of dominant subnetworks with similar signal executions.

3.3.2 Workflow to obtain a discretized representation of network dynamics

As shown in Figure 3.1, the PyDyNo workflow starts from a model that can be defined in the PySB or SBML format. Next, our algorithm builds a bipartite graph from the model reaction network. This bipartite graph has a set of nodes that correspond to model species, and another set of nodes that correspond to reactions, and edges only connect species nodes with reaction nodes. This bipartite graph enables us to identify the all reactions that produce a specific species. PyDyNo updates the direction of graph edges related to bidirectional reactions to indicate the net flux of the reaction at a specific time point. Then, PyDyNo starts tracking the signal flow from a user defined species target to obtain the dominant subnetwork. Each unique dominant subnetwork is assigned a specific label. This process is repeated for each simulation time point and results in a sequence of labels that provide a fingerprint of signal execution for a parameter vector. The approach to obtain dominant subnetworks is described in the next section.

3.3.3 Obtaining dominant subnetworks

3.3.3.1 Constructing the digital signature for a subnetwork

The digital signature of a model simulation is a temporal ordered sequence of labels denoting the dominant sub-network at each time point of the simulation. The dominant sub-network for a given time point is a subset of the signaling pathway over which the most signal is flowing to the production of a pre-determined target species (i.e., signaling protein or protein complex); each unique dominant sub-network is assigned a unique integer label for the digital signature.

Construction of the dominant sub-networks uses the instantaneous reaction rates at each simulation time point, which change over time as molecular species are consumed and produced. Hence, the dominant sub-networks can change at each time of the model simulations, resulting in a dynamic sequence of dominant sub-networks; i.e., the digital signature of the model simulation. The procedures for constructing the dominant sub-networks and for selecting dominant reaction are detailed in the proceeding paragraphs. Supplementary figure 4 shows an algorithm diagram version of the construction of digital signatures.

3.3.3.2 Constructing a dominant sub-network

The generation of the dominant sub-network at a given time point starts with creating a bipartite directed graph (digraph) representation of the model which consists of molecular species and reaction nodes. All molecular species and reactions within the model are encoded as nodes with unidirectional edges connecting molecular species with their respective reaction nodes. The directionality of each edge is determined by the sign of the reaction rate of the given species-reaction pair at the current time point; for reversible reactions the reaction rate is the sum of forward and reverse rate terms. The bipartite digraph provides an instantaneous snapshot of the direction of fluxes through

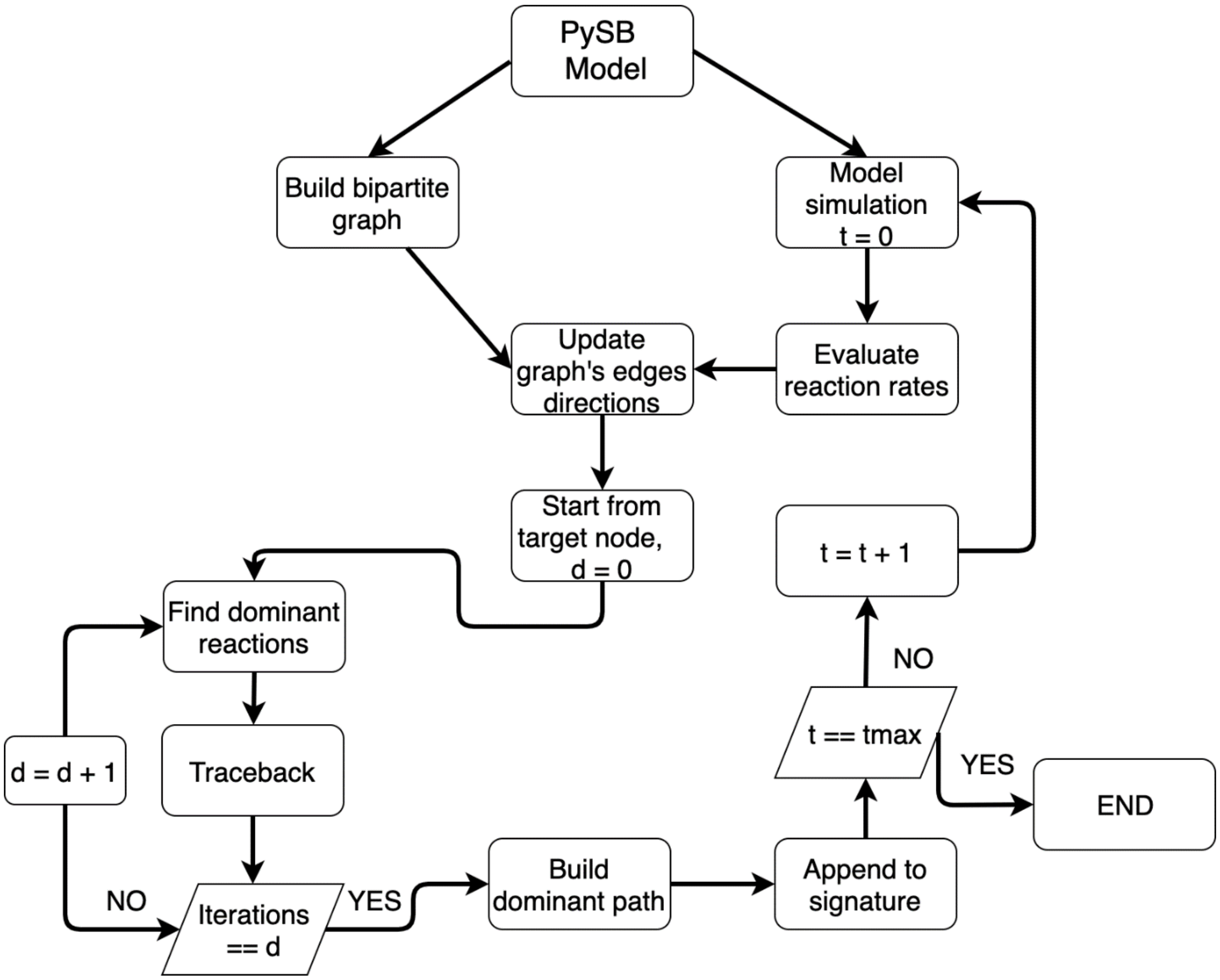


Figure 3.1 Algorithm to obtain dynamic fingerprints from a simulation. Briefly, the algorithm consists in building a bipartite graph from the model, and then simulating the model with a specific parameter vector. Next, the algorithm obtains the dominant reactions from a network and builds a subnetwork. This procedure is repeated for each simulated time point

the signaling network.

We then hierarchically construct a dominant pathway starting from the user-defined target species. For the first step, we identify the set of dominant reactions, i.e., those reactions which contribute most to the production of the target species. Dominant reactions are classified based on the instantaneous reaction rates; the conditions determining the dominant reactions are detailed in the following paragraph. We then trace back through the bipartite graph along those dominant reactions to the corresponding reactant species, which are added to the dominant sub-network. For each reactant species that was added to the sub-network we determine their dominant reactions and trace back through the bipartite graph to the next set of reactant species. This procedure is continued for a pre-determined number of iterations defined by the user parameter *depth*. Once the procedure is complete, the result is a species-to-species sub-network representing the dominant pathway over which most of the signal is flowing to produce the target at the current time point.

Note that we have defined the dominant sub-networks and their construction based on *production* of the target species. Alternatively, the procedure can be formulated to define the dominant sub-networks and their construction based on the *consumption* of the target species, which we detail in the Supplement.

3.3.3.3 Selecting dominant reactions

One aspect of network complexity is that signaling proteins can participate in multiple interactions [cite]; in a bipartite digraph this is represented by a molecular species having edges connecting it to multiple reactions. The goal therefore is to simplify the system and focus on only those reactions which are most important to the production of a species; we term this sub-set of reactions the dominant reactions. To determine the set of dominant reactions we build on some of the concepts from Noel et al. (Noel et al., 2011), who applied a tropical geometry framework to smooth ODE systems.

When the signaling network is modelled with a system of ODEs the rate of change of a molecular species, x_i, is:

$$\frac{dx_i}{dt} = \sum_{j=1}^{r} k_j S_{ij} x^{\alpha j} \qquad \textit{Eq 1}$$

where $k_j > 0$ are kinetic constants, x_i are variable concentrations, $S_{ij} = \beta_i^j - \alpha_i^j$ are the entries of the stoichiometric matrix, $\alpha_j = (\alpha_1^j, \dots, \alpha_n^j)$ are multi-indices, and $x^{\alpha j} = x_1^{\alpha_1^j} \dots x_n^{\alpha_n^j}$.

We treat the ODE of molecular species x_i as a polynomial function of the uni-directional rate terms,

$$v_i(t) = \frac{dx_i}{dt} = \sum_{j=1}^{r} M_j \qquad \textit{Eq 2}$$

where the $M_j = k_j S_{ij} x^{\alpha j}$ are the monomials representing the uni-directional rate terms. Since reversible reactions are bi-directional, they contribute two uni-directional rate monomials to the total rate of change of a species: one each for the forward (consumes x_i) and reverse (produces x_i) directions of the reaction. We account for this bi-directionality by combining the two uni-directional rate monomials into a single net reaction rate monomial term,

$$\Delta M_j = M_{j,for} + M_{j,rev} \qquad \textit{Eq 3}$$

where $M_{j,for}$ and $M_{j,rev}$ are the monomials corresponding to the forward and reverse reaction rate monomials of reaction *j* with respect to species x_i. *Eq 2* is then updated to,

$$v_i(t) = \frac{dx_i}{dt} = \sum_{j=1}^{r_{rev}} \Delta M_j + \sum_{k=1}^{r_{irrev}} M_k \qquad \textit{Eq 4}$$

where the first sum is over reversible reactions (r_{rev}) and the second sum is over irreversible reactions (r_{irrev}).

We now define the set of reaction rate terms from Eq 4 as:

$$R = \{\Delta M_1, \ldots, \Delta M_{r_{rev}}\} \cup \{M_1, \ldots, M_{r_{irrev}}\} \qquad \textit{Eq 5}$$

Since we only want to trace back the production of species x_i we can reduce R to its subset containing only the positive terms R_+,

$$R_+ = \{m \in R \mid m > 0\} \qquad \textit{Eq 6}$$

The most dominant monomial contributing to the production of x_i is then given by,

$$M_d = \max(R_+) \qquad \textit{Eq 7}$$

where M_d identifies the most dominant reaction in the production of x_i. Next, we want to identify any additional reactions which contribute to production of x_i with a similar magnitude as M_d. We therefore apply the concept of dominancy as defined by Noel et al.(Noel et al., 2011), where monomials M_i and $M_j$are said to be on a par with each other within a level $\rho > 0$ if

$$par(M_i, M_j) = \neg\, dom(M_i, M_j) = sep(M_i, M_j) < \rho \qquad \textit{Eq 8}$$

where $dom(M_i, M_j)$ is a binary function of monomials for which monomial M_j is said to dominate monomial M_k at a level $\rho > 0$ if,

$$dom(M_j, M_k) = sep(M_j, M_k) > \rho \qquad \textit{Eq 9}$$

And $sep(M_j, M_k)$ is the separation (i.e., Euclidean distance) in logarithmic space between the monomials,

$$sep(M_j, M_k) = \left|\log(|M_j|) - \log(|M_k|)\right| \qquad \textit{Eq 10}$$

Using this application of dominancy (Eq 8) the set of dominant reactions terms, D_{x_i}, is given by,

$$D_{x_i} = \{m \in R_+ \mid par(M_d, m)\} \qquad Eq\ 11$$

Thus, we construct a set of reaction terms containing the most dominant reaction term M_d and all terms that are on par with M_d. The level of separation ρ used to determine whether terms are on par is a user defined quantity.

3.3.4 Obtaining modes of signal execution

Simulating biochemical models with different initial protein levels or kinetic parameters may result in different dynamic fingerprints. These different fingerprints can be compared to determine their level of discrepancy. We can then group sequences that have similar dominant subnetworks and define these groups of sequences as modes of signal execution. To accomplish this grouping, it is necessary to find a suitable distance metric that accounts for the differences of interest when comparing sequences. Once we have a metric, we can obtain a distance matrix which can be used with clustering algorithms to identify the modes of signal execution.

3.3.4.1 Defining a suitable distance measure for clustering

Multiple distance measures exist that can be used the calculate the dissimilarity between two sequences. Each of the distances have different sensitivities to sequencing, timing and duration of a dominant subnetwork. Since our goal is to identify groups of simulations that have the same dominant subnetwork, we chose the Longest Common Subsequence (LCS) distance (Bergroth et al., 2000). This metric is more sensitive to sequencing, i.e., the order of appearance of dominant paths in a signature. By focusing on the differences in the state distribution, it allows us to identify different modes of signal execution and novel protein targets that modulate biochemical signals within a network depending on the parameters of the model.

The LCS distance is defined as:

$$d_{LCS} = A(x,x) + A(y,y) - 2A(x,y)$$

Where $A_s(x, y)$ corresponds to the number of elements in one sequence that can be uniquely matched with elements occurring in the same order (not necessarily contiguous) in the other sequence.

3.3.5 Sequence analysis

In PyDyNo, we use the LCS metric to calculate the distance between all pairs of dynamic signatures and obtain a distance matrix. Then, we use this matrix with clustering techniques to identify groups of dynamic fingerprints that have similar patterns in the sequence of dominant subnetworks. PyDyNo includes three clustering algorithms: Agglomerative (Rokach & Maimon, 2005), spectral (Planck & Luxburg, 2006) and HDBSCAN clustering (Campello et al., 2013). We also added the Silhouette score function (Rousseeuw, 1987) from Scikit-learn to determine the optimal number of clusters. Finally, we implemented three algorithms described by Gabadinho and colleagues (Gabadinho et al., 2011) to identify representative dynamic signatures from each of the clusters.

3.4 Results and discussion

To validate our method, we used PyDyNo to analyze the signal execution in the JNK3 Activation Reaction Model (JARM) (N. A. Perry et al., 2019). JARM describes all the interactions between MKK4, MKK7, arrestin-3 and JNK3 that lead to the double phosphorylation and activation of JNK3. JNK3 activation has been linked to neurodegenerative diseases (Antoniou et al., 2011). JARM comprises 28 molecular species and 50 parameters. To do the analysis, we used the 5000 more likely parameter vectors from the model calibration described in Chapter 2.

We used PyDyNo to track how doubly phosphorylated JNK3 was generated. When multiple reactions produced a protein, we defined that dominant reactions were those within 0.5 orders of magnitude of the largest reaction rate. Given that we had 5000 simulations with their respective parameter vectors, we obtained 5000 dynamic fingerprints. We then employed the spectral clustering algorithm with the Silhouette score and identified that JARM has three execution modes. To visualize the differences in dynamic signatures, we used the *plot_sequences* function from PyDyNo (Figure 3.2). Each horizontal line depicted in Figure 3.2 corresponds to a sequence of dominant subnetworks. Each color represents a specific dominant subnetwork that contains information about how the signal is flowing throughout the network.

After identifying the three execution modes in JARM, we hypothesized that each execution mode would respond differently to the same perturbation in the system. To test this hypothesis, we performed a 50% in silico knockdown of MKK7, an essential protein in the JNK3 cascade that phosphorylates JNK3 in its threonine site. As shown in Figure 3.3, we observed that after the MKK7 knockdown, each execution mode displays different dynamics and total concentration of activated JNK3 at the end of the simulation. Specifically, in mode 1 the amount of activated JNK3 was reduced from 59% ± 1% in wild type to 35% ± 3% after the knockdown, in mode 2 the reduction was from 58% ± 1% to 41% ± 2%, and in mode 3 the reduction was from 58% ± 1% to 39% ± 4%. These results confirmed our hypothesis that execution modes exhibit different responses to the same perturbation in the system.

Altogether, we described how our method discretizes simulation dynamics and showed its usefulness to study signal execution in complex biochemical networks. PyDyNo combines network analysis and clustering algorithms and is particularly useful to analyze parameter uncertainties and their relationship with different execution modes. One advantage of our approach is that it uses signal flow, which results from the interplay between kinetic parameters and species concentration, rather than only the inferred kinetic parameters as other analyses have used previously (Yao et al., 2016). This approach enables us to consider the variability in species initial protein levels that explain cell-to-cell variability (Spencer et al., 2009; Strasen et al., 2018) and their role in defining different cell states. We anticipate that PyDyNo will be used with experimental design approaches to detect the characteristic execution mode of a system and reduce the uncertainty in predictions.

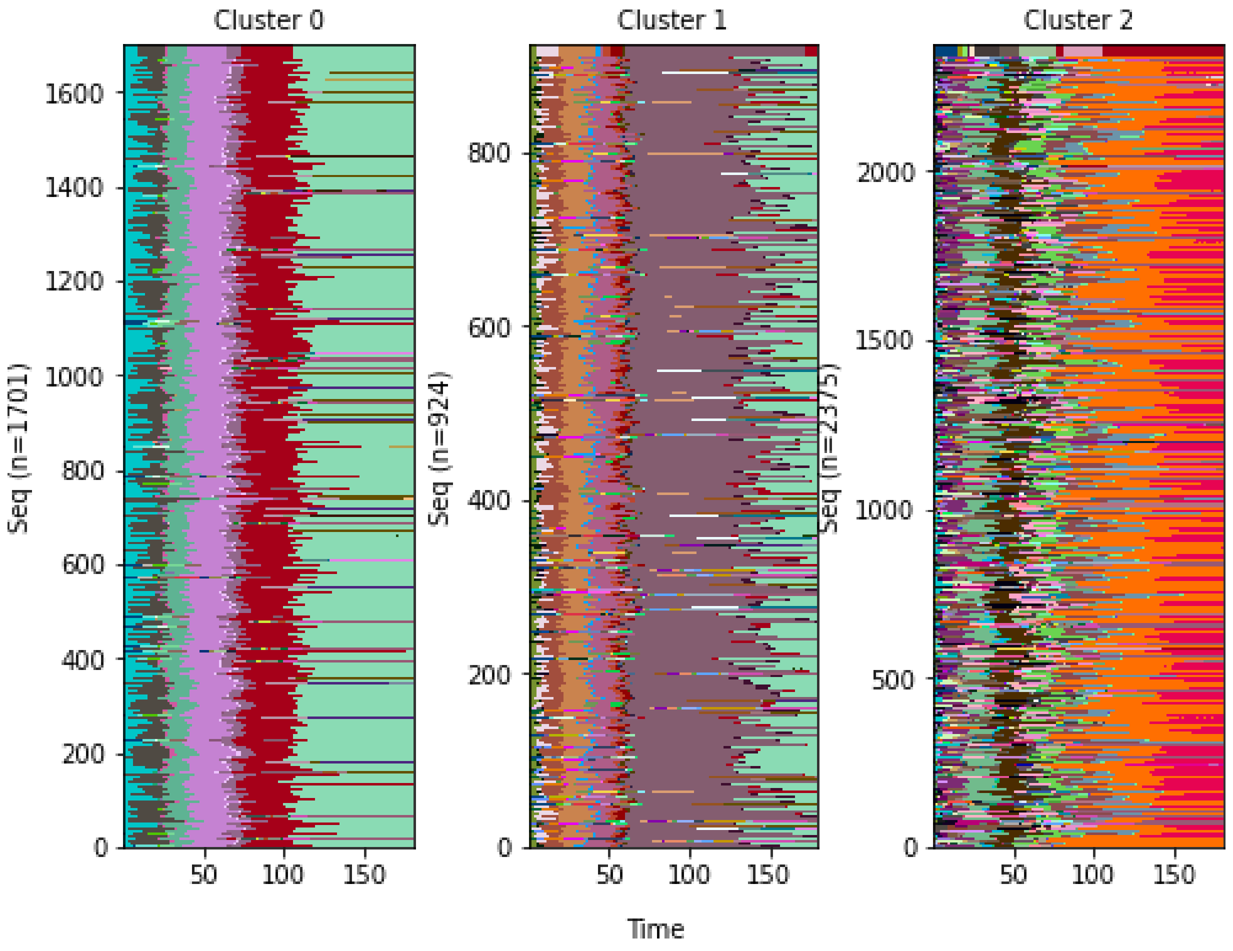


Figure 3.2 Modes of signal execution in JARM. Dynamic fingerprints organized by the clusters they belong to. Each cluster plot is composed of horizontal lines that correspond to dynamic fingerprints, i.e. sequences of dominant subnetworks, and each subnetwork is assigned a different color.

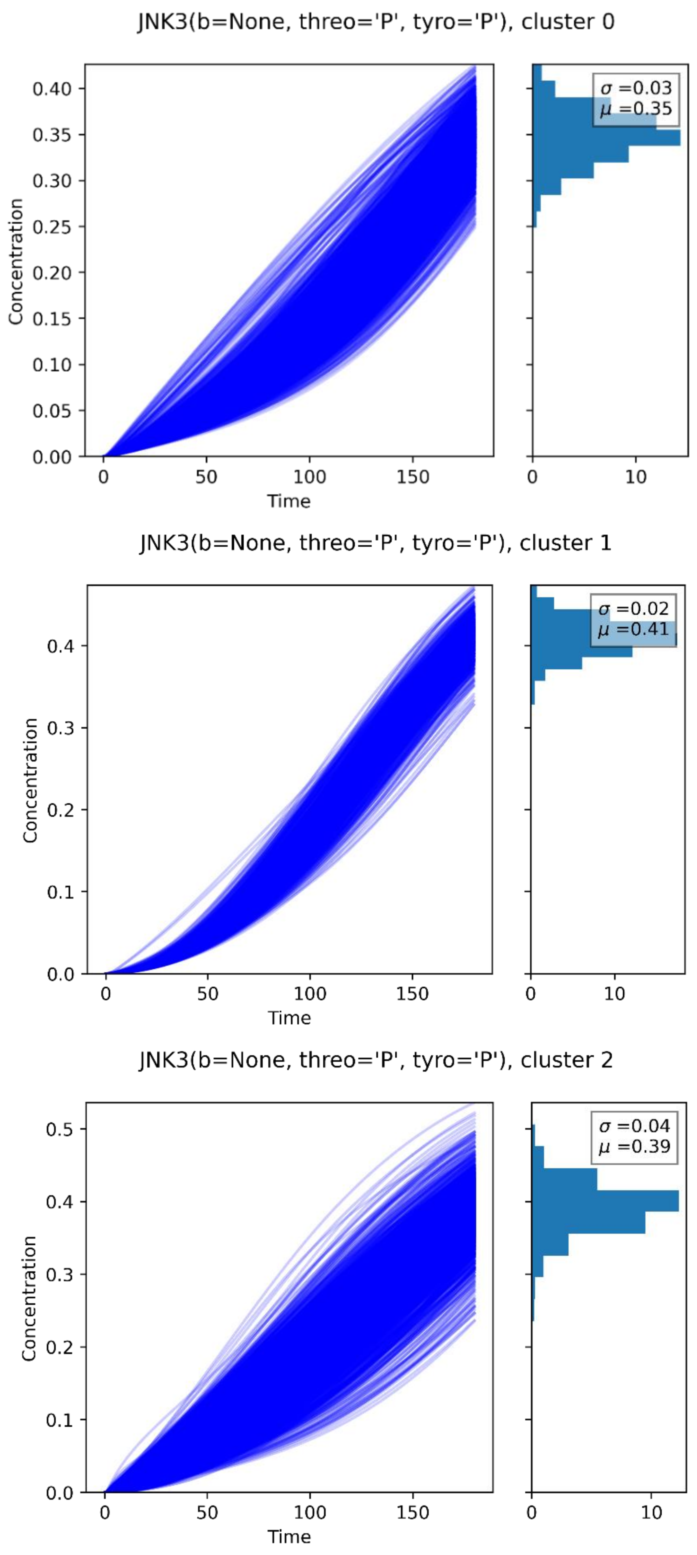


Figure 3.3 Amounts of activated JNK3 are markedly different for each execution mode after the 50% MKK7 knockdown. Protein concentration trajectories of activated JNK3 grouped by execution modes. Insets show the average and standard deviation of activated JNK3 at the last time point of the simulation.

Chapter 4

Probability-based mechanisms in biological networks with parameter uncertainty

Oscar O. Ortega, Blake A. Wilson, James C. Pino, Michael W. Irvin, Geena V. Ildefonso, Shawn P. Garbett, Carlos F. Lopez bioRxiv 2021.01.26.428266

4.1 Summary

Mathematical models of biomolecular networks are commonly used to study mechanisms of cellular processes, but their usefulness is often questioned due to parameter uncertainty. Here, we employ Bayesian parameter inference and dynamic network analysis to study dominant reaction fluxes in models of extrinsic apoptosis. Although a simplified model yields thousands of parameter vectors with equally good fits to data, execution modes based on reaction fluxes clusters to three dominant execution modes. A larger model with increased parameter uncertainty shows that signal flow is constrained to eleven execution modes that use 53 out of 2067 possible signal subnetworks. Each execution mode exhibits different behaviors to *in silico* perturbations, due to different signal execution mechanisms. Machine learning identifies informative parameters to guide experimental validation. Our work introduces a probability-based paradigm of signaling mechanisms, highlights systems-level interactions that modulate signal flow, and provides a methodology to understand mechanistic model predictions with uncertain parameters.

4.2 Introduction

Many biological processes can be represented as networks of interconnected biochemical components enabling the study of their dynamics and signaling mechanisms (Bonneau, 2008; K. A. Janes et al., 2008; Jordan et al., 2000; Weerts et al., 2018). These analyses typically entail building a network, either from prior knowledge or through network inference, developing a mathematical model of the network interactions, and subsequently calibrating the model to experimental data (Jaqaman & Danuser, 2006; Raue et al., 2011; Shockley et al., 2018). Although small networks have been studied with great success, the fact remains that for large networks many parameters remain difficult to ascertain and optimization routines yield multiple parameter sets that reproduce the protein concentration trajectories equally well (Eydgahi et al., 2013; Gutenkunst et al., 2007a; Mitra & Hlavacek, 2019b). This has led to a common practice whereby one or a few parameter vectors are chosen to make mechanistic predictions which can be validated by experiments with varying degrees of success (Albeck et al., 2008; Becker et al., 2010; K. A. Janes et al., 2005). However, criticisms remain regarding the usefulness of large and complex mathematical models of cellular processes with many uncertain parameters.

Information Theory based methods (Shannon, 1948) have been one of the successful approaches to date to explain input/output responses of intracellular signaling pathways (Brennan et al., 2012; Cheong et al., 2011; Levchenko & Nemenman, 2014; Shockley et al., 2019; Suderman et al., 2017). These approaches cast signal transduction in terms of channel capacities – the maximum amount of biochemical information that can travel from an input stimulus to an output response. These analyses have revealed that the maximum channel capacity of a biochemical process is context dependent and could require the cooperativity of multiple cells to achieve actual information transfer (Suderman et al., 2017). Previous work also applied information theoretic analysis to an allosterically regulated network and observed that the preferred path for information flow through the network was highly dependent on substrate concentrations (Shockley et al., 2019). Although these insights have been valuable to advance our understanding cellular regulatory processes, questions still remain about how reaction rates from non-equilibrium dynamics modulate signal flow in a biochemical network and further how these transient dynamics are impacted by parameter uncertainty.

Analysis of execution patterns in biochemical networks necessarily implies a detailed understanding of instantaneous fluxes throughout the system. However, reaction-flux based analysis is particularly challenging due to multiple concurrent biochemical interactions and their associated reaction rate fluctuations in time (Nobeli et al., 2009). Therefore, the number of interactions and temporal dynamics makes it difficult to establish whether persistent behaviors emerge from myriad biochemical reactions. Recent works in Tropical Geometry and Ultradiscretization Theory have proposed a mathematical formalism that makes it possible to map continuous functions into piecewise linear meshes in the Ultradiscrete space (Kato et al., 2017a). Approaches inspired in these novel mathematical treatments have been used to guide biochemical model reduction and simplification (Noel et al., 2011). For this work, we hypothesized dynamic analysis using these methods could enable us to identify dominant fluxes in a dynamic biochemical network, identify patterns of execution, and explore their dependence on model parameters. We reasoned that we could cast this analysis onto a Bayesian probability framework to assign statistical weight to the identified execution patterns, thus providing a novel statistical interpretation to cellular reaction mechanisms.

The remainder of the article is organized as follows. We first show how Bayesian inference of model parameters can yield tens of thousands of parameter vectors that all reproduce an experimental data set equally well. We then introduce a method, inspired in Ultradiscretization Theory and Tropical Algebra, to define a dynamic signal execution fingerprint, which can then be used to cluster execution modes according to model parameters. Surprisingly, we find that despite the thousands of parameter vectors that fit the experimental data of a biological system, only a handful of execution modes emerge as possible signal processing mechanisms. Subsequently, we demonstrate how parameter vectors that belong to different execution modes offer a biased view of signaling processes that could easily lead to misleading interpretations of network-driven processes. We further demonstrate how increases in parameter unidentifiability exacerbate the problem of model certainty in signal execution, but still identify the signal execution path probabilities associated with a given set of parameters. Our work therefore shows that network dynamics exploration, given available experimental data, could play a central role to identify true systems-level processes that shed light on signal processing mechanisms from a statistical perspective.

4.3 Materials and methods

4.3.1 Mathematical models of apoptosis

A coarse-grained ODE model of TRAIL-dependent apoptotic signaling (aEARM) was encoded with PySB (Lopez et al., 2013). This model grouped reactions into simple dynamic motifs representing key mechanistic blocks in TRAIL-dependent apoptosis pathway: TRAIL mediated DISC formation, initiator caspase-8 activation via the DISC and feedback activation via effector caspases (-3, -6, and -9), effector caspase activation, and apoptotic marker (PARP) cleavage were all encoded as simple catalysis reactions. MOMP formation (via initiator caspase-activated Bid) and accumulation of MOMP dependent pro-apoptotic signals were modeled as a Bid-dependent activation and subsequent feedback self-activation step. This activation-amplification motif reproduces the observed sigmoidal "snap-action" dynamics of MOMP-dependent pro-apoptotic effectors (e.g., Smac, CytoC). Initial values of the model components were drawn from values present in earlier apoptosis models (e.g., EARM). Initial values of the MOMP dependent signaling component took the same value as Bax and Bak in the EARM model. The ODE model was integrated using the Python LSODA ODE solver with relative and absolute tolerances set to 1e-2 and 1e-1 respectively (the model is encoded in copies per cell which takes values of 10,000 to 1M).

We also used the Lopez embedded Extrinsic Apoptosis Reaction Model (EARMv2.0) (Lopez et al., 2013) to analyze the resulting uncertainty from model calibration.

4.3.2 Experimental data

To calibrate aEARM and EARMv2.0 we used published data that contain the trajectories of the initiator reporter protein (IC-RP), and effector caspase reporter protein (EC-RP) (Spencer et al., 2009). In the model, the simulated trajectories from truncated Bid and cleaved PARP were fit to IC-RP and EC-RP, respectively.

4.3.3 Bayesian inference and parameter calibration

The model's reaction rate parameters were calibrated to normalized IC-RP and EC-RP fluorescence time-course data, using the Differential Evolutions Adaptive Metropolis

MCMC sampling algorithm (DREAM(ZS)) encoded in python as PyDREAM (Shockley et al., 2018). This Bayesian calibration uses as prior distributions log-normal distributions centered at biologically plausible rate values of forward binding, reverse binding and catalysis (1e-6 s-1 molecule-1, 1e-3 s-1, 1 s-1). The likelihood function assumes the IC-RP and EC-RP data are normally distributed with standard deviation calculated from multiple measurements. The sampling used employed a burn-in of 80,000 steps followed by 220,000 step sampling of the target distribution. Additional settings were applied to the gamma term in the DREAM algorithm: number of crossovers (nCR) = 25, adapt gamma = True, probability of gamma-unity (p_gamma_unity) = 0.1, resolution of gamma term (gamma_levels) = 8. Convergence was diagnosed by the Gelman-Rubin convergence diagnostic (i.e., GR ≤ 1.2) for each calibrated parameter. This calibration provided a wide range of kinetic parameter values and we note that even if there was experimental data for all species in a model, due to model sloppiness the parameter distributions would not be sufficiently constrained (Gutenkunst et al., 2007a).

4.3.4 Analysis of signal execution

To analyze signal execution throughout biochemical networks we used the method defined in Chapter 3. This method describes the steps to track the signal flow in the network and how to discretize the signal to obtain a dynamic fingerprint for simulations with different parameter vectors.

4.4 Results

4.4.1 Bayesian parameter optimization yields indistinguishable protein concentration dynamics.

To investigate the role of parameter uncertainty on signal execution through biochemical networks we focused on the extrinsic apoptosis form of programmed cell death (Elmore, 2007). Apoptosis is a ubiquitous biological process in metazoans used as a mechanism to maintain cell numbers and overall organism homeostasis (Koonin & Aravind, 2002). For the first part of our analysis we employed a modified version of the

Extrinsic Apoptosis Reaction Model (EARMv2.0) (Lopez et al., 2013). We found this abridged EARM (aEARM), depicted in Figure 4.1A, was the largest model we could build that would both preserve key biochemical interactions that represent extrinsic apoptosis, and in which all model parameters achieve convergence by the Gelman-Rubin diagnostics after parameter calibration with Bayesian methods. The model captures key biological features of apoptosis execution including signal initiation by TNF-Related Apoptosis Inducing Ligand (TRAIL), subsequent activation of initiator caspases (Caspase 8) (Kantari & Walczak, 2011) type 1 and type 2 activation of effector caspases (Caspase 3) (Özören & El-Deiry, 2002) and completion of apoptosis execution by cleavage of Poly(ADP-ribose) polymerase (PARP) (Kaufmann et al., 1993). Overall, aEARM comprises 22 molecular species and 34 kinetic parameters (see details in Methods). We used PyDREAM (Shockley et al., 2018) to calibrate the model to previously published experimental data that comprises the concentration dynamics of truncated Bid (tBid) and cleaved PARP (cPARP).

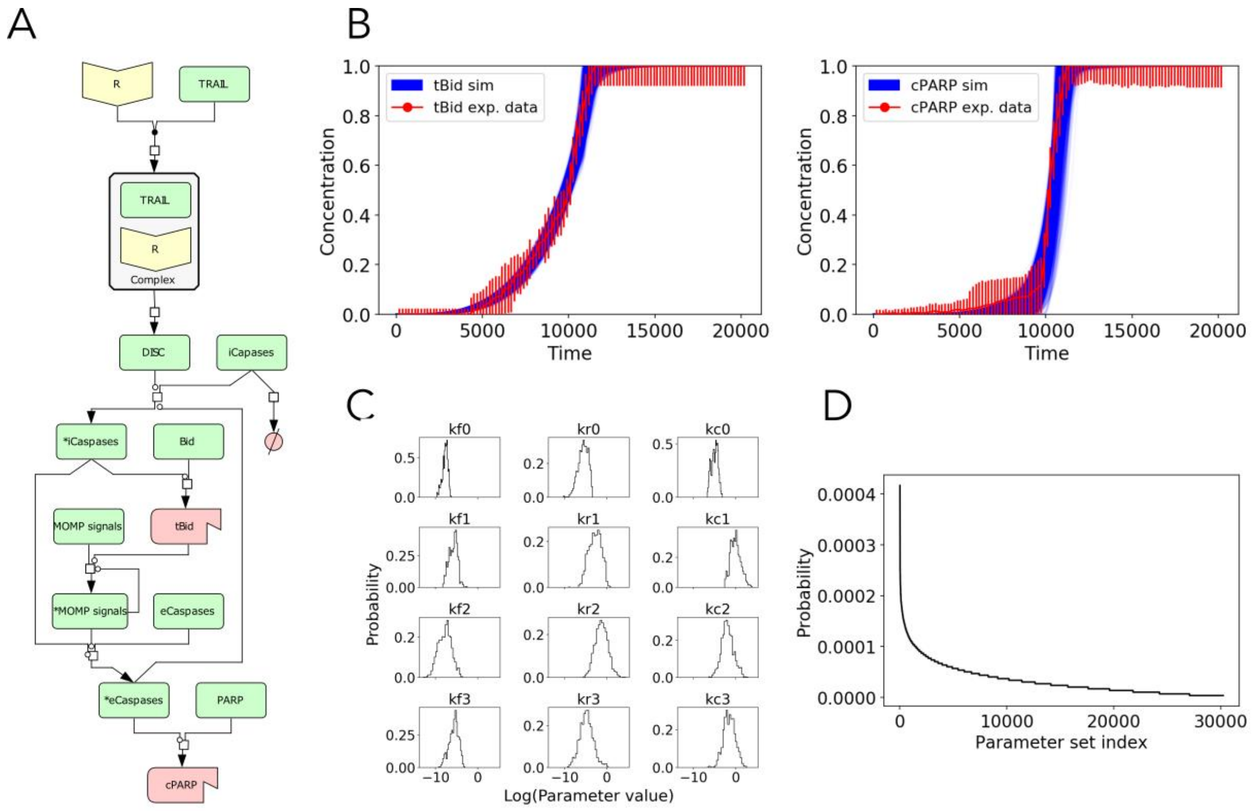


Figure 4.1 Abridged Extrinsic Apoptosis Reaction (aEARM) network and parameter calibration results. A) Reaction network using the Kitano convention. Yellow nodes are protein receptors, green nodes are generic proteins, and red nodes are truncated/cleaved proteins. B) Simulated trajectories of truncated Bid and Cleaved PARP calibrated to reproduce the experimental data. Red dots and bars indicate the mean and standard deviation of the experimental data and blue lines correspond to the simulated trajectories. C) Marginal probability distributions of the first 12 individual kinetic parameters that were recovered from the PyDREAM run by integrating out all other dimensions. Forward rates, reverse rate, and catalytic values were all found to be within biologically relevant ranges (H. X. Zhou, 2010). D) Probability of each of the unique parameter vectors sampled after burn-in in the PyDREAM calibration. To obtain the probability of each parameter set the number of visits to a specific parameter vector was normalized by the total number of visits.

Given that the model was calibrated to HeLa cell data, we hypothesize that signal patterns are representative of signal processing and execution of Type-II cells treated with death-inducing ligands such as TRAIL. In all, we ran the PyDREAM sampling for 100,000 steps after burn-in and collected 300,000 parameter vectors from which 27,242 were unique. All unique parameter vectors fit the data equally well (Figure 4.1B). All parameters were deemed to have converged by the Gelman-Rubin diagnostics (Gelman & Rubin, 1992). We obtained the marginal distributions from the sampled parameter vectors as show in Figure 4.1C. Given the Markov Chain Monte Carlo (MCMC) aspect of our parameter inference method, we were able to obtain parameter vector probabilities as shown in Figure 4.1D (Chiband & Greenberg, 2008). The probability distribution of parameter vectors exhibits characteristic exponential-like decay shape indicating that some parameters are more likely than others. With this calibrated model to experimental data, we then probed signal execution patterns in the aEARM network from a probabilistic perspective. We note that throughout the manuscript, a parameter vector refers to a set of positive real values, one value for each of the kinetic parameters defined in aEARM, used to run a simulation. A parameter distribution refers to the frequency of occurrence of different values from the same kinetic parameter.

4.4.2 A discretized flux-based analysis of signal execution in networks.

As shown in Figure 4.1B, all the parameter vectors obtained from the Bayesian calibration yield protein concentration dynamics indistinguishable from the experimental trajectories of tBid and cPARP. Individual parameters from these vectors take widely different values as depicted by their distributions in Figure 4.1C. This uncertainty in the parameter values affects the reaction rates of the protein interactions generating different reaction flux patterns in the network during signal execution. We therefore wanted to study the non-equilibrium flux of the reactions in the aEARM network and aimed to explore whether parameter uncertainty yielded specific patterns of signal execution.

Analysis of flux dynamics during signal execution requires tracking the signal flow through a network at all simulation time points as multiple concurrent reaction rates consume or produce molecular species. We assumed that the reactions with the highest flux at any given time dominate the network signal execution and provide a proxy to observe the effect of different parameter vectors in the network. Our aim was thus to identify the reaction rates with the highest flux throughout the whole network as simulations evolved over time. To analyze the non-equilibrium flux and find the dominant reaction paths during signal execution, we developed an algorithm inspired by Ultradiscretization Theory and Tropical Algebra as described in *Methods* (Kato et al., 2017b; Noel et al., 2011). Our approach enabled us to identify paths relevant for flux propagation in non-equilibrium states. We refer to these paths of flux propagation through the network as *execution modes* for the remainder of this manuscript.

We introduce the workflow for reaction flux discretization and execution mode identification as shown schematically in Figure 4.2A-B. Signal discretization requires three steps. First, we identify a target node (Fig 4.2B) for which the signal flux will be tracked. Second, we calculate the reaction rates that produce or consume the target node, identify the largest reaction rate (x) and test whether it is dominant over other reactions (y) using the discretization operation $|\log_{10} x| - |\log_{10} y| > \rho$, where r is the order of magnitude difference necessary to consider dominance (see Methods section for details). Third, we identify the chemical species produced by the dominant reaction(s) and jump to that species, thus starting the process again from the first step, and thereby tracking the dominant signal fluxes through the whole network and obtaining a subnetwork. This dominant subnetwork is assigned a unique integer label as shown in Figure 4.2A. The procedure is repeated for all simulation time points. As a result, the dynamic nature of signal execution for a given parameter vector is abstracted to a sequence of labels that can be compared to other sequences using a suitable metric (Figure 4.2B). We call this sequence of labels obtained from a simulation a *dynamic fingerprint* because it is unique for a given signal processing event with a specific parameter set.

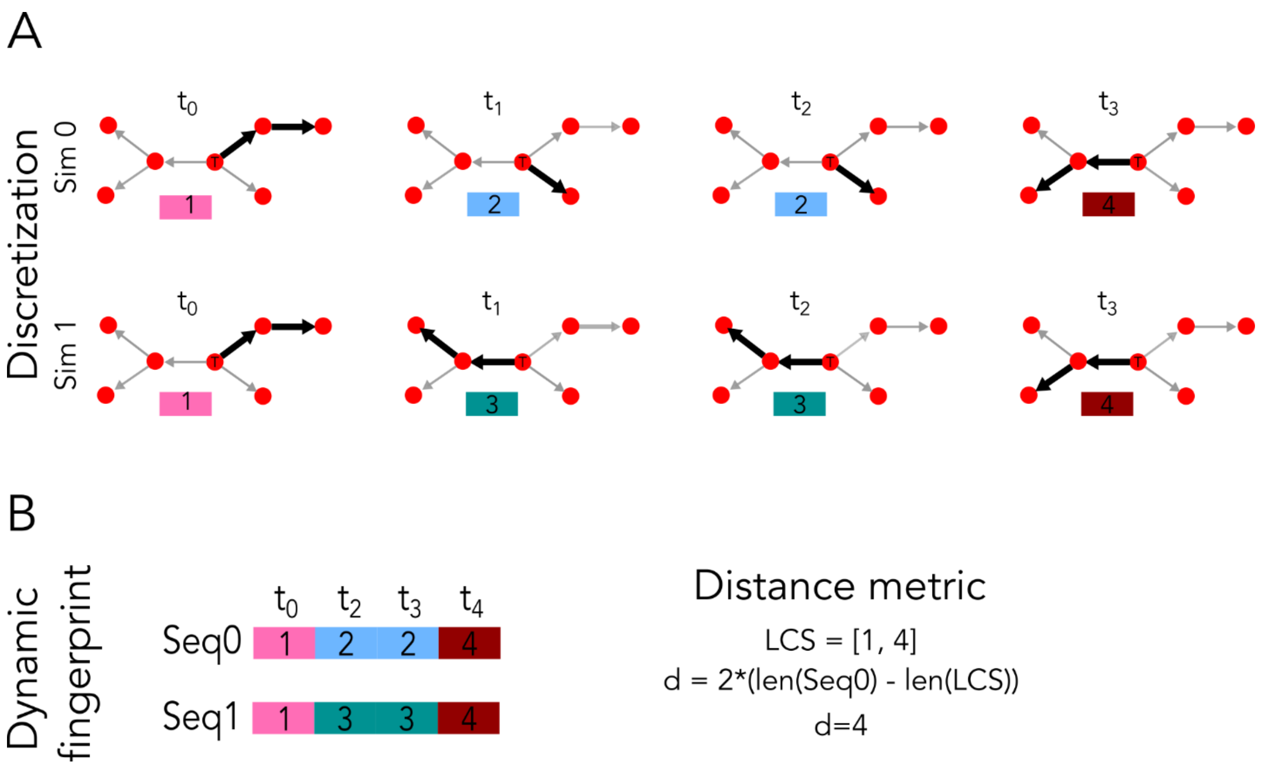


Figure 4.2 PyDyno workflow. A) First, the network of interaction is obtained from a model and a target node (labeled T) from where the signal is going to be tracked is defined. Red nodes are molecular species in a model, edges represent interactions between nodes, bolded edges are the dominant interactions. Next, at each time point of a simulation our analysis obtains a dominant subnetwork, bolded edges in the network, through which most of the signal is going through and this subnetwork is assigned a label. Sim 0 and Sim 1, simulations ran with different parameter sets, exhibit different dominant subnetworks. B) As each subnetwork is assigned a label, we can get a sequence of labels per simulation that can be compared to other simulations with the Longest Common Subsequence metric and obtain a distance matrix. This distance matrix can be used with clustering algorithms to obtain groups with similar modes of signal execution.

4.4.3 Key execution modes emerge despite parameter uncertainty

To identify the dynamic execution patterns in aEARM in response to death ligand cues, we carried out our signal discretization analysis for the 27,242 unique parameters and obtained dynamic fingerprints for each parameter vector. We then asked whether there were similarities among dynamic fingerprints across parameter sets. To investigate this question, we quantified the distance between each dynamic fingerprint using the Longest Common Subsequence (LCS) metric. We chose this metric due to its sensitivity to order differences in which successive subnetworks labels appear (Studer & Ritschard, 2015). This metric thus assigns a larger distance to a pair of dynamics fingerprints that execute the signal differently. Next, we calculated the pairwise distance between all dynamic fingerprints obtaining a 27,242 by 27,242 distance matrix. This matrix enabled us to use an agglomerative clustering algorithm (Rokach & Maimon, 2005) to probe whether clusters of dynamic fingerprints would emerge. As shown in Figure 4.3A, we found that all 27,242 dynamic fingerprints could all be classified into three clusters (Supplemental Table 2), which we denominate "execution modes". Given that each parameter vector has a defined probability (Figure 4.1D) and is associated with a dynamic fingerprint, we could calculate the probabilities of signal execution through each mode as 42%, 36%, and 22% for Execution Mode 1 (EM1), Execution Mode 2 (EM2), and Execution Mode 3 (EM3) respectively. These three execution modes account for all the parameter vectors inferred from the explored probability space and no vectors were found that did not belong to either of these modes. We note that these execution modes are comprise three subnetworks out of eight possible subnetworks for signal flow.

The dominant flux subnetwork for each execution mode is shown schematically in Figure 4.3B. We note the highlighted paths represent the dominant reaction fluxes, i.e. these fluxes are within an order of magnitude of the largest reaction at each node for the given parameter set and simulation time point. As shown, Execution Mode 1 (EM1) comprises events from initial death-ligand binding to the receptor, through formation of the Death Inducing Signaling Complex (DISC), and subsequent activation of initiator Caspase. The initiator Caspase then truncates and activates Bid, which in turn activates

MOMP, a species that abstracts mitochondrial outer membrane pore (MOMP) formation. Activated MOMP can then further activate MOMP in a positive feedback loop and activate the effector Caspase downstream. As highlighted in Figure 4.3B(EM1), activated MOMP is dominantly used to both activate more MOMP, through the positive feedback loop, and activate the effector Caspase.

The flux through the network in Execution Mode 2 (EM2) is similar to that of Mode 1 but the execution path differs at MOMP regulation. As highlighted in blue in Figure 4.3B EM2, activated MOMP is largely consumed in the positive feedback loop to activate more MOMP. The signal flux downstream of activated MOMP is at least an order of magnitude less than the highlighted route for the parameters in EM2. Therefore, effector Caspase activation and apoptosis execution takes place due to a smaller reaction flux in the network relative to the MOMP-level activity in EM2. For those parameters belonging to EM3, signal execution seems to flow largely toward PARP cleavage, with less MOMP-level regulation. Our results therefore show that despite uncertainties in inferred model parameters due to limited available data, the modes of signal execution are identifiable. Identifying a limited number of execution modes highlights the need to thoroughly characterize the model parameter space, given experimental constraints, to understand and make inferences about execution mechanisms. We note that using a single vector of parameters would lead to incomplete model prediction as no one single parameter vector captures the rich dynamics exhibited by all the statistically inferred parameter vectors.

To further understand the impact of each execution mode on MOMP regulation, we examined the relative concentration of activated MOMP (MOMP*) and the binding complexes in which it participates. We calculated the percentages of MOMP*, inactive MOMP bound to MOMP* (MOMP-MOMP*), and effector caspase bound to MOMP* (EC-MOMP*). As shown in Figure 4.3C, the relative abundance of these species over time exhibits different concentration patterns in each execution mode. In EM1, the relative abundance of EC-MOMP* is ~20%, indicating that the signal flow through this reaction is lower than in EM3 but still important in the overall dynamics. In EM2, 85 % of MOMP* is bound to inactive MOMP at all time points before cell death. This can be explained by a high MOMP activation rate due to Bid and the MOMP* positive feedback loop autoactivation. In contrast to EM2, the MOMP-MOMP* abundance in EM3 decreased to ~35%, while EC-MOMP* is increased to ~50%. This increase in EC-MOMP* abundance, indicates that the binding rate of MOMP* to EC is larger than the binding rate of MOMP to MOMP*. We note that the initial concentration of inactive MOMP in the model is an order of magnitude larger than that of EC. Thus, this result is in stark contrast with the result from EM2 where the reaction rates exhibit different relative values.

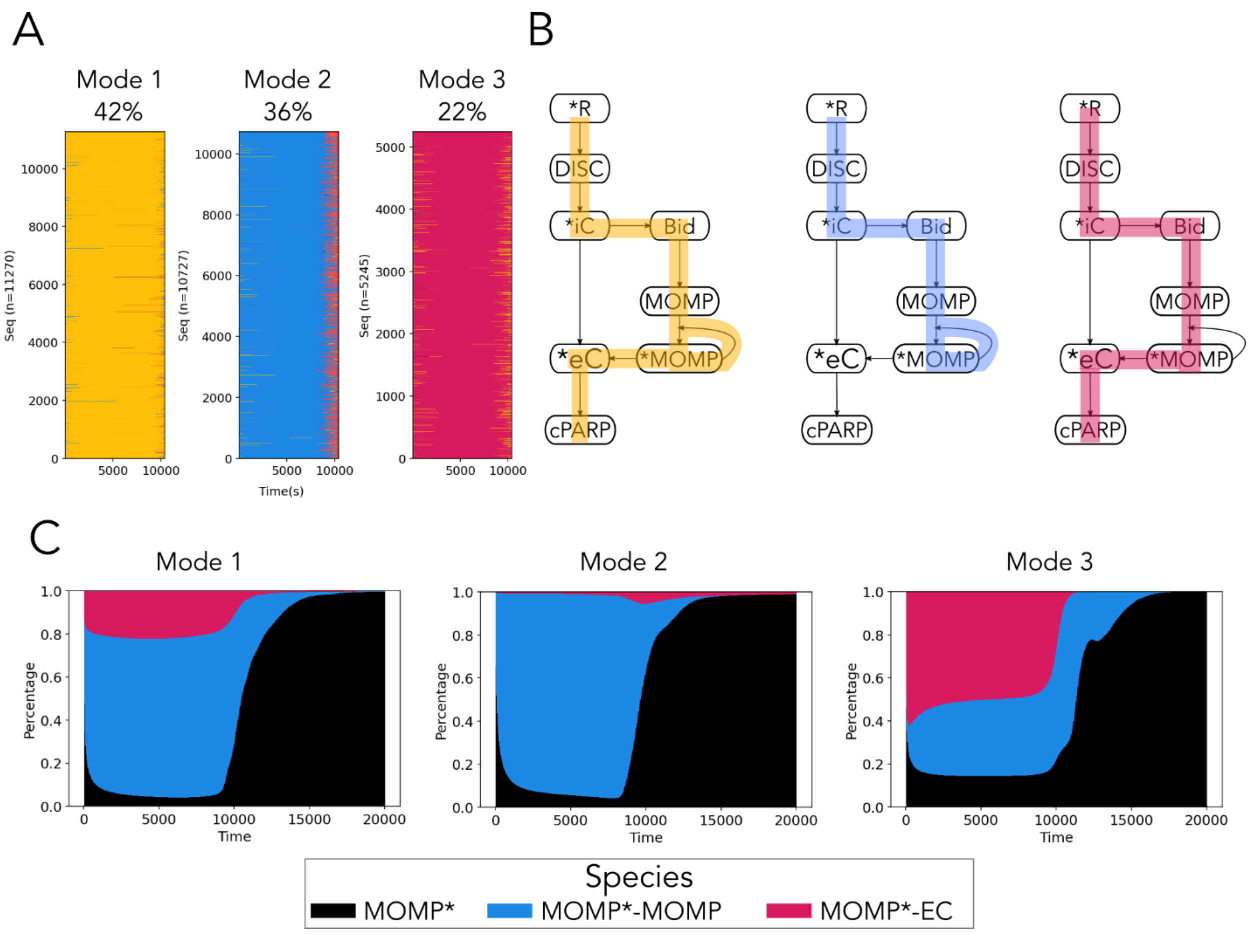


Figure 4.3 Modes of signal execution in aEARM. A) Dynamic fingerprints organized by the clusters they belong to. Each cluster plot is composed of horizontal lines that correspond to dynamic fingerprints, i.e. sequences of dominant subnetworks, and each subnetwork is assigned a different color. B) Signal execution modes as defined by the most common subnetwork in each cluster. The complete aEARM network is shown in black, and the dominant subnetworks for Mode 1, 2, and 3 are highlighted in yellow, blue, and red, respectively. C) Effect of the different signal execution modes in the relative concentration of activated MOMP and its associated complexes. For each cluster, we calculated the temporal relative concentration of MOMP*, MOM*P-MOMP and MOMP*-EC point by obtaining their individual average concentrations and dividing it by the sum of their concentrations. This visualization provides insights about the usage MOMP* in each cluster.

4.4.4 Signal execution modes respond differently to eCaspase perturbation

We then asked whether in-silico experiments could help us understand differences in signal execution that could lead to experimentally testable hypotheses. We therefore carried out *in-silico* knockdown experiments of eCaspase, as its activation is essential for the final steps of apoptosis execution (Mehal et al., 2006). In addition, effector caspase inhibitors are readily available for laboratory use (D. K. Perry et al., 1997; Solania et al., 2019). We hypothesized that each execution mode would exhibit different execution mechanisms when eCaspase was knocked down by 50%. To explore the impact of eCaspase knockdown for each execution mode, we compared the concentration dynamics for MOMP and cPARP given by wild type and eCaspase knockdown conditions.

For each execution mode we plotted the cPARP concentration trajectories and obtained the time of death (ToD) for each simulated cell as described in Methods. As shown in Figure 4.4A, the ToD in EM1 exhibits a modest decrease of 14.96 s, but also presents a larger standard deviation of 702 s. For EM 2 the ToD increased from 10351 ± 132 s (WT) to 10809 ± 226 s for eCaspase knockdown (Δt = 458 s). In contrast, EM3 eCaspase knockdown leads to a decreased ToD from 10261 ± 83 in WT to 9507 ± 516 s in the knockdown (Δt = - 754 ± 523 s) These results therefore show that each execution mode can exhibit significantly different – and a times juxtaposed –responses to the same perturbation.

We then probed the effect of the eCaspase knockdown on the reaction rates associated to MOMP* (a node where the signal bifurcates): MOMP* binding to MOMP, and MOMP* binding to EC. Specifically, we focused on the reaction rate peak and the time to reach peak of the reaction rate throughout the simulation, as shown in Figure 4.4B and supplementary Figure 4.2. The peaks of the MOMP*+MOMP binding reaction (Figure 4.4B upper row) appear unchanged across all execution modes, yet the time to reach the peaks vary significantly. The median time to peaks were 6.14%, 0.36%, and 11.76% faster for modes 1, 2, and 3, respectively. Concurrently, the peaks of the MOMP*+EC binding reaction (Figure 4.4B lower row) are reduced approximately 50% as expected by the 50% reduction of the available EC, and the median time to peaks were 6.77%, 0.4%, and 14.48% faster for modes 1, 2, and 3, respectively. In combination, for mode 1, the relative change of the

MOMP and EC reaction peaks have large interquartile ranges IQR= -10.37% to –1.01% and IQR=-1.13% to –11.39%, respectively, which explains the variability in the time to cell death. For mode 2, the time to the peak of MOMP and EC reactions change marginally and given that the EC peak is 50% of the WT condition, this leads to longer times to accumulate the necessary number of EC molecules for cells to commit to apoptosis. Finally, for mode 3, the median time to reach the MOMP reaction peak and the EC reaction peak is 11.76% and 14.48%, faster than in the WT condition, respectively. This causes faster activation of MOMP and EC which leads to earlier apoptosis in cells. To summarize, although the biochemical signal flows differently in each execution mode, the protein concentration dynamics exhibit similar outcomes (Figure 4.4A Wild Type). However, when a perturbation is made to the network, the outcome can vary significantly, as shown for each execution mode.

4.4.5 Reducing execution mode uncertainty through parameter measurements

Given that the aEARM calibrated parameter vectors yield three execution modes with their respective probabilities, there is uncertainty about which execution mode is most representative of the cellular process. We then asked whether we could identify parameters that, if measured experimentally, would reduce the execution mode uncertainty. We hypothesize that identifying key parameters that inform execution mechanisms could guide experiments to improve our knowledge about network-driven signal processing. To measure the uncertainty of the execution modes, we used Shannon's entropy $H = -\sum_{i=1}^{n} P(x_i) \log_2 P(x_i)$ (Shannon, 1948). As aEARM has 3 execution modes the maximum entropy in the system is $\log_2 3 = 1.58$, which would signify that each execution modes have a 33% probability. Using the probabilities of the previously obtained execution modes (Figure 4.3A) and Shannon's formula we calculated an entropy of 1.54 indicating a high uncertainty in the execution across all modes

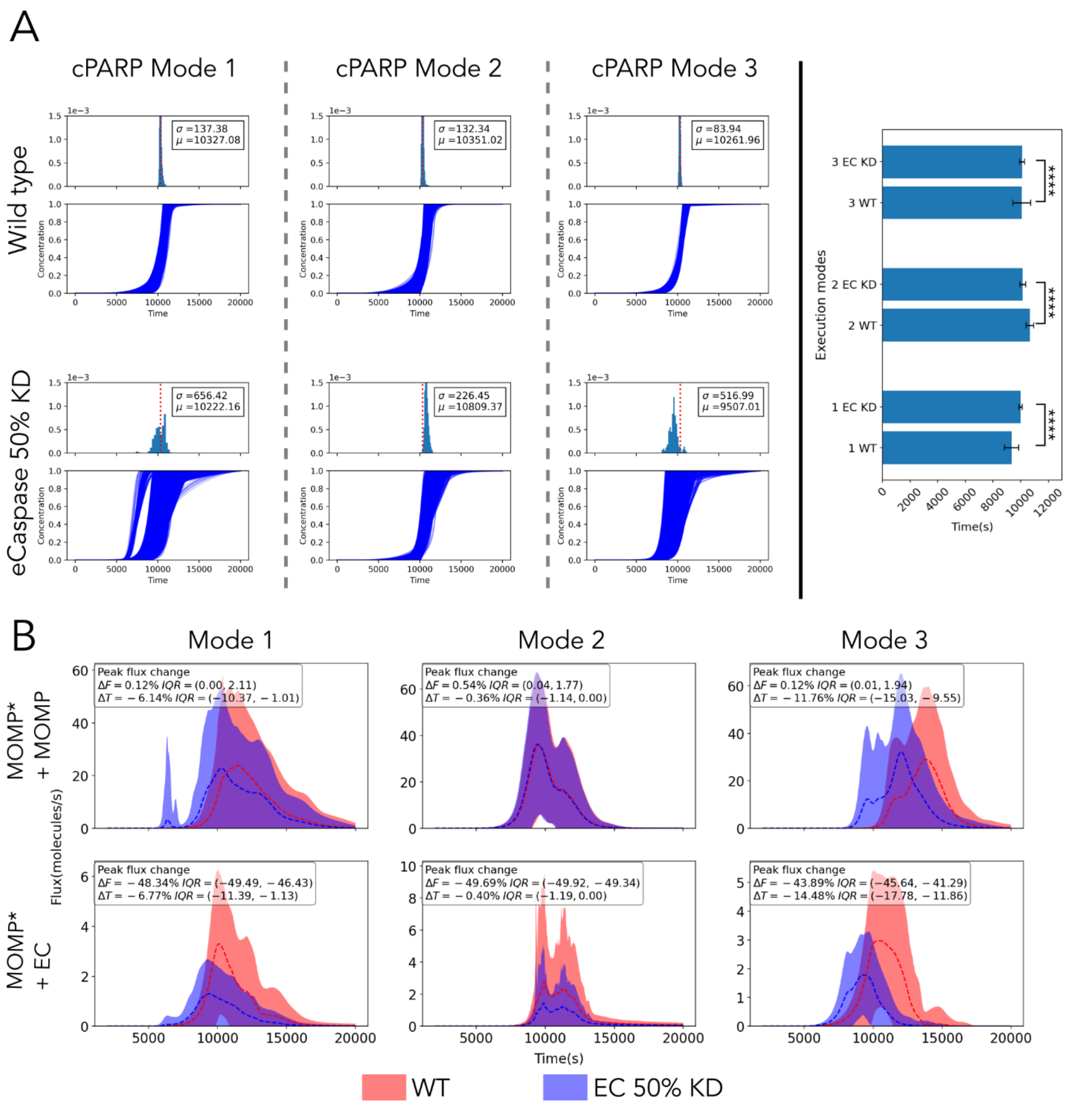


Figure 4.4 Time of death responses are markedly different for the same perturbation. A) Cleaved PARP (cPARP) protein concentration trajectories for the “wild type” case (top row) grouped by Execution Modes. Mode 1, 2, 3 have 11270, 10727, and 5245 trajectories, respectively. Inset includes the average time to death and the standard deviation calculated from all trajectories in each execution mode. PARP cleavage exhibits a markedly different trajectory pattern (bottom row) after eCaspase is knocked down by 50%. B) MOMP* + MOMP and MOMP* + EC reaction rate trajectories. Dashed lines correspond to the mean of all reaction rates trajectories in an execution mode and the shadows represent the standard deviations. Trajectories from the “wild type” condition are colored in red and trajectories from the 50% effector caspase KD are colored in blue, and show key differences in their dynamics. Insets include the median percentage change in the reaction rate peak (ΔF) and the time to reach that peak (ΔT) in the EC KD condition relative to the wild type condition. The interquartile range is included as a measure of the variation in the ΔF and ΔT changes.

To determine the most informative parameters that should be measured to reduce execution mode uncertainty, we used XGBOOST (T. Chen & Guestrin, 2016), a gradient boosted Machine Learning technique that can classify parameter vectors into their corresponding execution modes. We used the calibrated parameter sets as training data where each individual kinetic parameter (kf, kr, kc) is a feature, and the mode of execution is our target variable.

Feature importance analysis from the XGBOOST analysis shows that parameters kf7 and kf6 contribute the most to training loss reduction during the classification task (Figure 4.5A). As illustrated in Figure 4.5C, parameters kf6 and kf7 correspond to the binding rate of MOMP* to inactivated MOMP, and MOMP* to EC, respectively. These two parameters are part of the reactions where the signal flux is bifurcated in the network, indicating that their values play an important role in the definition of the execution modes. To show the differences in parameter values for each execution mode we plotted the values of the kf6 and kf7 parameters. As shown in Figure 4.5B the execution modes have different distributions of the kf6 and kf7 parameters with some overlap. As depicted in Figure 4.5D, we simulated 100 measurements of the kf7 parameter and found that these measurements have various degrees of entropy reduction. Therefore, measuring MOMP-related parameters could help further reduce execution mode uncertainty and improve model-based predictions.

4.4.6 Modes of signal execution in a detailed apoptosis model with increased parameter uncertainty

Based on our results with aEARM, we then asked how a larger model with higher parameter uncertainty would fare under the presented signal execution analysis. We shifted to a larger extrinsic apoptosis reaction model (EARM V2.0), which has been studied and characterized in previous work (Lopez et al., 2013). As illustrated in Figure 4.6A, EARMV2.0 is considerably larger than aEARM as the biochemical interactions are described with higher molecular resolution. In all, EARM V2.0 has 77 molecular species and 105 kinetic parameters. As described in Methods, we used PyDREAM to calibrate the model to published experimental data (Spencer et al., 2009). Although, the calibration yielded parameter vectors that fit the experimental data indistinguishably well (supplemental Figure 4.4), we note that only 62 model parameters converged according to the Gelman-Rubin diagnostic ($GR < 1.2$) after two million iterations (see Supplemental Table 3 and Supplemental Figure 4.5). Distributions of 9 converged parameters are shown in Figure 4.6B. The remaining parameters exhibited GR values between 1.21 and 13.52. From a Bayesian perspective, non-convergent parameters imply that the experimental data simply cannot constrain their values to a distribution and thus results in higher variability. As our analysis is focused on understanding execution modes in network-driven processes, a model with poorly identified parameters presents an opportunity to explore how signal execution could be best interpreted and understood in large model systems with high parameter uncertainty.

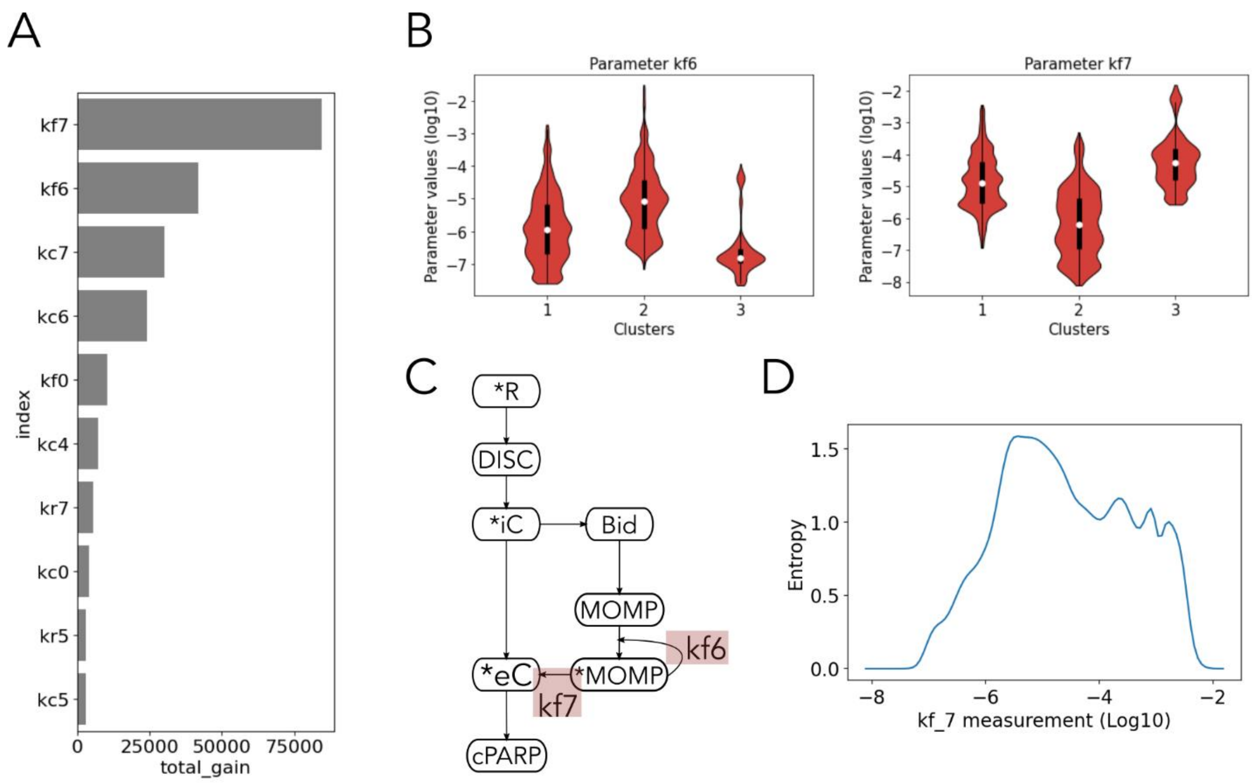


Figure 4.5 Parameter measurements reduce execution mode uncertainty. A) List of the 10 parameters that contribute the most to model prediction. Parameters with higher total gains, compared to another parameter, provide larger improvements to accuracy in model prediction. B) Parameter values of kf6 and kf7 grouped by the execution mode they belong to. A Gaussian kernel was used to estimate the density probability of parameter values in each execution mode. C) Schematic representation of the aEARM network. Kinetic parameters kf6 and kf7 and their corresponding reactions are highlighted in the network. D) Changes in the execution modes entropy after simulated measurements of kf7.

We followed the same procedure used in the previous sections to explore the execution modes in EARM V2.0 (See Methods for details). Our analysis found that calibration to the experimental data constrains the signal flow to eleven execution modes, that can be represented by 53 dominant subnetworks out of 2067 possible subnetworks. As shown in Figure 4.6C, the apoptosis execution signal could flow through any of these paths with varying degrees of probability, with Execution Mode 1 (EM1) exhibiting a probability of ~20 and the first four modes capturing ~50% of the signal probability, thus suggesting high path entropy as we have seen in previous work (Shockley et al., 2019). Videos can be found in the supplement that show animations of signal flow for all execution modes in the context of EARM V2.0.

Next, we tested whether each execution mode exhibits different responses to the same perturbation. We selected EM1 and EM2 for analysis as these modes exhibit the highest probability for signal execution. As illustrated in Figure 4.6D, the mBid interaction with Mcl1 is dominant in EM1. In contrast, the mBid interactions with Mcl1 and Bcl2 are both dominant in EM2, thus highlighting the importance of both antiapoptotic proteins to understand the signaling mechanisms during apoptosis execution during the cell response to an apoptotic inducer.

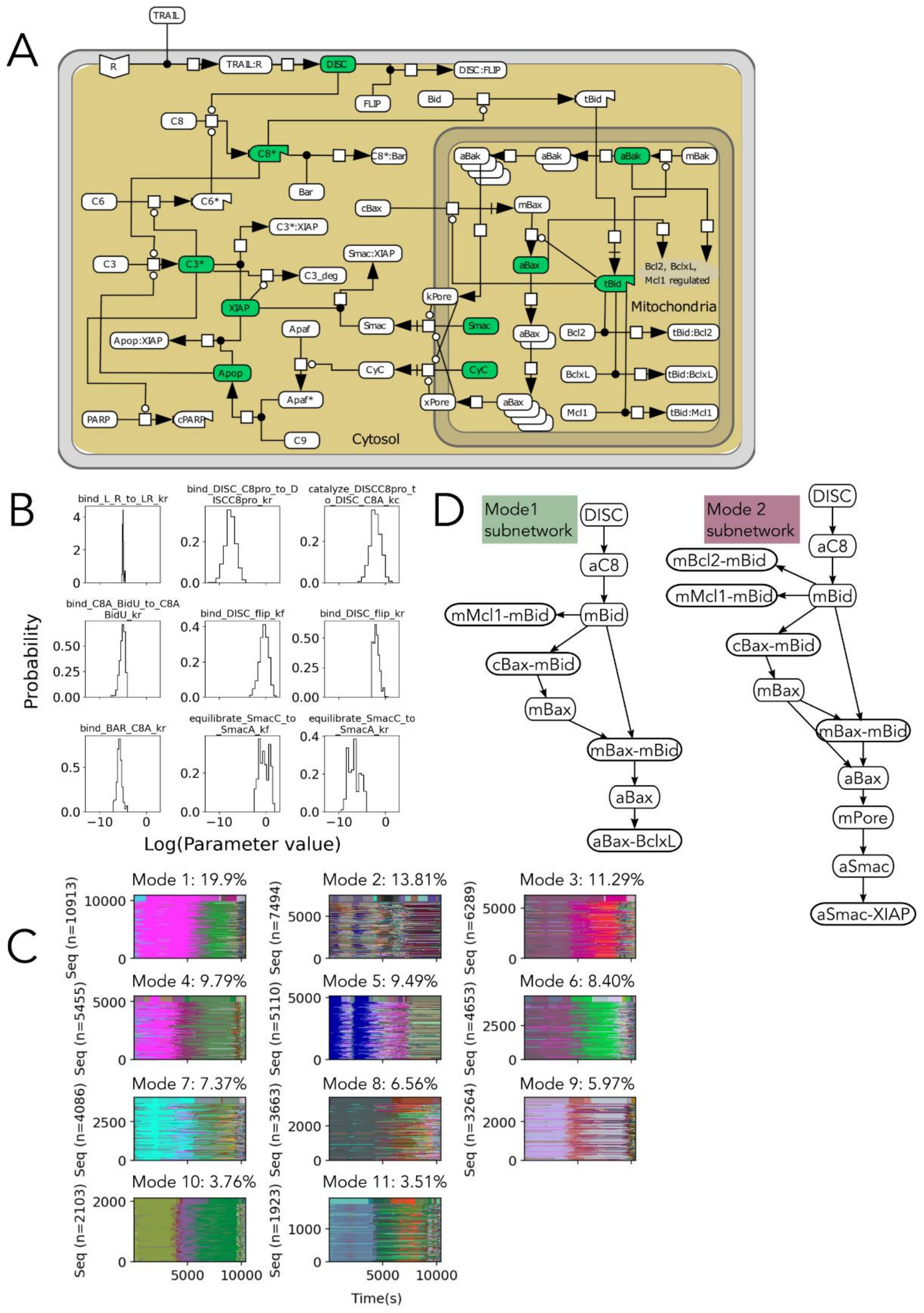


Figure 4.6 Modes of signal execution in the full Extrinsic Apoptosis Reaction Network. A) Network of the interactions between the proteins in the apoptosis pathway. Proteins highlighted in green are nodes where the signal flux can be divided. The convention of Kitano (Kitano et al., 2005) was

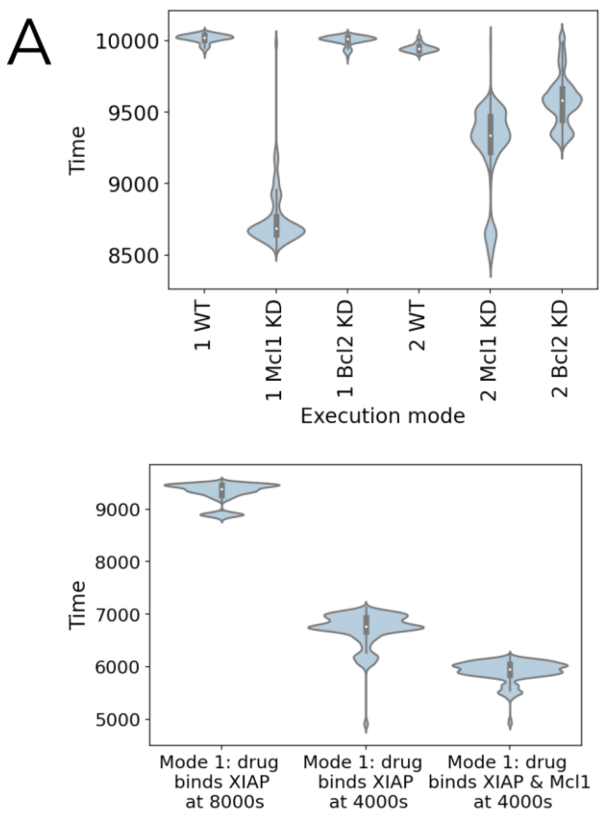
followed. B) Marginal probability distributions of 9 individual kinetic parameters converged by the Gelman-Rubin diagnostic. C) Dynamic fingerprints organized by the execution modes they belong to. Each cluster plot is composed of horizontal lines that correspond to dynamic fingerprints, i.e. sequences of dominant subnetworks, and each subnetwork is assigned a different color. Execution modes are sorted from highest to lowest probability. D) Signal execution in Mode 1 (left) and Mode 2 (right) as defined by the most common subnetwork in each mode at t=7000s.

We then performed two *in silico experiments* for EM1 and EM2: (*i*) a 50% knockdown (KD) of the antiapoptotic protein Mcl1 as well as (*ii*) a 50% knockdown of the antiapoptotic protein Bcl2. For the Mcl1 KD, we found that the EM1 median ToD decreased from 10022.46 s (WT) to 8686.52 s (Figure 4.7C-upper panel). This is expected since mBid and Mcl1 interactions are dominant in this mode. By contrast the median ToD in EM2 decreased from 9943.65 s (WT) to 9335.85 s. This modest decrease in ToD can be attributed to the fact that although Mcl1 and mBid interactions are important in EM2, the dominance of Bcl2 compensates for the absence of Mcl1 and reduces the impact on ToD for the Mcl1 KD.

For the Bcl2 KD, we found that the median ToD in EM1 has a minor change from 10022.46 s to 10011.87 s, expected because mBid activity is not significantly affected by Bcl2 in this mode. By contrast, in EM2, mBid activity is modulated by Mcl1 and Bcl2. Thus, a reduction in the initial protein levels of Bcl2 enables more mBid proteins to activate pro apoptotic proteins and this leads to an increase in ToD to 9580 s (Figure 4.7C-upper panel). Taken together this data shows that distinct execution modes respond differently to the same perturbation and that their responses can be predicted based on the dominant reactions for a given execution mode.

To further emphasize the importance of transient dynamics on signal processing, we explored EM1 dynamic fingerprints and found that SMAC inhibition of XIAP occurs at later time points of the simulations (>8640 s). Therefore, we hypothesized that XIAP inhibition would be more effective earlier during signal execution. To test this, we added an XIAP inhibitor to EARMV2.0 at either 4000 s or 8000 s. As shown, when the inhibitor is added at the later time point, we observed a small reduction (Figure 4.7A lower panel) in the median ToD from 9943.65 s in the WT to 9380.44 s ($\Delta t = 563.21$ s). In contrast, when the inhibitor is added at the earlier time point, when SMAC is not yet released from the mitochondria, the inhibitor binds to XIAP enabling C3 to cleave PARP and thereby reducing the median ToD to 6766.10 ($\Delta t = 3177.55$ s).

As combination therapies have become important to combat drug resistance (Gayvert et al., 2017; Sarah, 2017), especially in cancers, we explored whether our analysis provided information about potential targets for cotreatment. As we previously mentioned, Mcl1 and XIAP are dominant antiapoptotic proteins in EM1, thus we hypothesized that inhibition of both proteins would yield a shorter ToD compared to only inhibiting XIAP. To test this, we added two drugs that independently inhibit XIAP and Mcl1 and obtained a ToD of 5951.11 s representing a 12% reduction in the ToD compared to XIAP inhibition only (Figure 4.7A lower panel). Finally, to guide experiments that would identify the most likely execution mode out of the 11 execution modes obtained, we developed an XGBOOST model of execution mode estimation and performed feature importance analysis. As shown in Figure 4.7B, we found that the parameters controlling the kinetics of mBid binding to BcxL, and XIAP binding to C3 yield the most information about execution modes in EARMV2.0. Taken together, these results suggest that the analysis of signaling dynamics from uncertain parameters help us identify dominant reactions that control signal flow in a network during signal processing and how these networks are more sensitive to perturbations of those reactions.



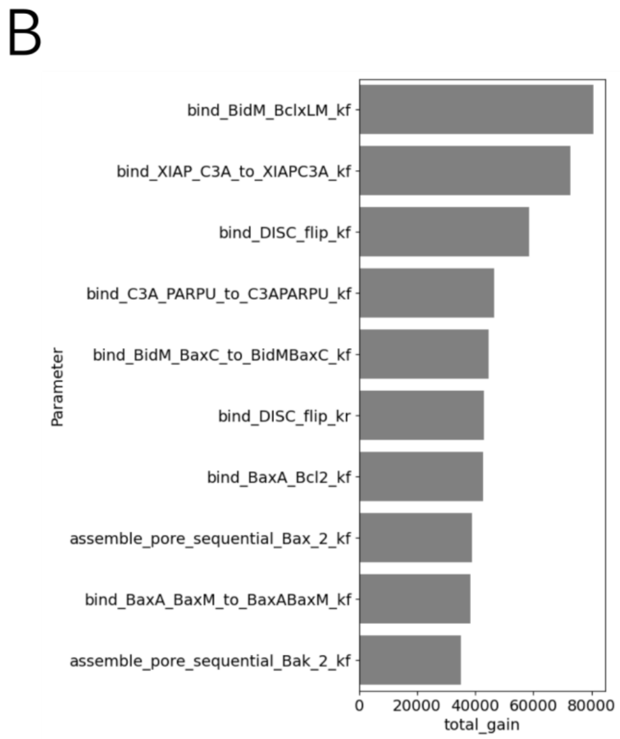


Figure 4.7 Upper panel: Time to death distributions in the execution modes 1 and 2 in the “wild type” condition, after a 50% Mcl1 knockdown, and after a 50% knockdown of Bcl2. The boxplot inside the distributions shows the median, first quartile and third quartile of the datasets. Execution modes 1 and 2 show substantial differences in their response to the knockdowns. Lower panel: Time to death distributions in the execution mode 1 after adding a drug that binds XIAP at t=8000 s, and at t=4000, and a drug that binds XIAP and Mcl1 at 4000 s. B) List of the 10 parameters in EARMv2.0 that contribute the most to model prediction.

4.5 Discussion

It has been long recognized that model parameter optimization to experimental data is key to investigate the dynamical properties that control cell behavior (Read et al., 2018). Unfortunately, parameter optimization usually yields large parameter uncertainties due to a general lack of quantitative data as well as model identifiability (Ashyraliyev et al., 2009). Even a complete set of time course data is insufficient to constraint most rate parameters (Gutenkunst et al., 2007a). In this work. we wanted to examine the effects of parameter uncertainty on signal execution through a biochemical network. Despite the many parameter vectors which reproduce the experimental protein dynamics, we found that the signal flow in a network was constrained to only a few modes of execution. Our analysis further shows that within a Bayesian calibration scheme, it is possible to assign probabilities to each execution mode, thus greatly improving our understanding of signal dynamics. Therefore, the probabilistic approach introduced in this work could open a novel perspective to understand network-driven processes from a statistical perspective.

In this work we also showed that large models with high parameter uncertainty such as EARMv2.0 can be used to make model-based predictions, but those predictions should be considered within the probabilistic context provided by execution modes obtained from the calibrated parameters. Our analysis shows that parameter uncertainty as a result of model calibration can be mapped to signal execution modes that respond differently to perturbations, thus demonstrating that using a single best fit parameter vector is insufficient to understand signal dynamics in complex models. Further, our analysis allowed us to identify biochemical species, model parameters and times to maximize a given perturbation. This information about signal flow could be used to study drug-induced network rewiring processes (e.g. (Lee et al., 2012)), provide mechanistic explanations to drug responses, and predict sequential combinations of drugs that could better modulate a response signal in biochemical networks.

Finally, although our approach provided novel insights about signal execution in an important biological network, it has certain limitations. Our analysis assumes that reactions with high flux are the most important for signal processing in a network. However, this may not always be the case for other networks or for networks with temporal changes in model topologies (Klinke, 2010). Although our approach is computationally expensive, particularly as models increase in size, requiring hundreds of thousands of parameter samples to reach a convergence criterion, we believe this is a relatively small price to pay in contrast to the number of experiments that would be necessary to attain the same level of mechanistic knowledge about a network-driven process.

4.5.1 Availability

All the code to reproduce the figures that contain model calibration, modes of signal execution, visualizations, and hypothesis exploration, is open source and can be found as Jupyter Notebooks in this GitHub repository: https://github.com/LoLab-VU/pydyno. These shareable and reusable notebooks contain all the source code and markup text that explains the rationale for each step in the analysis.

Chapter 5

Interactive Multiresolution Visualization of Cellular Network Processes

Ortega OO, Lopez CF. Interactive Multiresolution Visualization of Cellular Network Processes. iScience. 2020 Jan 24;23(1):100748

5.1 Summary

Cellular signaling pathways are controlled by networks of biomolecular interactions that process signals from environmental cues (Blinov et al., 2006; Lemmon & Schlessinger, 2010; Sachs et al., 2005). These molecular networks give rise to nonlinear dynamic processes that are difficult to explain and predict using reductionist methods (A. C. Ahn et al., 2006). Mathematical models of cellular signaling pathways have become commonplace to gain insights and describe the molecular mechanisms that control cellular processes (Albeck et al., 2008; Gaddy et al., 2017; K. a Janes et al., 2005; N. A. Perry et al., 2019). In general, these models continue to grow in size and complexity, which makes the exploration of network structure and dynamics increasingly challenging. Visualization tools present one effective way to explore network processes and acquire conceptual insights about signal-execution mechanisms. In addition, visualization tools can facilitate the detection of execution patterns and aid in hypothesis generation for experimental validation. However, most tools focus on static single-resolution network representations of models and generally lack support to visualize model dynamics. Therefore, there is an unmet need for tools that facilitate multi-resolution visualizations of model networks and simulated dynamics.

5.2 Introduction

Numerous tools have been developed to visualize network representations of models that capture relationships between model components. Some examples include molecular species networks (Bergmann et al., 2017), hierarchical species networks

(Paduano & Forbes, 2015), species-reactions networks (Schaff et al., 2016), contact maps (Boutillier et al., 2018; Cheng et al., 2014; L. A. Harris et al., 2016), model-defined rules (Boutillier et al., 2018; Cheng et al., 2014), and rule-based networks (Danos et al., 2012; Smith et al., 2012), among many others (Dang et al., 2015; Kolpakov et al., 2019; Tiger et al., 2012). Although these tools have been highly useful within their domains, they exhibit limitations when it comes to visualizing the structures of increasingly complex networks with an ever-larger number of nodes and edges labels. Moreover, standalone visualization tools can be difficult to incorporate into model-building and analysis workflows, further compounding the lack of reproducibility in analysis pipelines.

Identifying reactions that drive cellular processes is central to dynamic network analysis, yet it is highly challenging without visualization tools to facilitate an intuitive understanding of the signal execution mechanisms. A handful of tools to visualize dynamic network processes have been published, notably COPASI (Bergmann et al., 2017) and the Kappa Dynamic Influence Network (KDIN) (Forbes et al., 2017). COPASI uses a network in which nodes represent biochemical species and edges represent biochemical interactions. Species concentrations obtained from a simulation are encoded in the size of the box around the network nodes. Kappa employs a network in which nodes are the model rules and the edges indicate that the rules have common reactant or product species. KDIN quantitatively represents the temporal influence that each biochemical rule exerts on other rules. Although both tools yield useful information about dynamic network processes, information about the reactions that drive the dynamic consumption and production of different biomolecules, essential to understanding signal execution mechanisms, is not easily obtained. In addition, these tools have been developed for software-specific environments, thus limiting their use in general modeling and analysis workflows.

In this work we tackle three main visualization challenges that, we believe, will accelerate the conceptual understanding of biological network processes: (1) develop legible and comprehensible visualizations of increasingly large networks; (2) generate intuitive dynamic network visualizations of model simulations; and (3) facilitate the integration of visualizations into model building and analysis pipelines. To tackle these challenges, we developed Python Visualization of Processes and Reactions (PyViPR), a Python-based framework that provides multiple static and dynamic representations of biological processes. Importantly, PyViPR unifies tools typically used in isolation, enables access to community-detection algorithms, and encodes model simulations into node and edge attributes, thus enabling the study of network dynamics at multiple resolutions. PyViPR embeds all its visualization and analysis capabilities within Jupyter Notebooks (Kluyver et al., 2016) to facilitate reproducibility and dissemination of model analysis pipelines. PyViPR currently supports the rendering of rule-based models declared in the PySB framework (Lopez et al., 2013), BioNetGen (BNG) (L. A. Harris et al., 2016), and Kappa language (Boutillier et al., 2018), as well as models encoded in the SBML format (Hucka et al., 2003), thus providing a general tool to visualize models of biochemical network processes. In what follows we describe PyViPR's design and implementation, followed by a demonstration of key PyViPR capabilities in the exploration of cellular signal processing.

5.3 Materials and methods

5.3.1 Overview

PyViPR embeds static and dynamic network visualizations of different biochemical model components into a Jupyter Notebook (Kluyver et al., 2016). To generate these visualizations, PyViPR requires as input data a model or simulation result encoded in one of the accepted formats (See input data section below). Once a model or simulation result has been passed to one of the PyViPR functions, in the back-end PyViPR uses the Python package NetworkX (Hagberg et al., 2008) to generate node edge graphs that store information from model components (e.g., molecular species, reactions, rules) and simulation results. Species nodes can be clustered based on community detection algorithms or model compartments. Then, the NetworkX graph is converted into a JSON file that is passed to the JavaScript front-end which employs Cytoscape.js (Franz et al., 2015) and some of its extensions to render the graphs, apply layout algorithms, expand and collapse compound nodes (Dogrusoz et al., 2018), and enable the dynamic visualization of model simulation results for the visualization of networks within Jupyter Notebooks.

5.3.2 Input data

PyViPR currently includes three interfaces that enable the support of multiple model and graph formats: (i) PySB interface, which uses the PySB package (Lopez et al., 2013) to handle models encoded in the BioNetGen, SBML, PySCeS, E-Cell, and PySB format, (ii) Tellurium interface, which uses the Tellurium package (Medley et al., 2018) to handle models encoded in SBML and Antimony, and (iii) Graph interface, which uses NetworkX and Cytoscape.js to handle graphs encoded in GraphML, SIF, SBGN XML, Cytoscape JSON, GEXF, GML and YAML.

5.3.3 Output data

All visualizations are rendered as networks within a Jupyter Notebook. These networks can be locally downloaded in the following formats: PNG, SIF, GraphML and JSON.

5.3.4 PyViPR main visualization functions

PyViPR enables the interactive visualization of different model components as well as simulated trajectories of molecular species and reaction rates. The main visualization functions include:

- sp_rxns_bidirectional_view(model): Shows a bipartite network where one set of nodes are species and the other set are bidirectional reactions. Edges connect reaction nodes with their respective reactants and products species.
- sp_view(model): Shows the unipartite network of interacting species.
- projected_bireactions_view(model): Shows the unipartite network of reactions projected from the bipartite species-reactions graph.
- sp_comm_louvain_view(model): Shows the unipartite network of interacting species grouped by the Louvain community detection algorithm.

- sp_dyn_view(model): Shows a species network. Edges size and color are updated according to reaction rate values, and nodes pie charts slices are updated according to the concentration of species.
- cluster_rxns_by_rules_view(model): Shows the unipartite graph of the interactions between the reactions in a model. Reaction nodes are grouped by the rules that generated them.
- highlight_nodes_view(species, reactions): Highlights the species and/or reactions passed as arguments.

A more detailed description of these and other visualization functions can be found on the PyViPR documentation website (https://pyvipr.readthedocs.io/)

5.3.5 Model calibration

For the calibration of EARM, nominal values for rate constants were set to their published values in EARM 2.0 (Lopez et al., 2013). All rate constants were allowed to change two orders of magnitude above and below their nominal values, indicating a lack of knowledge about the likely parameter values. Experimental time-courses of the initiator caspase reporter protein (IC-RP), mitochondrial intermembrane space reporter protein (IMS-RP), and effector caspase reporter protein (EC-RP) were used from previously published data (Spencer et al., 2009). In the model, tBid, cytosolic Smac, and cleaved PARP were fit to the data for IC-RP, IMS-RP, and EC-RP, respectively. Model calibration was then performed using the simplePSO package, which is a python implementation of the Particle Swarm Optimization algorithm (Kennedy & Eberhart, 1995). The fit of simulated trajectories to experimental data was measured using the sum of the squared differences:

$$\chi^2 = \sum_t \sum_i \frac{1}{2\sigma_i^2(t)} [x_{model}^i(t;\, \Theta) - x_{data}^i(t)]^2$$

Where Θ is the parameter vector used to run a simulation, $x_{model}^i(t;\, \Theta)$ correspond to the simulated trajectory of molecular species i with parameter vector Θ, x_{data}^i is the experimental data from species i, and σ_i^2 is the variance from the experimental data of species i. The index i runs over all species (observables) that have experimental data, and the index t runs over all time points at which the data was measured.

We first ran PSO 100 times to determine a reasonable cost function threshold to consider that a calibrated parameter is a good fit. We chose the parameter set with the lowest function, which corresponds to a value of 2.8, and visually inspected that the fit was good. Then, we ran PSO 10000 times, and only kept parameter sets that had a cost function of 2.8 or less.

5.3.6 Parameter selection for analysis

To obtain the two maximally different parameter sets from the 6572 calibrated parameter sets, we first standardize the kinetic parameter values as kinetic parameters with a variance that is order of magnitudes larger than others might dominate the distance function and lead to parameters that are mostly different at that specific kinetic parameter. To standardize the values of a kinetic parameter we remove the mean and scale to unit variance:

$$z = (x - u)/s$$

Where x is a value of a specific kinetic parameter, u and s are the mean and the standard deviation of the specific kinetic parameter, respectively.

After standardizing the kinetic parameter values, we use the Euclidean metric to calculate the pairwise distances between the 6572 calibrated parameter sets, and then choose the two parameter sets that yield the largest distance.

5.4 Results

5.4.1 PyViPR Overview

PyViPR is a Python package that operates within the Jupyter notebooks environment (Kluyver et al., 2016). In this manner, PyViPR takes full advantage of a Literate Programming paradigm (Knuth, 2001), which enables the definition of both code and documentation concurrently and allows users to develop shareable workflows for model definition, visualization, and analysis. PyViPR leverages the capabilities of PySB to generate model objects, import models from BNGL and SBML formats, and provide simulation-based results for dynamic visualization. In addition, PyViPR integrates Cytoscape.js (Franz et al., 2015), a well-established JavaScript library for graph visualization, into the Python environment to interactively render static and dynamic visualization of model networks. Therefore, PyViPR merges software packages that would traditionally be used in isolation onto a common modeling environment. Further, PyViPR benefits from community-driven software development, and improvements made to any of its software dependencies are automatically accrued by the framework. PyViPR encourages community-driven collaboration through its open-source philosophy built around GitHub: https://github.com/LoLab-VU/PyViPR.

A typical PyViPR workflow comprises the following steps. First, a supported model file is passed to one of the PyViPR visualization functions. PyViPR then uses NetworkX (Hagberg et al., 2008) to convert the model components into graph nodes and edges. The user could then simplify the graph through community detection (e.g., with the Louvain algorithm (Blondel et al., 2008)) on the NetworkX graph object. The software will then create a compound node and place all the nodes from a community within it. For dynamic visualization, PyViPR maps the simulated species concentrations and reaction data to node and edge properties. The resulting NetworkX graph is transferred to cytoscape.js via a JSON dictionary and rendered real-time in a Jupyter notebook for visualization. We note that the user can interact with all graph objects in a Jupyter notebook rendering to, e.g. change the layout, groupings, or placement of a given graph.

PyViPR supports visualization of the two main approaches used to build chemical kinetics models of cellular regulatory networks. In the first approach, reaction networks are generated by enumerating all the molecular species and reactions that can occur in a cellular process. This reaction network can then be translated into a set of Ordinary Differential Equations (ODEs) or stochastic equations (Aldridge et al., 2006). In the second approach, rule-based modeling formalisms (Boutillier et al., 2018; Faeder et al., 2009; Lopez et al., 2013) are used to circumvent the need to enumerate all the species and reactions by hand. In these formalisms, species are defined as structured objects that can have binding and state sites, and reaction rules define interactions between specific domains or binding sites on a given species. Then, rule-based modeling tools automatically generate a reaction network by identifying all possible species that have the conditions required to undergo the interaction defined in a rule. PyViPR supports visualization of both model encodings through a Tellurium (Choi et al., 2018) interface for reaction network models and a PySB (Lopez et al., 2013) interface for rule-based models.

In addition to biochemical network visualization, PyViPR supports the following graph formats widely used in the systems-biology community: GraphML, SIF, SBGN XML, Cytoscape JSON, GEXF, GML, and YAML. Additionally, rendered graphs in a Jupyter Notebook can be downloaded in the following formats: PNG, SIF, GraphML, and JSON.

5.4.2 Design Choices for PyViPR

Numerous approaches have been developed to visualize temporal networks. Beck et al. (Beck et al., 2017) surveyed a range of existing tools and derived a taxonomy based on temporal representation, either as an animation or as a static timeline. From this perspective, PyViPR would be classified as a hybrid visualization that uses the node-edge paradigm to visualize networks, an animation for visual representation of time, and superimposition of pie charts embedded in nodes as well as edges width and color, to represent the temporal changes in species concentration and reaction flux, respectively. PyViPR was designed with the following visualization goals:

G.1) Highlight functionally related species by grouping them in compound nodes.
G.2) Understand how a signal is executed in a biochemical network and how it depends on parameter values.
G.3) Provide easy-to-use interactive visualizations for investigating the topology and simulation results of biochemical models.

Murray et al. (Murray et al., 2017a) identify a task taxonomy for biological pathways analysis across three categories: attribute, relation, and modification tasks. PyViPR specifically supports attribute tasks, to obtain information about a species node, and relationship tasks to identify types of relationships between nodes (e.g. protein binding, protein translocation), the direction of nodes interactions, and grouping relationships (e.g. model compartments, communities). With respect to the temporal features tasks described in the task taxonomy of network evolution analysis by Ahn et al. (J. W. Ahn et al., 2014), PyViPR focuses on the temporal features of aggregated events. More specifically, PyViPR aims to make it easy to observe at any point in time, the reaction rates that have a higher flux than other reactions.

To satisfy the design criteria introduced above, we made the following design choices:

DC.1) Employ node-link diagrams for all static and dynamic visualizations to show interactions between model components. We decided to use node-link diagrams because they are commonly used by biology experimentalists (Cerami et al., 2011; Demir et al., 2010) and computational modelers (Murray et al., 2017b) and that would facilitate the interpretation and communication of results.

DC.2) Dynamically map simulated species concentrations and reaction rates values onto pie charts embedded within nodes and edge color and width, respectively. Our main goal is to clearly show the reactions that carry most flux and drive the behavior of the system over time. Therefore, we followed the design principles for the representation of flow quantity and direction discussed by Bernhard et al. (Jenny et al., 2018) and used color brightness to represent reaction flow quantity on edges.

DC.3) Include search mechanisms, multiple layout options, zoom and grouping functionality to organize model components, and focus on important details, thus enabling interactive exploration of complex biological networks.

5.4.3 Network Creation from Multiple Model Components

PyViPR supports visualization of multiple model components, including molecular species, reactions, rules, compartments, macro functions (Lopez et al., 2013), and modules comprising independent model elements (Lopez et al., 2013). These components are depicted by either simple nodes, which are fundamental units in a graph (Figure 5.1A), or compound nodes, which can contain children nodes and are used to group simple nodes with shared attributes or through user-defined groupings. Interactions between these different model components correspond to unidirectional or bidirectional reactions and are represented by arrows (Figure 5.1B).

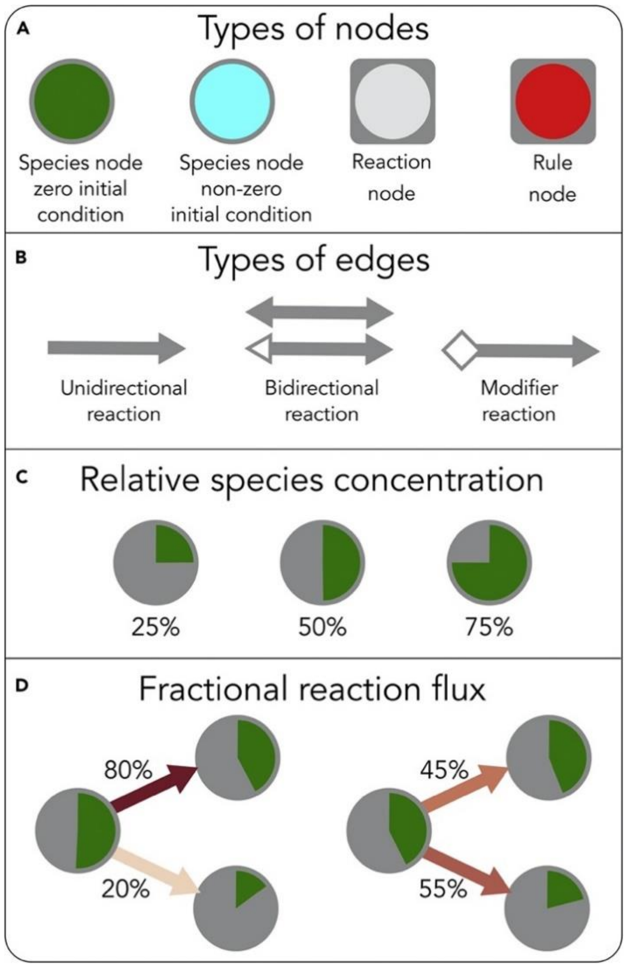


Figure 5.1 PyViPR Visual Encodings. (A) Node types used for visualizations as labeled.
(B) Edge types used for interactions: unidirectional interactions (left) are depicted with a unidirectional arrow and represent irreversible biochemical reactions. Bidirectional interactions (middle) are depicted with bidirectional arrows and represent reversible reactions. Arrows fill state indicate directionality from reactant (hollow) to product (solid) species. Solid bidirectional arrows represent bidirectional interactions lacking directionality information. Modifier reaction (right) are

depicted with an arrow tail shaped with a hollow diamond and a solid arrow head and represent reactions where the species is both a reactant and a product of the reaction. (C) Pie charts embedded within nodes indicate the concentration of a species relative to its maximum value in the simulation. (D) Color shade of arrows indicate the fractional reaction flux for interactions.

To create a bipartite network, PyViPR first obtains the list of species and rules/reactions from a model and adds them as nodes to the network. Then, PyViPR uses edges to connect species nodes with their respective rule/reaction node. To reduce the network resolution a bipartite graph can be projected onto a unipartite graph that contains only the species or rules/reactions nodes. This unipartite species graph can then be organized by grouping the species nodes using the biological compartments on which they are located. Similarly, a unipartite rules graph can be grouped by the macro functions used to create them or the model modules where they are defined. This allows users to interactively explore and revise the model network topology at different resolutions.

A key feature in PyViPR is the use of community detection algorithms to automatically cluster nodes and thereby simplify network complexity. For example, the Louvain method detects communities by optimizing the graph modularity. In this method, optimization is achieved by first iterating over all nodes and assigning each node to a community that results in the greatest local modularity increase, then each small community is grouped into one node and the first step is repeated until no modularity increase can occur (Blondel et al., 2008). As a result, the Louvain algorithm finds groups of highly connected nodes that could have similar biological functions or represent molecular-complex formation processes (Fortunato, 2010) (design goal 1). Other community detection algorithms based on label propagation (Cordasco & Gargano, 2010; Raghavan et al., 2007), fluid communities (Parés et al., 2018), and centrality (Girvan & Newman, 2002) methods are also available in PyViPR. Alternatively, users can also manually define clusters of nodes interactively for a "human in the loop" type optimization (Däschinger et al., 2017; Holzinger, 2016). Taking advantage of the PySB interface to BioNetGen, we also incorporated (1) compact rule visualization, (2) atom-rule graph, and (3) tunable compression pipeline as implemented by Sekar et al. (Sekar et al., 2017) into the PyViPR workflow to enable a more thorough and complete visualization of large rule-based models.

5.4.4 Dynamic Visualization in PyViPR

PyViPR supports dynamic visualization of deterministic and stochastic model simulations. This visualization mode uses a unipartite network (Design Choice DC.1) in which nodes represent model species and edges represent reactions between the species. Species concentrations and reaction rates are encoded into the properties of nodes and edges, respectively (Design Choice DC.2).

To represent temporal concentration changes during a simulation, we embedded pie charts within the graph species nodes. Pie chart slices within each node depict the species concentration relative to the maximum amount attained throughout the simulation. Pie chart slices are updated at each time point during animation (Figure 5.1C). Absolute species concentrations at a given time point are also accessible as tooltips through a click-hold gesture on a species node.

PyViPR aims to highlight reactions with high rates of consumption or production, as these can drive complex network processes (Design Goal 2). To attain this goal, simulated reaction rates are encoded on both the color shade and the thickness of arrows that connect interacting species. For each species PyViPR obtains its related reactions and then calculates the fractional flux of each reaction using a normalization function:

$$f_{i,c}(t) = \frac{r_{i,c}(t)}{\sum_{j=1}^{n} r_{j,c}}$$

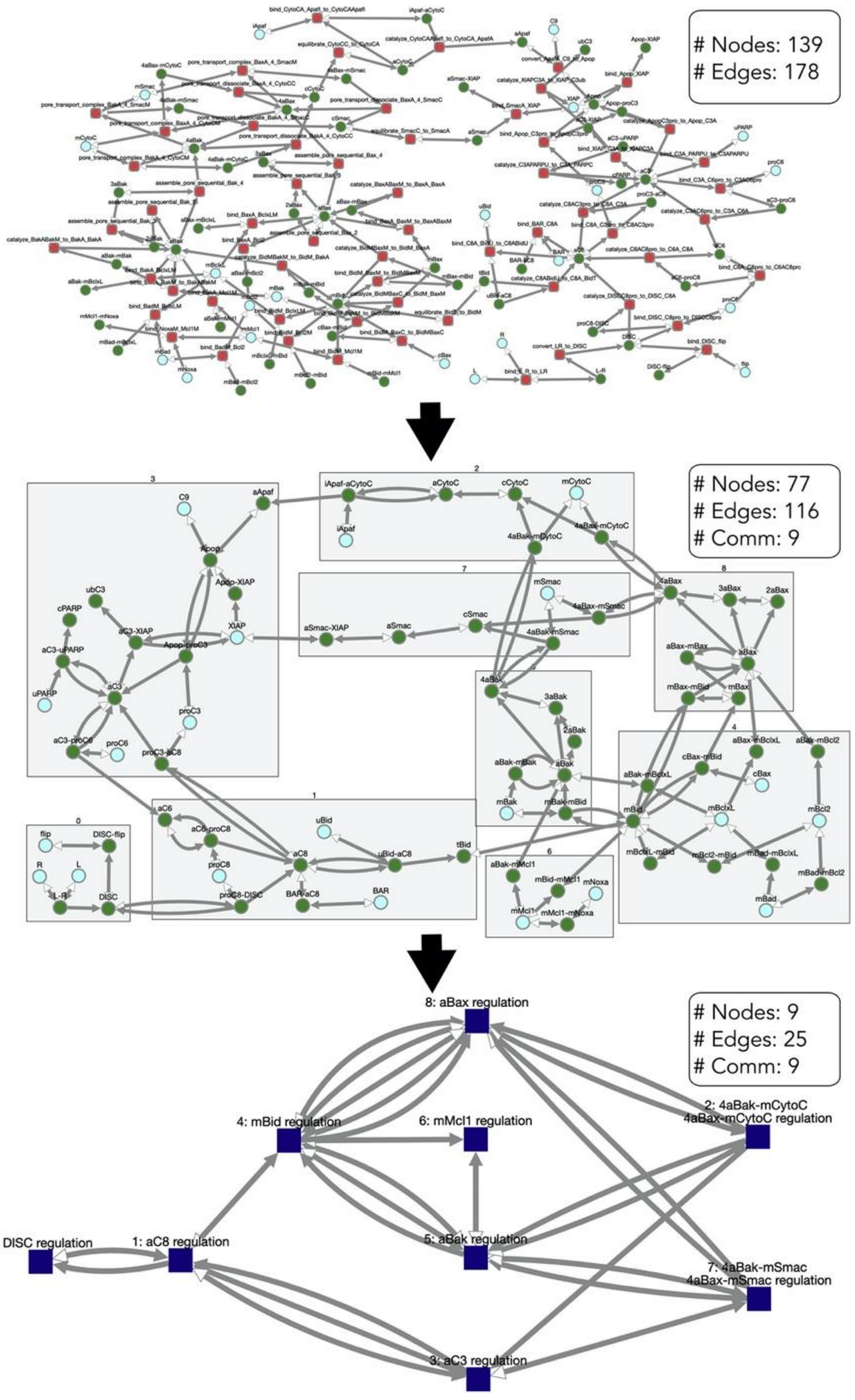
where $r_{i,c}$ is the reaction rate value at a specific time point, n is the number of reactions, and the sub-index c indicates the type of reaction (consumption or production). Fractional fluxes are then linearly mapped to a color shade ranging from low (light shade) to high (dark shade) flux representations (Figure 5.1D). In addition, reaction rate values, relative to their maximum value throughout the simulation, are represented by edge thickness. Absolute reaction rate values for each interaction and at any given time point are also accessible as tooltips using the click-hold gesture.

5.4.5 Exploration of a Biological Process with PyViPR: Apoptosis Execution

To illustrate the visualization capabilities of PyViPR, we use the Extrinsic Apoptosis Reaction Model (EARM v2.0) (Lopez et al., 2013) to perform an exploratory analysis of the receptor-mediated apoptosis signaling network. Briefly, EARM v2.0 describes the biochemical interactions from an initial death ligand cue to a cleaved PARP response. Initiator caspases trigger interactions among the Bcl-2 family of proteins that lead to mitochondrial outer membrane permeabilization (MOMP). MOMP, in turn, propagates the signal to effector caspase activation and PARP cleavage. EARM is a sizable model that comprises 74 molecular species, 127 parameters, 62 rules, and 100 reactions. We explored the EARM network using the following steps: (1) visualization of the apoptosis species-rules bipartite network; (2) application of the Louvain community detection algorithm to functionally cluster species nodes; (3) study of the simulation dynamics at a coarse-grained community level; and (4) identification of molecular targets that modulate model behavior.

5.4.6 Multiresolution Visualization and Exploration of EARM

We wanted to study the architecture of the network defined in EARM to find insights about molecular organization and function in apoptosis execution. We first visualized a species-rules bipartite network (Figure 5.2, upper panel). However, this network is difficult to explore, as no discernible structures are readily apparent. We then projected the species-rules bipartite graph onto a species unipartite graph and clustered highly connected nodes using the Louvain algorithm (Figure 5.2, middle panel). These communities can also be further collapsed to obtain the EARM communities graph, a coarse-grained representation of the apoptosis pathway (Figure 5.2, lower panel).



# Nodes: 139
# Edges: 178
# Nodes: 77
# Edges: 116
# Comm: 9
# Nodes: 9
# Edges: 25
# Comm: 9
8: aBax regulation
4: mBid regulation
6: mMcl1 regulation
2: 4aBak-mCytoC 4aBax-mCytoC regulation
0: DISC regulation
1: aC8 regulation
5: aBak regulation
7: 4aBak-mSmac 4aBax-mSmac regulation
3: aC3 regulation

Figure 5.2 Multiresolution Visualization of a Reaction Network. Upper panel: EARM species rules bipartite graph. Green nodes represent molecular species with initial condition set to zero, cyan nodes are species with nonzero initial conditions, and red nodes represent rules defined in the model. Middle panel: EARM species graph obtained from projecting the bipartite graph into a unipartite graph. Densely connected nodes have been automatically grouped into communities detected with the Louvain algorithm. Lower panel: EARM communities graph obtained by collapsing each communities into a single node. Community node names are assigned by the species with the highest number of interactions within its community. All edges correspond to interactions between species nodes as specified in the EARM model.

The Louvain community detection algorithm identified nine communities, numbered 0–8, which is summarized in Table 5.1. Briefly, these communities capture biologically relevant and functional processes throughout the apoptosis pathway. Community 1, describing Caspase 8 activation and Bid truncation, is linked with Communities 3 and 4, the starting points for type I and type II cellular apoptosis, respectively (Özören & El-Deiry, 2002). Interestingly, Mcl-1, a potent apoptosis inhibitor, was placed in a separate community from all the other Bcl-2 inhibitors, highlighting its unique inhibitory interactions that have been well documented (Yang-Yen, 2006). Community 4 is also connected to Communities 5 and 8 that correspond to Bak and Bax activation, polymerization, and pore formation, respectively. These communities capture mitochondrial regulation events that lead to eventual MOMP formation in type II apoptosis execution (Kale et al., 2017; YIN, 2000). These MOMP-related communities are connected to Communities 2 and 7, which correspond to MOMP-driven release of cytochrome c and Smac from the mitochondria. Finally, these communities connect to Community 3, which corresponds to the activation of executioner Caspase 3 (C3) and subsequent PARP cleavage, which signals that the cell has executed apoptosis. As shown, C3 can be directly activated by Caspase 8 (C8) (type I) or by the apoptosome formed after cytochrome c is released via MOMP (type II).

Community Number	Apoptosis Subprocess	References
0	Ligand-receptor interactions that lead to the DISC formation and regulation by Flip	(Pennarun et al., 2010)
1	Initiator Caspase 8 activation by DISC and Caspase 6 and subsequent truncation of Bid by activated Caspase 8	(Kantari & Walczak, 2011)
2	Release of cytochrome C through mitochondrial Bax and Bak pores	(Garrido et al., 2006)
3	Activation and regulation of effector Caspase 3, formation of apoptosomes, and cleavage of PARP	(Zou et al., 2003)
4	Bcl-2 family of interactions responsible for translocation of Bax to the mitochondria by mitochondrial Bid (mBid) and inhibition of Bax, Bak, and mBid by BclxL and Bcl2	(Kale et al., 2017; YIN, 2000)
5	Formation of mitochondrial Bax pores	(Annis et al., 2005; Westphal et al., 2014)
6	Mcl1 inhibition of pro-apoptotic proteins	(Yang-Yen, 2006)
7	Release of Smac through mitochondrial Bax and Bak pores and its subsequent inhibition by XIAP	(Deng et al., 2002)
8	Formation of mitochondrial Bak pores	(Dewson et al., 2009; Westphal et al., 2014)

Table 5.1 Summary of the Biological Functions Enclosed in Each Community.

The Louvain algorithm also led to some interesting observations regarding molecular interactions. For example, Caspase 3, the effector caspase, is the species with the highest within-community node degree, indicating that it is a highly regulated protein in apoptosis execution. Also, mBid has the highest number of interactions across communities, indicating that it plays a key regulatory role in apoptosis execution.

Taken together, we find that Louvain community detection could be used as an interactive "coarse-graining" methodology to automatically group biochemical interactions, simplify mechanism exploration, and identify important proteins within a biochemical network.

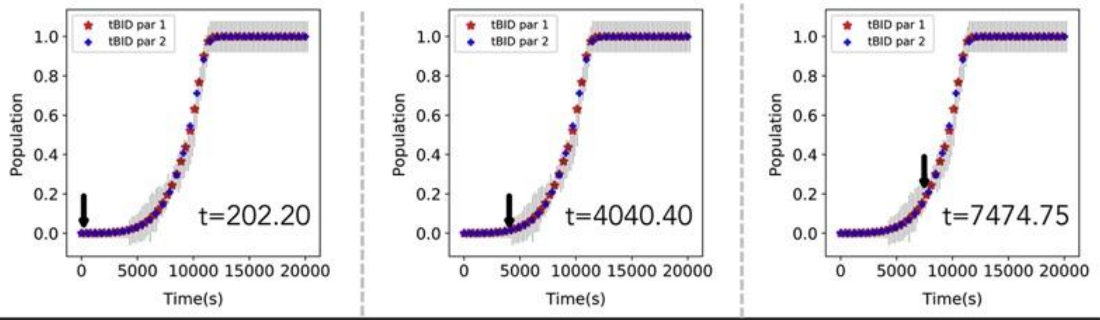
5.4.7 Parameter Sets Fit Experimental Data but Yield Different Network Dynamics

To demonstrate the advantages of dynamic visualization, we calibrated EARM to previously published time course experimental data (Spencer et al., 2009) using the Particle Swarm Optimization (PSO) algorithm (Kennedy & Eberhart, 1995). Ten thousand PSO runs were carried out, which resulted in 6,572 parameter sets with an error ≤ 2.8 (See Methods section for details). It is well established that multiple parameter sets can fit experimental data equally well, due to parameter unidentifiability and model sloppiness (Gutenkunst et al., 2007b). To explore the mechanistic implications of different parameter sets on EARM execution, we compared the dynamics generated by two different parameter sets, labeled Parameter Set 1 (PS1) and Parameter Set 2 (PS2) (Table S1), as described below.

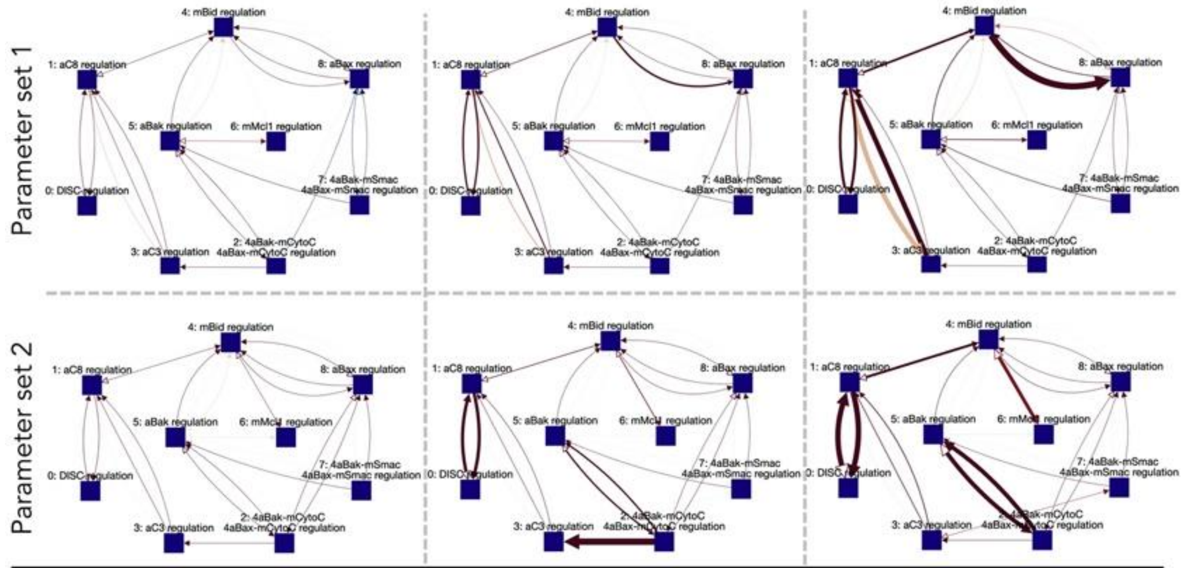
We hypothesized that these two parameter sets with different kinetic parameter values would yield distinct signal mechanisms. Thus, we first asked whether a trajectory plot of tBid dynamics could yield useful mechanistic information about apoptosis execution with different parameter sets. As shown in Figure 5.3A, both parameter sets generated tBid trajectories that were essentially indistinguishable, yielding no mechanistic information from the two distinct parameter sets. We then employed the EARM communities graph to compare the global dynamic signal execution for both parameter sets. Figure 5.3B shows three time points in signal execution for PS1 (upper panel) and PS2 (lower panel). As shown, there is little activity between communities in both parameter sets in the early time points

(Figure 5.3B left). However, for PS1 at t = 4040s we observe that Community 1, which regulates C8 activation, exhibits increased flux toward Community 3, which controls C3 activation. This indicates that C3 is being activated by C8. Despite the activation of the effector Caspase, apoptosis does not take place because the antiapoptotic XIAP inhibits active C3 activity. Community 4, which regulates mBid, also exhibits increased signal flux toward the Community 8 (active Bax regulation), indicating that the pores formed in the mitochondria are dominated by Bax oligomerization.

A EARM calibration to experimental data



B Global reaction flow dynamics



C Local reaction flow dynamics

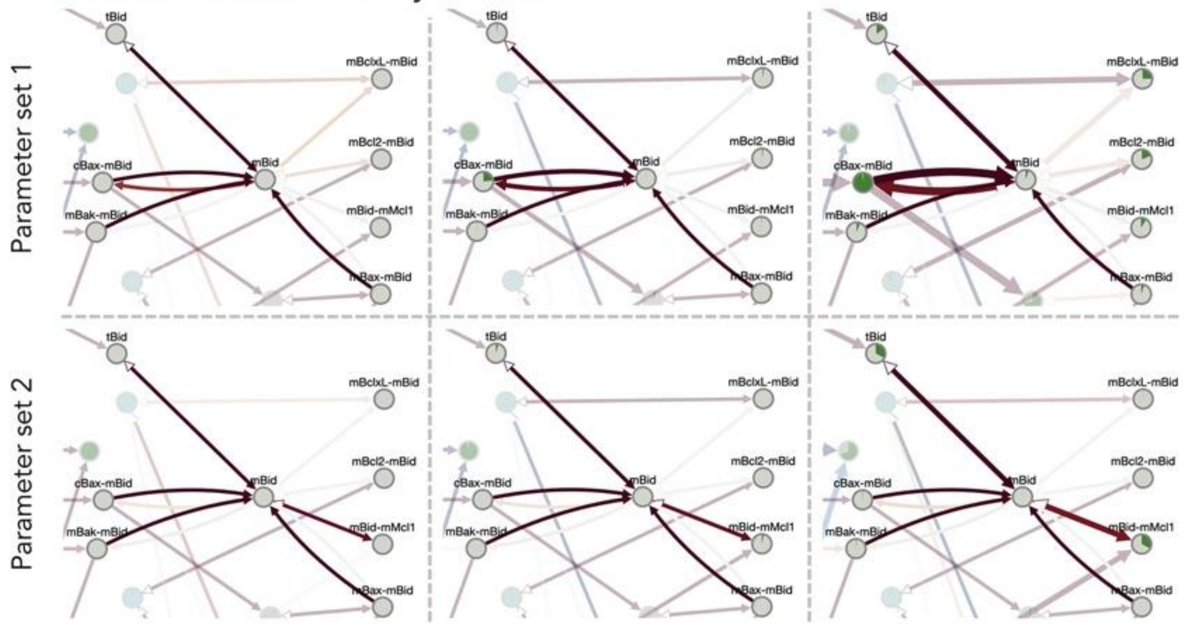


Figure 5.3 Dynamic Visualization of EARM at Different Resolutions. Panel (A) includes three plots of the simulated tBid and the experimental data used for calibration. Gray lines correspond to the experimental data with error bars indicating the standard deviation. Arrows indicate the concentration level at the corresponding time point. The time points shown here are the same ones used to obtain snapshots of the dynamic visualization in the following panels. Panel (B) shows, for PS1 and PS2, the global reaction flow dynamics between the communities detected with the Louvain algorithm. Panel (C) shows, for PS1 and PS2, the temporal changes in the strength of the interactions between mitochondrial Bid and the anti-apoptotic and pro-apoptotic proteins. Pie chart slices within the nodes show the concentration of a species relative to the maximum amount of the concentration attained across all time points of the simulation. Edges color shade and thickness represent the fractional flux and relative reaction value, respectively.

PS2 also exhibited increased signal flux between communities but with different interaction patterns compared with those seen in PS1. Specifically, it exhibited significant flux between Community 4 and Community 6 (mMcl1 regulation) at t = 7474s, suggesting that mBid was being inhibited by mMcl1. Also, there was significant signal flux from Community 2, which regulates cytochrome c release from the mitochondria, toward Community 3, indicating that pores were already formed in the mitochondria and cytochrome c was being released to aid with the formation of the apoptosome. Therefore, dynamic visualization of signal flow across communities confirms our hypothesis about signal execution and demonstrates the usefulness of PyViPR to explore the complex dynamics that occur in biochemical processes.

To further explore the effects of kinetic parameters in model behavior, we focused on local signal flow through mBid and its interactions, as they are tightly linked to MOMP and cellular time-to-death (Spencer et al., 2009). As shown in Figure 5.3C, for PS1 we observed that most of mBid was used to transport cytosolic Bax to the mitochondrial outer membrane (MOM), whereas no activation of Bak occurred, suggesting that pore activity in the MOM was primarily due to Bax (see Video S1). We, therefore, hypothesized that the model with PS1 depends on Bax for apoptosis execution. In contrast, for PS2, we observed that mBid activity was primarily inhibited by the anti-apoptotic Mcl1 (see Video S2). We thus hypothesized that under PS2 an MCL-1 knockdown would free mBid to activate Bax and Bak and more rapidly commit cells to apoptosis. We tested both hypotheses derived from our visualization-based analysis using in silico experiments. First, we knocked out Bax and simulated EARM with PS1 (Figure 5.4). We found that knocking out Bax protected cells from apoptosis induction with TRAIL, confirming that Bax plays an important role in apoptosis regulation. We then knocked out Mcl1 and simulated EARM with PS2. We found that the time-to-death was reduced by 22.6%, corroborating that Mcl1 inhibition delayed apoptosis. As a control, we knocked out Mcl1 for PS1 and Bax for PS2 and found that the dynamics of cPARP were not considerably affected.

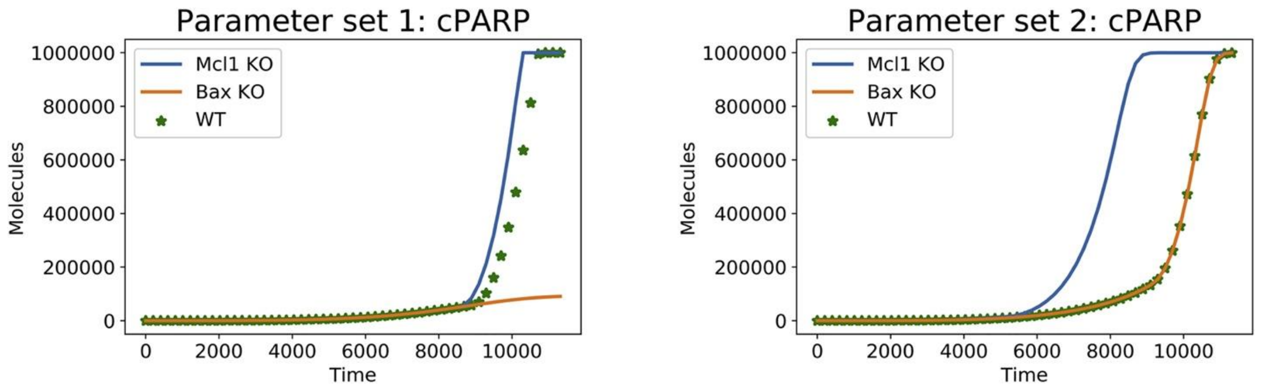


Figure 5.4 In Silico Knock outs of Bcl-2 Proteins Modulates PARP Cleavage. EARM was run using Parameter Set 1 (left) and Parameter Set 2 (right) and with knockout (KO) of either Bax or Mcl-1 as labeled. Bax KO has a significant effect on the dynamics of PARP cleavage (yellow line) for Parameter Set 1 but almost no noticeable effect for Parameter Set 2 (right). In contrast, Mcl-1 KO has almost no effect for Parameter Set 1 (left) but a significant effect in Parameter Set 2 (right).

Taken together, our results demonstrate that despite multiple parameters fitting the data equally well, apoptosis is executed differently for each parameter set. Our observations align with experimental results that show cellular dependence on Bcl-2 regulators for apoptosis execution (Deng et al., 2002; P. Zhou et al., 1997). Importantly, visualization of the dynamic process enabled us to identify key reactions under different parameter sets and generate testable hypotheses to better understand the execution mechanism.

5.5 Discussion

In this paper, we presented PyViPR, a tool to visualize the structure and dynamics of biochemical network models. PyViPR enables a straightforward workflow of model creation, analysis, visualization, and hypothesis generation. Additionally, PyViPR integrates community detection algorithms to organize the nodes of biochemical networks and ease the exploration of complex networks. Lastly, PyViPR provides an interface for intuitive dynamic visualization that facilitates the observation of signal flow across biochemical models.

Multiple tools exist for static visualization of biological networks. Some of the tools used to visualize reaction-based models include Dynetica (Eidum et al., 2014), COPASI (Bergmann et al., 2017), CySBML (König et al., 2012), and Omix (Droste et al., 2011). Although these tools provide useful visualizations of biochemical models, they are implemented as Graphical User Interfaces, which can hinder the creations of pipelines for model creation, visualization, and analysis. Also, these tools can become difficult to use as network complexity increases. PyViPR aims to address these issues by enabling access to community detection algorithms for network simplification and facilitating the model definition, visualization, and analysis pipelines in a single Jupyter Notebook environment.

Various tools for visualization of rule-based models have also been published. These include Simmune (Cheng et al., 2014), BioNetGen (Sekar et al., 2017; Smith et al., 2012), rxncon (Tiger et al., 2012), Virtual Cell (Vasilescu et al., 2018), and Kappa (Boutillier et al., 2018). All of these tools take advantage of the structured definition of molecules and rules to generate intelligible visualizations of large models. PyViPR does not use these structured definitions and instead uses the rule-based modeling framework to obtain the set of reactions from a given model. This set of reactions is often larger than the number of rules, thus limiting the size of models that can be intelligibly visualized with PyViPR. To address this potential shortcoming, we leveraged the flexibility of a Python-based environment and provided an interface to BioNetGen's atom-rules graph algorithm.

Visualization tools to explore the dynamics of temporal network processes can be classified into three groups based on the components animated: (1) Species nodes animation, where the simulated species concentration is mapped onto the size/color of nodes (e.g. COPASI (Bergmann et al., 2017), Narrator (Mandel et al., 2007), CytoModeler (Xia et al., 2011)); (2) species nodes and edge animation, where the simulated species concentration is mapped onto the size/color of nodes, whereas reaction rate values are encoded into the edge thickness (e.g. DBSolve (Gizzatkulov et al., 2010)); and (3) rules nodes and edges animation (e.g. DIN-Viz (Forbes et al., 2017)), where the number of hits of a rule is mapped into the node size, and the influence of one rule on another is encoded into the edge width. Similar to the first two groups of dynamic visualization approaches, PyViPR maps the species concentrations into nodes. The main difference, however, is that PyViPR encodes the reaction rates into edges width and colors in a more insightful way as it highlights the edges that carry most of the signal flow. Additionally, PyViPR is better suited for dynamic visualization of large networks, as it can use community detection algorithms to cluster nodes and then animate the coarse-grained network with the simulation results. Lastly, it is difficult to compare PyViPR with the third group of visualization tools, as PyViPR uses a species graph, whereas the latter uses a rules graph to encode the simulation results. However, one advantage that PyViPR has is that the visualization can be easily communicated to non-modeling scientists, as it only requires knowledge about the biological network being studied.

We believe that PyViPR could be incorporated into existing modeling and simulation workflows provided by Python-based tools such as Tellurium notebooks (Medley et al., 2018) and PySCeS (Olivier et al., 2005). In the future, we plan to incorporate community-detection algorithms that consider the weight of the edges for the clustering of nodes. Additionally, we plan to improve the synchronization from the JavaScript frontend to the Python backend to enable users to interactively modify model parameters and components.

All the model exploratory analyses, which include model calibration, visualization, hypothesis exploration, and testing, were performed in Jupyter Notebooks. These shareable and reusable notebooks contain all the source code and markup text that explains the rationale for each step in the analysis. We believe that access to these resources will promote reproducibility and transparency by enabling other researchers to rerun or expand the presented model analysis. We invite the community to contribute to open-source tools such as PyViPR to improve model analysis and visualization (see Supplement Information Section and https://mybinder.org/v2/gh/LoLab-VU/PyViPR/master).

5.5.1 Limitations of the Study

Although PyViPR can visualize a broad range of systems biology models, it is not a panacea for model visualization. Specifically, PyViPR has limitations to generate intelligible networks of rule-based models with rules that generate a few hundreds of molecular reactions. This limitation emerges because PyViPR visualizations are created from the molecular reactions, which are typically more numerous than model rules, instead of the monomers and rules encoded in a model. In this case, specialized visualization tools such as atom-rules (Sekar et al., 2017), rxncon (Tiger et al., 2012), and Kappa (Boutillier et al., 2018) could be better suited to obtain intelligible visualizations of rule-based models.

Chapter 6

Discussion and future directions

Mathematical models are becoming increasingly used to describe biochemical processes and understand the systems behavior that arises from protein-protein interactions involved in a signaling pathway. These models have provided insights into the multiple mechanisms that cells use to generate a response to perturbations. Some of these mechanisms include a snap-action switch controlling apoptosis (Albeck et al., 2008), spatial localization of proteins by scaffolds (Locasale et al., 2007), and a conveyor belt mechanism for signal amplification of JNK3 (N. A. Perry et al., 2019). As models have proven useful to understand complex dynamics, in recent years, the U.S. Food and Drug Administration (FDA) has advocated for the use of mechanistic models to advance the discovery, development and clinical use of therapeutic drugs (Madabushi et al., 2019; Sorger et al., 2011). These models, also known as Quantitative Systems Pharmacology (QSP) models, aim to describe the biochemical networks targeted by drugs, provide an understanding of the mechanisms of action of compounds under study, and investigate exposure-response relations for many therapeutics.

In this thesis, we develop tools and analyses to gain confidence in model results, contributing to the advancement of three specific areas: 1) Model reproducibility. 2) Quantification of parameter uncertainty and its effect on predictions and signal execution and 3) Model visualization. Improvements in these areas are essential for systems biology and QSP models to gain insights into the mechanisms that modulate cellular processes in different contexts.

6.1 Advances in model reproducibility

Regarding model reproducibility, multiple tools exist to address different parts of this issue. For example, the SBML format was created to share, communicate and store models (Hucka et al., 2003). Although SBML ensures a standard format for models, problems reproducing model analysis still exist as information about the protocols used to generate simulations is limited. Thus, the Simulation Experiment Description Markup Language (SED-ML) (Waltemath et al., 2011) was developed to address this lack of information about simulation experiments.

Although SBML and SED-ML have provided a common framework to describe models and simulations, a recent study found that 49% of the published models analyzed are not reproducible (Tiwari et al., 2021). Furthermore, they observed that the percentage of not reproducible models increased with model size, and more than 80% of models with more than 81 species are not reproducible. In this scenario of large models, rule-based modeling tools like PySB can play an important role. PySB facilitates the declaration of large models by using a syntax similar to biochemical reactions and automatically generates the ordinary differential equations that describe cellular processes. This framework enables easy revision and extendibility, which is essential for reproducibility purposes. However, if no information is shared about how different simulations and paper figures are generated, it can lead to the same reproducibility problems.

In our work, we employed a novel workflow that takes advantage of the vast ecosystem of tools available for the Python community and ensures reproducibility of all model results and analysis without the need of installing any software. In this workflow, we first use PySB to build, simulate and analyze a model within a Python Jupyter Notebook (Kluyver et al., 2016) and then store those notebooks in a free GitHub repository. Once the Jupyter Notebook with all the information is stored in GitHub, we used *binder* (Jupyter et al., 2018) to enable other researchers to run those notebooks in a browser-based executable environment where they can immediately reproduce the results previously obtained. We applied this workflow to the JNK3 activation reaction model, and the results can be reproduced here: https://github.com/LoLab-VU/JARM

Our approach for reproducibility can be used for models and analyses that require small-size files as GitHub and *binder* have storage and memory limits. In cases where large files and high computer processing power is necessary, we can envision a solution with

Docker containers (Boettiger, 2015) that include all the software, model, and analyses and can be run in a local computer system.

6.2 Advances in model visualization

To facilitate the visualization of model networks, simulation results and strengthen the reproducibility of modeling workflows, we developed the Python Visualization of Processes and Reactions (PyViPR) tool. PyViPR complements the already existing suite of tools that include NetworkViewer (Cheng et al., 2014), ReactionFlow (Dang et al., 2015), RuleBender (Smith et al., 2012), and DIN (Forbes et al., 2017) by providing two novel visualizations: 1) Clustered networks by community detection algorithms, and 2) Dynamic networks. These visualizations were designed to effectively display increasingly large networks and show the temporal patterns of species concentrations and reaction rates in an interactive manner. Additionally, PyViPR integrates other tools like atom-rules (Sekar et al., 2017) to visualize rule-based models and other graph formats like GraphML, SIF, and SBGN. We develop PyViPR as a Jupyter widget to be easily included in the modeling workflow and thus guarantee the reproducibility of model building, analysis, and visualization.

The dynamic visualizations generated by PyViPR enable the rapid generation of hypotheses by explicitly showing the protein-protein interactions with highest reaction rates, which then can be targeted to module signal execution. Additionally, PyViPR facilitates the communication of results with non-modeling scientists as models and simulations can be shown in the context of all protein-protein interactions.

In the future, we plan first to create an input format to import model networks and simulation results from other modeling tools into PyViPR for visualization. Second, we will add support to visualize stochastic simulations. Third, we plan to incorporate community-detection algorithms that consider the weight of the edges for the clustering of nodes. Lastly, we plan to improve the synchronization from the JavaScript frontend to the Python backend to enable users to modify model parameters and components interactively

6.3 Advances in the quantification of model uncertainty

Various approaches exist to calibrate models to experimental data (Mitra & Hlavacek, 2019b). Bayesian calibration tools have gained relevance in the fields of genetics, genomics, bioinformatics, and computational systems biology as they provide essential information about parameter distributions and their correlations (Wilkinson, 2007). In Chapter 2 of this work, we employed PySB to build a model of the JNK3 activation cascade and then used PyDREAM, a Bayesian calibration tool, to fit the model to experimental data. This calibration yielded parameter probability distributions constrained by the experimental data available and enabled us to compare kinetic parameters taking into account the uncertainty of their values. Additionally, given the statistical nature of PyDREAM we could identify the most likely parameter vectors that reproduce the experimental data. Altogether, the experimental data, our model with calibrated parameter distributions, and the analysis of reaction rates provided insights into the mechanism that amplifies the activation of JNK3.

For the development and analysis of the JNK3 activation reaction model (JARM), we applied the experimentation/modeling cycle that starts by gathering experimental data, building a mechanistic model, calibrating and analyzing the model to generate new hypotheses that provide more data to refine the model. By using this approach, we were able to identify limitations with the first set of experimental data and suggest new experiments that improved our understanding of the JNK3 activation cascade. This result demonstrates the power of combining models with experimental approaches to understand the mechanisms that modulate biological processes. In the future, to gain more confidence in JARM, we could validate the prediction that 0.49 µM is the optimal concentration of the scaffold protein arrestin-3 for maximal JNK3 activation.

Biological experiments have uncertainties that during the calibration process are propagated to uncertainties in model parameter values. Additionally, as we build larger models, it is common to have many more parameters than experimental data, i.e., an underdetermined system, which results in model unidentifiability and large parameter uncertainties. These uncertainties are propagated to model predictions, and thus it is important to understand the mechanisms involved in the generation of different predictions. We tackle this issue by developing PyDyNo, a tool that tracks the signal flow throughout the network with different calibrated parameter vectors.

PyDyNo leverages tropical algebra, chemical kinetics, network analysis, and clustering algorithms to identify modes of signal execution in model networks. We applied this tool to the Extrinsic Apoptosis Reaction Model (EARM) and found that despite the many parameter vectors that fit the experimental data, there are only a few modes of signal execution in the model network. These different execution modes predict different outcomes when the same perturbation is applied to the system. Furthermore, we showed that calibrating biochemical models with a Bayesian approach enabled us to assign probabilities to each execution mode, significantly enhancing our understanding of signal dynamics. As a result, the probabilistic approach developed in this thesis may provide a novel framework to understand cellular signaling processes.

As execution modes from a model have a distinctive response, PyDyNo could be used with experimental design approaches to identify the most likely execution mode of a biological system and thus reduce the uncertainty in predictions. This approach could play an important role in the development of quantitative systems pharmacology models, as uncertainty reduction and high predictive power is necessary for drug development applications (J. P. F. Bai et al., 2019).

The execution modes identified by PyDyNo lead to the question: Do cells actually have different execution modes or are these execution modes a result of the uncertainty of the experiments and model identifiability? We believe that it is likely a combination of the two as cells might have different crowding environments (H.-X. Zhou et al., 2008) that can lead to different kinetics. As we showed in Chapter 4, different kinetic parameters might have unique execution modes. Also, these different execution modes can be part of a bet-hedging strategy (Jolly et al., 2018) where the kinetics heterogeneity can be viewed as an strategy that aims to optimizes the survival of a clonal population in a dynamic environment. Thus, as a future direction it would be important and interesting to experimentally determine whether the execution modes exist in cell populations.

Finally, previous works have explored the dynamic patterns of signal flow in complex networks. Erin Shockley and colleagues combined flux analysis with information theory to study how the COX-2 pathway integrates signals with varying input concentrations (Shockley et al., 2019). Harush and Barzel studied multiple networks with different topologies and used a perturbative formalism to analyze their signal flow at steady state (Harush & Barzel, 2017). Overall, these analyses have provided key biological and network insights about signal flow in complex networks. However, these approaches were developed to study signal flow at steady state conditions, omitting transient dynamics. These dynamics are important as, over time, they change the signal flow in a network and provide information about the optimal timing of perturbations to modulate a cell response. Our analysis provides a novel approach that considers the non-equilibrium dynamics of signaling networks and calibration parameter uncertainty to create a paradigm shift towards a more probabilistic understanding of signal execution in biochemical networks.

REFERENCES

Abdel-Basset, M., Abdel-Fatah, L., & Sangaiah, A. K. (2018). Chapter 10 - Metaheuristic Algorithms: A Comprehensive Review. In A. K. Sangaiah, M. Sheng, & Z. Zhang (Eds.), *Computational Intelligence for Multimedia Big Data on the Cloud with Engineering Applications* (pp. 185–231). Academic Press. https://doi.org/https://doi.org/10.1016/B978-0-12-813314-9.00010-4

Ahn, A. C., Tewari, M., Poon, C.-S., & Phillips, R. S. (2006). The Limits of Reductionism in Medicine: Could Systems Biology Offer an Alternative? *PLoS Medicine*, *3*(6), e208. https://doi.org/10.1371/journal.pmed.0030208

Ahn, J. W., Plaisant, C., & Shneiderman, B. (2014). A task taxonomy for network evolution analysis. *IEEE Transactions on Visualization and Computer Graphics*, *20*(3), 365–376. https://doi.org/10.1109/TVCG.2013.238

Albeck, J. G., Burke, J. M., Spencer, S. L., Lauffenburger, D. A., & Sorger, P. K. (2008). Modeling a Snap-Action, Variable-Delay Switch Controlling Extrinsic Cell Death. *PLoS Biology*, *6*(12), e299. https://doi.org/10.1371/journal.pbio.0060299

Aldridge, B. B., Burke, J. M., Lauffenburger, D. a, & Sorger, P. K. (2006). Physicochemical modelling of cell signalling pathways. *Nature Cell Biology*, *8*(11), 1195–1203. https://doi.org/10.1038/ncb1497

Alon, U. (2006). An Introduction to Systems Biology: Design Principles of Biological Circuits (1st ed.). In *Star* (Vol. 10). Chapman and Hall/CRC. https://doi.org/https://doi.org/10.1201/9781420011432

Annis, M. G., Soucie, E. L., Dlugosz, P. J., Cruz-Aguado, J. A., Penn, L. Z., Leber, B., & Andrews, D. W. (2005). Bax forms multispanning monomers that oligomerize to permeabilize membranes during apoptosis. *EMBO Journal*, *24*(12), 2096–2103. https://doi.org/10.1038/sj.emboj.7600675

Antoniou, X., Falconi, M., Di Marino, D., & Borsello, T. (2011). JNK3 as a Therapeutic Target for Neurodegenerative Diseases. *Journal of Alzheimer's Disease*, *24*, 633–642. https://doi.org/10.3233/JAD-2011-091567

Ashyraliyev, M., Fomekong-Nanfack, Y., Kaandorp, J. A., & Blom, J. G. (2009). Systems biology: Parameter estimation for biochemical models. *FEBS Journal*, *276*(4), 886–902. https://doi.org/10.1111/j.1742-4658.2008.06844.x

Bai, H., Yang, K., Yu, D., Zhang, C., Chen, F., & Lai, L. (2011). Predicting kinetic constants of protein-protein interactions based on structural properties. *Proteins: Structure, Function and Bioinformatics*, *79*(3), 720–734. https://doi.org/10.1002/prot.22904

Bai, J. P. F., Earp, J. C., & Pillai, V. C. (2019). Translational Quantitative Systems Pharmacology in Drug Development : from Current Landscape to Good Practices.

The AAPS Journal, 1–13. https://doi.org/10.1208/s12248-019-0339-5

Baker, M., & Penny, D. (2016). Is there a reproducibility crisis? *Nature*, *533*(7604), 452–454. https://doi.org/10.1038/533452A

Bar-Even, A., Noor, E., Savir, Y., Liebermeister, W., Davidi, D., Tawfik, D. S., & Milo, R. (2011). The Moderately Efficient Enzyme: Evolutionary and Physicochemical Trends Shaping Enzyme Parameters. *Biochemistry*, *50*(21), 4402–4410. https://doi.org/10.1021/bi2002289

Barabási, A. L., & Oltvai, Z. N. (2004). Network biology: Understanding the cell's functional organization. *Nature Reviews Genetics*, *5*(2), 101–113. https://doi.org/10.1038/nrg1272

Bartocci, E., & Lió, P. (2016). Computational Modeling, Formal Analysis, and Tools for Systems Biology. *PLoS Computational Biology*, *12*(1), 1–22. https://doi.org/10.1371/journal.pcbi.1004591

Barzel, B., & Barabási, A. L. (2013). Universality in network dynamics. *Nature Physics*, *9*(10), 673–681. https://doi.org/10.1038/nphys2741

Beck, F., Burch, M., Diehl, S., & Weiskopf, D. (2017). A Taxonomy and Survey of Dynamic Graph Visualization. *Computer Graphics Forum*, *36*(1), 133–159. https://doi.org/10.1111/cgf.12791

Becker, V., Schilling, M., Bachmann, J., Baumann, U., Raue, A., Maiwald, T., Timmer, J., & Klingmüller, U. (2010). Covering a broad dynamic range: Information processing at the erythropoietin receptor. *Science*, *328*(5984), 1404–1408. https://doi.org/10.1126/science.1184913

Bergmann, F. T., Hoops, S., Klahn, B., Kummer, U., Mendes, P., Pahle, J., & Sahle, S. (2017). COPASI and its applications in biotechnology. *Journal of Biotechnology*, *261*(June), 215–220. https://doi.org/10.1016/j.jbiotec.2017.06.1200

Bergroth, L., Hakonen, H., & Raita, T. (2000). A survey of longest common subsequence algorithms. *Proceedings - 7th International Symposium on String Processing and Information Retrieval, SPIRE 2000*, 39–48. https://doi.org/10.1109/SPIRE.2000.878178

Blinov, M. L., Faeder, J. R., Goldstein, B., & Hlavacek, W. S. (2006). A network model of early events in epidermal growth factor receptor signaling that accounts for combinatorial complexity. *BioSystems*, *83*(2-3 SPEC. ISS.), 136–151. https://doi.org/10.1016/j.biosystems.2005.06.014

Blondel, V. D., Guillaume, J. L., Lambiotte, R., & Lefebvre, E. (2008). Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, *2008*(10). https://doi.org/10.1088/1742-5468/2008/10/P10008

Boettiger, C. (2015). An Introduction to Docker for Reproducible Research. *SIGOPS Oper. Syst. Rev.*, *49*(1), 71–79. https://doi.org/10.1145/2723872.2723882

Bonneau, R. (2008). Learning biological networks: From modules to dynamics. *Nature Chemical Biology*, *4*(11), 658–664. https://doi.org/10.1038/nchembio.122

Boutillier, P., Maasha, M., Li, X., Medina-Abarca, H. F., Krivine, J., Feret, J., Cristescu, I., Forbes, A. G., & Fontana, W. (2018). The Kappa platform for rule-based modeling. *Bioinformatics*, *34*(13), i583–i592. https://doi.org/10.1093/bioinformatics/bty272

Brennan, M. D., Cheong, R., & Levchenko, A. (2012). Systems biology. How information theory handles cell signaling and uncertainty. *Science (New York, N.Y.)*, *338*(6105), 334–335. https://doi.org/10.1126/science.1227946

Campello, R. J. G. B., Moulavi, D., & Sander, J. (2013). Density-Based Clustering Based on Hierarchical Density Estimates. *Advances in Knowledge Discovery and Data Mining*, 160–172. https://doi.org/10.1007/978-3-642-37456-2_14

Cerami, E. G., Gross, B. E., Demir, E., Rodchenkov, I., Babur, Ö., Anwar, N., Schultz, N., Bader, G. D., & Sander, C. (2011). Pathway Commons, a web resource for biological pathway data. *Nucleic Acids Research*, *39*(SUPPL. 1), 685–690. https://doi.org/10.1093/nar/gkq1039

Chen, T., & Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 785–794. https://doi.org/10.1145/2939672.2939785

Chen, W., Niepel, M., & Sorger, P. (2010). Classic and contemporary approaches to modeling biochemical reactions. *Genes & Development*, *24*, 1861–1875. https://doi.org/10.1101/gad.1945410.Freely

Cheng, H. C., Angermann, B. R., Zhang, F., & Meier-Schellersheim, M. (2014). NetworkViewer: Visualizing biochemical reaction networks with embedded rendering of molecular interaction rules. *BMC Systems Biology*, *8*(1), 1–16. https://doi.org/10.1186/1752-0509-8-70

Cheong, R., Rhee, A., Wang, C. J., Nemenman, I., & Levchenko, A. (2011). Information Transduction Capacity of Noisy Biochemical Signaling Networks. *Science*, *334*(6054), 354–358. https://doi.org/10.1126/science.1204553

Chiband, S., & Greenberg, E. (2008). *Understanding the Metropolis-Hastings Algorithm*. *49*(4), 327–335.

Choi, K., Medley, J. K., König, M., Stocking, K., Smith, L., Gu, S., & Sauro, H. M. (2018). Tellurium: An extensible python-based modeling environment for systems and synthetic biology. *BioSystems*, *171*(July), 74–79. https://doi.org/10.1016/j.biosystems.2018.07.006

Collins, F. S., & McKusick, V. A. (2001). Implications of the Human Genome Project for Medical Science. *JAMA*, *285*(5), 540–544.

Cordasco, G., & Gargano, L. (2010). Community detection via semi-synchronous label propagation algorithms. *2010 IEEE International Workshop on Business Applications of Social Network Analysis, BASNA 2010*. https://doi.org/10.1109/BASNA.2010.5730298

Dang, T. N., Murray, P., Aurisano, J., & Forbes, A. G. (2015). ReactionFlow: An interactive visualization tool for causality analysis in biological pathways. *BMC Proceedings*, *9*(Suppl 6), 1–18. https://doi.org/10.1186/1753-6561-9-S6-S6

Danos, V., Feret, J., Fontana, W., Harmer, R., Hayman, J., Krivine, J., Thompson-Walsh, C., & Winskel, G. (2012). Graphs, Rewriting and Pathway Reconstruction for Rule-Based Models. *FSTTCS 2012 - IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science*, *18*(Fsttcs), 276–288. https://doi.org/10.4230/LIPIcs.FSTTCS.2012.276

Däschinger, M., Knote, A., Green, R., & Von Mammen, S. (2017). *A human-in-the-loop environment for developmental biology*. *September*, 475–482. https://doi.org/10.7551/ecal_a_078

Deb, K. (1999). An introduction to genetic algorithms. *Sadhana*, *24*(4), 293–315. https://doi.org/10.1007/BF02823145

Dekkers, A., & Aarts, E. (1991). Global optimization and simulated annealing. *Mathematical Programming*, *50*(1), 367–393. https://doi.org/10.1007/BF01594945

Demir, E., Cary, M. P., Paley, S., Fukuda, K., Lemer, C., Vastrik, I., Wu, G., D'Eustachio, P., Schaefer, C., Luciano, J., Schacherer, F., Martinez-Flores, I., Hu, Z., Jimenez-Jacinto, V., Joshi-Tope, G., Kandasamy, K., Lopez-Fuentes, A. C., Mi, H., Pichler, E., … Bader, G. D. (2010). The BioPAX community standard for pathway data sharing. *Nature Biotechnology*, *28*(9), 935–942. https://doi.org/10.1038/nbt.1666

Deng, Y., Lin, Y., & Wu, X. (2002). TRAIL-induced apoptosis requires Bax-dependent mitochondrial release of Smac/DIABLO. *Genes and Development*, *16*(1), 33–45. https://doi.org/10.1101/gad.949602

Dewson, G., Kratina, T., Czabotar, P., Day, C. L., Adams, J. M., & Kluck, R. M. (2009). Bak Activation for Apoptosis Involves Oligomerization of Dimers via Their α6 Helices. *Molecular Cell*, *36*(4), 696–703. https://doi.org/https://doi.org/10.1016/j.molcel.2009.11.008

Dhanasekaran, D. N., & Reddy, E. P. (2008). JNK signaling in apoptosis. *Oncogene*, *27*(48), 6245–6251. https://doi.org/10.1038/onc.2008.301

Dogrusoz, U., Karacelik, A., Safarli, I., Balci, H., Dervishi, L., & Siper, C. (2018). *Efficient methods and readily customizable libraries for managing complexity of large networks*. 1–18.

Droste, P., Miebach, S., Niedenfuhr, S., Wiechert, W., & Noh, K. (2011). Visualizing multi-omics data in metabolic networks with the software Omix—A case study. *Biosystems*, *105*(2), 154–161.

Egea, J. A., Rodríguez-Fernández, M., Banga, J. R., & Martí, R. (2007). Scatter search for chemical and bio-process optimization. *Journal of Global Optimization*, *37*(3), 481–503. https://doi.org/10.1007/s10898-006-9075-3

Eidum, D., Asthana, K., Unni, S., Deng, M., & You, L. (2014). Construction, visualization, and analysis of biological network models in Dynetica. *Quantitative Biology*, *2*(4), 142–150. https://doi.org/10.1007/s40484-014-0036-4

Elmore, S. (2007). Apoptosis: A Review of Programmed Cell Death. *Toxicologic Pathology*, *35*(4), 495–516. https://doi.org/10.1080/01926230701320337

Eydgahi, H., Chen, W. W., Muhlich, J. L., Vitkup, D., Tsitsiklis, J. N., & Sorger, P. K. (2013). Properties of cell death models calibrated and compared using Bayesian approaches. *Molecular Systems Biology*, *9*(644), 644. https://doi.org/10.1038/msb.2012.69

Faeder, J. R., Blinov, M. L., & Hlavacek, W. S. (2009). Rule-based modeling of biochemical systems with BioNetGen. *Methods in Molecular Biology (Clifton, N.J.)*, *500*, 113–167. https://doi.org/10.1007/978-1-59745-525-1_5

Forbes, A. G., Burks, A., Lee, K., Li, X., Boutillier, P., Krivine, J., & Fontana, W. (2017). Dynamic Influence Networks for Rule-based Models. *IEEE Transactions on Visualization and Computer Graphics*, *2626*(c). https://doi.org/10.1109/TVCG.2017.2745280

Fortunato, S. (2010). Community detection in graphs. *Physics Reports*, *486*(3–5), 75–174. https://doi.org/10.1016/j.physrep.2009.11.002

Franz, M., Lopes, C. T., Huck, G., Dong, Y., Sumer, O., & Bader, G. D. (2015). Cytoscape.js: A graph theory library for visualisation and analysis. *Bioinformatics*, *32*(2), 309–311. https://doi.org/10.1093/bioinformatics/btv557

Gabadinho, A., Ritschard, G., Studer, M., & Müller, N. S. (2011). *Extracting and Rendering Representative Sequences BT - Knowledge Discovery, Knowlege Engineering and Knowledge Management* (A. Fred, J. L. G. Dietz, K. Liu, & J. Filipe (eds.); pp. 94–106). Springer Berlin Heidelberg.

Gaddy, T. D., Wu, Q., Arnheim, A. D., & Finley, S. D. (2017). Mechanistic modeling quantifies the influence of tumor growth kinetics on the response to anti-angiogenic treatment. *PLoS Computational Biology*, *13*(12), 1–23. https://doi.org/10.1371/journal.pcbi.1005874

Garrido, C., Galluzzi, L., Brunet, M., Puig, P. E., Didelot, C., & Kroemer, G. (2006). Mechanisms of cytochrome c release from mitochondria. *Cell Death & Differentiation*, *13*(9), 1423–1433. https://doi.org/10.1038/sj.cdd.4401950

Gayvert, K. M., Aly, O., Platt, J., Bosenberg, M. W., Stern, D. F., & Elemento, O. (2017). A Computational Approach for Identifying Synergistic Drug Combinations. *PLOS Computational Biology*, *13*(1), e1005308. https://doi.org/10.1371/journal.pcbi.1005308

Gelman, A., & Rubin, D. B. (1992). Inference from iterative simulation using multiple sequences. *Statistical Science*, 7(4), 457–472. https://doi.org/10.1214/ss/1177011136

Girvan, M., & Newman, M. E. J. (2002). Community structure in social and biological networks. *Proceedings of the National Academy of Sciences of the United States of America*, *99*(12), 7821–7826. https://doi.org/10.1073/pnas.122653799

Gizzatkulov, N. M., Goryanin, I. I., Metelkin, E. A., Mogilevskaya, E. A., Peskov, K. V, & Demin, O. V. (2010). DBSolve Optimum: a software package for kinetic modeling which allows dynamic visualization of simulation results. *BMC Systems Biology*, *4*(1), 109. https://doi.org/10.1186/1752-0509-4-109

Gutenkunst, R. N., Waterfall, J. J., Casey, F. P., Brown, K. S., Myers, C. R., & Sethna, J. P. (2007a). Universally sloppy parameter sensitivities in systems biology models. *PLoS Computational Biology*, *3*(10), 1871–1878. https://doi.org/10.1371/journal.pcbi.0030189

Gutenkunst, R. N., Waterfall, J. J., Casey, F. P., Brown, K. S., Myers, C. R., & Sethna, J. P. (2007b). Universally sloppy parameter sensitivities in systems biology models - Supporting Text 3 : Fragility of Other Predictions Fraction of Ras active Fraction of Mek active. *PLoS Computational Biology*, *3*(10), 3–4.

Hagberg, A. A., Schult, D. A., & Swart, P. J. (2008). Exploring network structure, dynamics, and function using NetworkX. *Proceedings of the 7th Python in Science Conference (SciPy)*, *SciPy*, 11–15. https://doi.org/10.1016/j.jelectrocard.2010.09.003

Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M. H., Brett, M., Haldane, A., del Río, J. F., Wiebe, M., Peterson, P., … Oliphant, T. E. (2020). Array programming with NumPy. *Nature*, *585*(7825), 357–362. https://doi.org/10.1038/s41586-020-2649-2

Harris, L. A., Hogg, J. S., Tapia, J. J., Sekar, J. A. P., Gupta, S., Korsunsky, I., Arora, A., Barua, D., Sheehan, R. P., & Faeder, J. R. (2016). BioNetGen 2.2: Advances in rule-based modeling. *Bioinformatics*, *32*(21), 3366–3368. https://doi.org/10.1093/bioinformatics/btw469

Harush, U., & Barzel, B. (2017). Dynamic patterns of information flow in complex networks. *Nature Communications*, *8*(1), 1–11. https://doi.org/10.1038/s41467-017-01916-3

Heinemann, T., & Raue, A. (2016). Model calibration and uncertainty analysis in signaling networks. *Current Opinion in Biotechnology*, *39*, 143–149. https://doi.org/10.1016/j.copbio.2016.04.004

Ho, D. T., Bardwell, A. J., Grewal, S., Iverson, C., & Bardwell, L. (2006). Interacting JNK-docking sites in MKK7 promote binding and activation of JNK mitogen-activated protein kinases. *The Journal of Biological Chemistry*, *281*(19), 13169–13179. https://doi.org/10.1074/jbc.M601010200

Holzinger, A. (2016). Interactive machine learning for health informatics: when do we need the human-in-the-loop? *Brain Informatics*, *3*(2), 119–131. https://doi.org/10.1007/s40708-016-0042-6

Hucka, M., Finney, A., Sauro, H. M., Bolouri, H., Doyle, J. C., Kitano, H., Arkin, A. P., Bornstein, B. J., Bray, D., Cornish-Bowden, A., Cuellar, A. A., Dronov, S., Gilles, E. D., Ginkel, M., Gor, V., Goryanin, I. I., Hedley, W. J., Hodgman, T. C., Hofmeyr, J. H., … Wang, J. (2003). The systems biology markup language (SBML): A medium for representation and exchange of biochemical network models. *Bioinformatics*, *19*(4), 524–531. https://doi.org/10.1093/bioinformatics/btg015

Janes, K. A., Albeck, J. G., Gaudet, S., Sorger, P. K., Lauffenburger, D. A., & Yaffe, M. B. (2005). Cell signaling: A systems model of signaling identifies a molecular basis set for cytokine-induced apoptosis. *Science*, *310*(5754), 1646–1653. https://doi.org/10.1126/science.1116598

Janes, K. A., Reinhardt, H. C., & Yaffe, M. B. (2008). Cytokine-Induced Signaling Networks Prioritize Dynamic Range over Signal Strength. *Cell*, *135*(2), 343–354. https://doi.org/10.1016/j.cell.2008.08.034

Janes, K. a, Albeck, J. G., Gaudet, S., Sorger, P. K., Lauffenburger, D. a, & Yaffe, M. B. (2005). A systems model of signaling identifies a molecular basis set for cytokine-induced apoptosis. *Science (New York, N.Y.)*, *310*(5754), 1646–1653. https://doi.org/10.1126/science.1116598

Jaqaman, K., & Danuser, G. (2006). Linking data to models: Data regression. *Nature Reviews Molecular Cell Biology*, *7*(11), 813–819. https://doi.org/10.1038/nrm2030

Jenny, B., Stephen, D. M., Muehlenhaus, I., Marston, B. E., Sharma, R., Zhang, E., & Jenny, H. (2018). Design principles for origin-destination flow maps. *Cartography and Geographic Information Science*, *45*(1), 62–75. https://doi.org/10.1080/15230406.2016.1262280

Jolly, M. K., Kulkarni, P., Weninger, K., Orban, J., & Levine, H. (2018). Phenotypic plasticity, bet-hedging, and androgen independence in prostate cancer: Role of non-genetic heterogeneity. *Frontiers in Oncology*, *8*(MAR), 1–12. https://doi.org/10.3389/fonc.2018.00050

Jordan, J. D., Landau, E. M., & Iyengar, R. (2000). Signaling networks: The origins of cellular multitasking. *Cell*, *103*(2), 193–200. https://doi.org/10.1016/S0092-8674(00)00112-4

Jupyter, P., Bussonnier, M., Forde, J., Freeman, J., Granger, B., Head, T., Holdgraf, C., Kelley, K., Nalvarte, G., Osheroff, A., Pacer, M., Panda, Y., Perez, F., Ragan-Kelley, B., & Willing, C. (2018). Binder 2.0 - Reproducible, interactive, sharable environments for science at scale. *Proceedings of the 17th Python in Science Conference*, *Scipy*, 113–120. https://doi.org/10.25080/majora-4af1f417-011

Kale, J., Osterlund, E. J., & Andrews, D. W. (2017). BCL-2 family proteins: changing partners in the dance towards death. *Cell Death and Differentiation*, *25*(1), 65–80. https://doi.org/10.1038/cdd.2017.186

Kantari, C., & Walczak, H. (2011). Caspase-8 and Bid: Caught in the act between death receptors and mitochondria. *Biochimica et Biophysica Acta (BBA) - Molecular Cell Research*, *1813*(4), 558–563. https://doi.org/https://doi.org/10.1016/j.bbamcr.2011.01.026

Kato, T., Tsujimoto, S., & Zuk, A. (2017a). Spectral Analysis of Transition Operators, Automata Groups and Translation in BBS. *Communications in Mathematical Physics*, *350*(1), 205–229. https://doi.org/10.1007/s00220-016-2702-z

Kato, T., Tsujimoto, S., & Zuk, A. (2017b). Spectral Analysis of Transition Operators, Automata Groups and Translation in BBS. *Communications in Mathematical Physics*, *350*(1), 205–229. https://doi.org/10.1007/s00220-016-2702-z

Kaufmann, S. H., Desnoyers, S., Ottaviano, Y., Davidson, N. E., & Poirier, G. G. (1993). Specific Proteolytic Cleavage of Poly(ADP-ribose) Polymerase: An Early Marker of Chemotherapy-induced Apoptosis. *Cancer Research*, *53*(17), 3976–3985.

Kennedy, J., & Eberhart, R. (1995). Particle swarm optimization. *Neural Networks, 1995. Proceedings., IEEE International Conference On*, *4*, 1942–1948 vol.4. https://doi.org/10.1109/ICNN.1995.488968

Keshet, Y., & Seger, R. (2010). The MAP Kinase Signaling Cascades: A System of Hundreds of Components Regulates a Diverse Array of Physiological Functions. In R. Seger (Ed.), *MAP Kinase Signaling Protocols: Second Edition* (pp. 3–38). Humana Press. https://doi.org/10.1007/978-1-60761-795-2_1

Khalid, S., Drasche, A., Thurner, M., Hermann, M., Ashraf, M. I., Fresser, F., Baier, G., Kremser, L., Lindner, H., & Troppmair, J. (2016). cJun N-terminal kinase (JNK) phosphorylation of serine 36 is critical for p66Shc activation. *Scientific Reports*, *6*, 20930. https://doi.org/10.1038/srep20930

Kholodenko, B. N., Demin, O. V., Moehren, G., & Hoek, J. B. (1999). Quantification of short term signaling by the epidermal growth factor receptor. *Journal of Biological Chemistry*, *274*(42), 30169–30181. https://doi.org/10.1074/jbc.274.42.30169

Kitano, H. (2002). Systems Biology: A brief overview. *Science*, *295*(March), 1662–1664.

Kitano, H., Funahashi, A., Matsuoka, Y., & Oda, K. (2005). Using process diagrams for the graphical representation of biological networks. *Nature Biotechnology*, *23*(8), 961–966. https://doi.org/10.1038/nbt1111

Klinke, D. J. (2010). Signal transduction networks in cancer: Quantitative parameters influence network topology. *Cancer Research*, *70*(5), 1773–1782. https://doi.org/10.1158/0008-5472.CAN-09-3234

Kluyver, T., Ragan-kelley, B., Pérez, F., Granger, B. E., Bussonnier, M., Frederic, J., Kelley, K., Hamrick, J., Grout, J., Corlay, S., Ivanov, P., Avila, D., Abdalla, S., & Willing, C. (2016). Jupyter Notebooks—a publishing format for reproducible computational workflows. *Positioning and Power in Academic Publishing: Players, Agents and Agendas*, 87–90. https://doi.org/10.3233/978-1-61499-649-1-87

Knuth, D. E. (2001). Literate Programming. *The Computer Journal*, *27*(2), 97–111. http://www.literateprogramming.com/knuthweb.pdf%5Cnpapers2://publication/uuid/1054D560-6236-44D9-9459-EFC794CD28DF

Kolpakov, F., Akberdin, I., Kashapov, T., Kiselev, L., Kolmykov, S., Kondrakhin, Y., Kutumova, E., Mandrik, N., Pintus, S., Ryabova, A., Sharipov, R., Yevshin, I., & Kel, A. (2019). BioUML: an integrated environment for systems biology and collaborative analysis of biomedical data. *Nucleic Acids Research*, *47*(W1), W225–W233. https://doi.org/10.1093/nar/gkz440

König, M., Dräger, A., & Holzhütter, H. G. (2012). CySBML: A cytoscape plugin for SBML. *Bioinformatics*, *28*(18), 2402–2403. https://doi.org/10.1093/bioinformatics/bts432

Koonin, E., & Aravind, L. (2002). Origin and evolution of eukaryotic apoptosis: the bacterial connection. *Cell Death and Differentiation*, *9*, 394–404. https://doi.org/10.1038/sj/cdd/4400991

Lee, M. J., Ye, A. S., Gardino, A. K., Heijink, A. M., Sorger, P. K., MacBeath, G., & Yaffe, M. B. (2012). Sequential application of anticancer drugs enhances cell death by rewiring apoptotic signaling networks. *Cell*, *149*(4), 780–794. https://doi.org/10.1016/j.cell.2012.03.031

Legrain, P., Aebersold, R., Archakov, A., Bairoch, A., Bala, K., Beretta, L., Bergeron, J., Borchers, C. H., Corthals, G. L., Costello, C. E., Deutsch, E. W., Domon, B., Hancock, W., He, F., Hochstrasser, D., Marko-Varga, G., Salekdeh, G. H., Sechi, S., Snyder, M., … Omenn, G. S. (2011). The Human Proteome Project: Current State and Future Direction. *Molecular & Cellular Proteomics*, *10*(7), M111.009993. https://doi.org/https://doi.org/10.1074/mcp.M111.009993

Lei, K., Nimnual, A., Zong, W.-X., Kennedy, N. J., Flavell, R. A., Thompson, C. B., Bar-Sagi, D., & Davis, R. J. (2002). The Bax subfamily of Bcl2-related proteins is essential for apoptotic signal transduction by c-Jun NH(2)-terminal kinase. *Molecular and Cellular Biology*, *22*(13), 4929–4942. https://doi.org/10.1128/mcb.22.13.4929-4942.2002

Lemmon, M. A., & Schlessinger, J. (2010). Cell signaling by receptor tyrosine kinases. *Cell*, *141*(7), 1117–1134. https://doi.org/10.1016/j.cell.2010.06.011

Levchenko, A., & Nemenman, I. (2014). Cellular noise and information transmission. *Current Opinion in Biotechnology*, *28C*, 156–164. https://doi.org/10.1016/j.copbio.2014.05.002

Locasale, J. W., Shaw, A. S., & Chakraborty, A. K. (2007). Scaffold proteins confer diverse regulatory properties to protein kinase cascades. *Proceedings of the National Academy of Sciences*, *104*(33), 13307–13312. https://doi.org/10.1073/pnas.0706311104

Lopez, C. F., Muhlich, J. L., Bachman, J. A., & Sorger, P. K. (2013). Programming biological models in Python using PySB. *Molecular Systems Biology*, *9*(646), 1–19. https://doi.org/10.1038/msb.2013.1

Machado, D., Costa, R. S., Rocha, M., Ferreira, E. C., Tidor, B., & Rocha, I. (2011). Modeling formalisms in systems biology. *AMB Express*, *1*(1), 1–14. https://doi.org/10.1186/2191-0855-1-45

Madabushi, R., Benjamin, J. M., Grewal, R., Pacanowski, M. A., Strauss, D. G., Wang, Y., Zhu, H., & Zineh, I. (2019). The US Food and Drug Administration's Model-Informed Drug Development Paired Meeting Pilot Program: Early Experience and Impact. *Clinical Pharmacology and Therapeutics*, *106*(1), 74–78. https://doi.org/10.1002/cpt.1457

Mandel, J. J., Fuß, H., Palfreyman, N. M., & Dubitzky, W. (2007). Modeling biochemical transformation processes and information processing with Narrator. *BMC Bioinformatics*, *8*(1), 103. https://doi.org/10.1186/1471-2105-8-103

Medley, J. K., Choi, K., König, M., Smith, L., Gu, S., Hellerstein, J., Sealfon, S. C., & Sauro, H. M. (2018). Tellurium notebooks—An environment for reproducible dynamical modeling in systems biology. *PLoS Computational Biology*, *14*(6), 1–24. https://doi.org/10.1371/journal.pcbi.1006220

Mehal, W. Z., Inayat, I., & Flavell, R. A. (2006). Caspases 3 and 7: Key Mediators of Mitochondrial Events of Apoptosis. *Science*, *February*, 847–851.

Meurer, A., Smith, C. P., Paprocki, M., Čertík, O., Kirpichev, S. B., Rocklin, M., Kumar, Am., Ivanov, S., Moore, J. K., Singh, S., Rathnayake, T., Vig, S., Granger, B. E., Muller, R. P., Bonazzi, F., Gupta, H., Vats, S., Johansson, F., Pedregosa, F., … Scopatz, A. (2017). SymPy: symbolic computing in Python. *PeerJ Computer Science*, *3*, e103. https://doi.org/10.7717/peerj-cs.103

Meza, J. C. (2011). Newton's method. *WIREs Computational Statistics*, *3*(1), 75–78. https://doi.org/https://doi.org/10.1002/wics.129

Mitra, E. D., & Hlavacek, W. S. (2019a). Parameter estimation and uncertainty quantification for systems biology models. *Current Opinion in Systems Biology*, *18*, 9–18. https://doi.org/10.1016/j.coisb.2019.10.006

Mitra, E. D., & Hlavacek, W. S. (2019b). Parameter estimation and uncertainty quantification for systems biology models. In *Current Opinion in Systems Biology* (Vol. 18, pp. 9–18). Elsevier Ltd. https://doi.org/10.1016/j.coisb.2019.10.006

Mitra, E. D., Suderman, R., Colvin, J., Ionkov, A., Hu, A., Sauro, H. M., Posner, R. G., & Hlavacek, W. S. (2019). PyBioNetFit and the Biological Property Specification Language. *IScience*, *19*, 1012–1036. https://doi.org/10.1016/j.isci.2019.08.045

Morange, M. (1998). A history of molecular biology / Michel Morange ; translated by Matthew Cobb. In *Molecular biology*. Harvard University Press.

More, J. J. (1978). Levenberg--Marquardt algorithm: implementation and theory. In Watson G.A. (Ed.), *Numerical analysis* (Vol. 630). Springer Berlin Heidelberg. https://doi.org/https://doi.org/10.1007/BFb0067700

Muniyappa, H., & Das, K. C. (2008). Activation of c-Jun N-terminal kinase (JNK) by widely used specific p38 MAPK inhibitors SB202190 and SB203580: a MLK-3-MKK7-dependent mechanism. *Cellular Signalling*, *20*(4), 675–683. https://doi.org/10.1016/j.cellsig.2007.12.003

Murray, P., McGee, F., & Forbes, A. G. (2017a). A taxonomy of visualization tasks for the analysis of biological pathway data. *BMC Bioinformatics*, *18*(2), 21. https://doi.org/10.1186/s12859-016-1443-5

Murray, P., McGee, F., & Forbes, A. G. (2017b). A taxonomy of visualization tasks for the analysis of biological pathway data. *BMC Bioinformatics*, *18*(S2), 21. https://doi.org/10.1186/s12859-016-1443-5

Neumann, L., Pforr, C., Beaudouin, J., Pappa, A., Fricker, N., Krammer, P. H., Lavrik, I. N., & Eils, R. (2010). Dynamics within the CD95 death-inducing signaling complex decide life and death of cells. *Molecular Systems Biology*, *6*(352), 352. https://doi.org/10.1038/msb.2010.6

Nobeli, I., Favia, A. D., & Thornton, J. M. (2009). Protein promiscuity and its implications for biotechnology. *Nature Biotechnology*, *27*(2), 157–167. https://doi.org/10.1038/nbt1519

Noel, V., Grigoriev, D., & Vakulenko, S. (2011). *Tropical geometries and dynamics of biochemical networks . Application to hybrid cell cycle models* . 1–19.

Olivier, B. G., Rohwer, J. M., & Hofmeyr, J. H. S. (2005). Modelling cellular systems with PySCeS. *Bioinformatics*, *21*(4), 560–561. https://doi.org/10.1093/bioinformatics/bti046

Özören, N., & El-Deiry, W. S. (2002). Defining characteristics of types I and II apoptotic cells in response to TRAIL. *Neoplasia*, *4*(6), 551–557. https://doi.org/10.1038/sj.neo.7900270

Paduano, F., & Forbes, A. G. (2015). Extended LineSets: A visualization technique for the interactive inspection of biological pathways. *BMC Proceedings*, *9*(Suppl 6), 1–13. https://doi.org/10.1186/1753-6561-9-S6-S4

Parés, F., Gasulla, D. G., Vilalta, A., Moreno, J., Ayguadé, E., Labarta, J., Cortés, U., & Suzumura, T. (2018). Fluid communities: A competitive, scalable and diverse community detection algorithm. *Studies in Computational Intelligence*, *689*, 229–240. https://doi.org/10.1007/978-3-319-72150-7_19

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, É. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, *12*(85), 2825–2830. http://jmlr.org/papers/v12/pedregosa11a.html

Pennarun, B., Meijer, A., de Vries, E. G. E., Kleibeuker, J. H., Kruyt, F., & de Jong, S. (2010). Playing the DISC: Turning on TRAIL death receptor-mediated apoptosis in cancer. *Biochimica et Biophysica Acta (BBA) - Reviews on Cancer*, *1805*(2), 123–140. https://doi.org/https://doi.org/10.1016/j.bbcan.2009.11.004

Perry, D. K., Smyth, M. J., Stennicke, H. R., Salvesen, G. S., Duriez, P., Poirier, G. G., & Hannun, Y. A. (1997). Zinc is a potent inhibitor of the apoptotic protease, caspase-3: A novel target for zinc in the inhibition of apoptosis. *Journal of Biological Chemistry*, *272*(30), 18530–18533. https://doi.org/10.1074/jbc.272.30.18530

Perry, N. A., Kaoud, T. S., Ortega, O. O., Kaya, A. I., Marcus, D. J., Pleinis, J. M., Berndt, S., Chen, Q., Zhan, X., Dalby, K. N., Lopez, C. F., Iverson, T. M., & Gurevich, V. V. (2019). Arrestin-3 scaffolding of the JNK3 cascade suggests a mechanism for signal amplification. *Proceedings of the National Academy of Sciences*, *116*(3), 810–815. https://doi.org/10.1073/pnas.1819230116

Planck, M., & Luxburg, U. Von. (2006). A Tutorial on Spectral Clustering A Tutorial on Spectral Clustering. *Statistics and Computing*, *17*(March), 395–416. https://doi.org/10.1007/s11222-007-9033-z

Raghavan, U. N., Albert, R., & Kumara, S. (2007). Near linear time algorithm to detect community structures in large-scale networks. *Physical Review E - Statistical, Nonlinear, and Soft Matter Physics*, *76*(3), 1–11. https://doi.org/10.1103/PhysRevE.76.036106

Raue, A., Kreutz, C., Maiwald, T., Bachmann, J., Schilling, M., Klingmüller, U., & Timmer, J. (2009). Structural and practical identifiability analysis of partially observed dynamical models by exploiting the profile likelihood. *Bioinformatics*, *25*(15), 1923–1929. https://doi.org/10.1093/bioinformatics/btp358

Raue, A., Kreutz, C., Maiwald, T., Klingmuller, U., & Timmer, J. (2011). Addressing parameter identifiability by model-based experimentation. *IET Systems Biology*, *5*(2), 120–130. https://doi.org/10.1049/iet-syb.2010.0061

Read, M. N., Alden, K., Timmis, J., & Andrews, P. S. (2018). Strategies for calibrating models of biology. *Briefings in Bioinformatics*, *21*(1), 24–35. https://doi.org/10.1093/bib/bby092

Rokach, L., & Maimon, O. (2005). Clustering methods. *Data Mining and Knowledge Discovery Handbook*, 321–352. https://doi.org/10.1007/0-387-25465-X_15

Rousseeuw, P. J. (1987). Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, *20*(C), 53–65.

https://doi.org/10.1016/0377-0427(87)90125-7

Sachs, K., Perez, O., Pe'er, D., Lauffenburger, D. A., & Nolan, G. P. (2005). Causal protein-signaling networks derived from multiparameter single-cell data.[see comment][erratum appears in Science. 2005 Aug 19;309(5738):1187]. *Science*, *308*(5721), 523–529.

Sarah, C. (2017). Cancer: Identifying synergistic drug combinations. In *Nature reviews. Drug discovery* (Vol. 16, Issue 5, p. 314). NLM (Medline). https://doi.org/10.1038/nrd.2017.76

Schaff, J. C., Vasilescu, D., Moraru, I. I., Loew, L. M., & Blinov, M. L. (2016). Rule-based modeling with Virtual Cell. *Bioinformatics*, *32*(18), 2880–2882. https://doi.org/10.1093/bioinformatics/btw353

Schreiber, G., Haran, G., & Zhou, H.-X. (2009). Fundamental aspects of protein-protein association kinetics. *Chemical Reviews*, *109*(3), 839–860. https://doi.org/10.1021/cr800373w

Sekar, J. A. P., Tapia, J. J., & Faeder, J. R. (2017). Automated visualization of rule-based models. *PLoS Computational Biology*, *13*(11), 1–23. https://doi.org/10.1371/journal.pcbi.1005857

Shannon, C. E. (1948). A Mathematical Theory of Communication. *The Bell System Technical Journal*, *27*(July 1928), 379–423.

Shockley, E. M., Rouzer, C. A., Marnett, L. J., Deeds, E. J., & Lopez, C. F. (2019). Signal integration and information transfer in an allosterically regulated network. *Npj Systems Biology and Applications*, *5*(1). https://doi.org/10.1038/s41540-019-0100-9

Shockley, E. M., Vrugt, J. A., & Lopez, C. F. (2018). PyDREAM: High-dimensional parameter inference for biological models in python. *Bioinformatics*, *34*(4), 695–697. https://doi.org/10.1093/bioinformatics/btx626

Singh, R., Letai, A., & Sarosiek, K. (2019). Regulation of apoptosis in health and disease: the balancing act of BCL-2 family proteins. *Nature Reviews Molecular Cell Biology*, *20*(3), 175–193. https://doi.org/10.1038/s41580-018-0089-8

Smith, A. M., Xu, W., Sun, Y., Faeder, J. R., & Marai, G. E. (2012). RuleBender: integrated modeling, simulation and visualization for rule-based intracellular biochemistry. *BMC Bioinformatics*, *13*(Suppl 8), S3. https://doi.org/10.1186/1471-2105-13-S8-S3

Solania, A., González-Paéz, G. E., & Wolan, D. W. (2019). Selective and Rapid Cell-Permeable Inhibitor of Human Caspase-3. *ACS Chemical Biology*, *14*(11), 2463–2470. https://doi.org/10.1021/acschembio.9b00564

Sorger, P. K., Allerheiligen, S. R. ., Abernethy, D. R., Altman, R. B., Brouwer, K. L. ., Califano, A., D'Argenio, D. Z., Iyengar, R., Jusko, W. J., Lalonde, R., Lauffenburger, D. A., Shoichet, B., Stevens, J. L., Subramaniam, S., Van der Graaf, P., & Vicini, P. (2011).

Quantitative and Systems Pharmacology in the Post-genomic Era: New Approaches to Discovering Drugs and Understanding Therapeutic Mechanisms (National Institutes of Health, Bethesda, MD, 2011). In *National Institutes of Health*. https://doi.org/10.12693/APhysPolA.102.295

Spencer, S. L., Gaudet, S., Albeck, J. G., Burke, J. M., & Sorger, P. K. (2009). Non-genetic origins of cell-to-cell variability in TRAIL-induced apoptosis. *Nature*, *459*(7245), 428–432. https://doi.org/10.1038/nature08012

Stites, E. C., Aziz, M., Creamer, M. S., Von Hoff, D. D., Posner, R. G., & Hlavacek, W. S. (2015). Use of mechanistic models to integrate and analyze multiple proteomic datasets. *Biophysical Journal*, *108*(7), 1819–1829. https://doi.org/10.1016/j.bpj.2015.02.030

Storn, R., & Price, K. (1997). Differential Evolution – A Simple and Efficient Heuristic for global Optimization over Continuous Spaces. *Journal of Global Optimization*, *11*(4), 341–359. https://doi.org/10.1023/A:1008202821328

Strasen, J., Sarma, U., Jentsch, M., Bohn, S., Sheng, C., Horbelt, D., Knaus, P., Legewie, S., & Loewer, A. (2018). Cell-specific responses to the cytokine TGFβ are determined by variability in protein levels. *Molecular Systems Biology*, *14*(1), e7733. https://doi.org/10.15252/msb.20177733

Studer, M., & Ritschard, G. (2015). What matters in differences between life trajectories: a comparative review of sequence dissimilarity measures. *Journal of the Royal Statistical Society: Series A (Statistics in Society)*, n/a-n/a. https://doi.org/10.1111/rssa.12125

Suderman, R., Bachman, J. A., Smith, A., Sorger, P. K., & Deeds, E. J. (2017). Fundamental trade-offs between information flow in single cells and cellular populations. *Proceedings of the National Academy of Sciences*, *114*(22), 5755–5760. https://doi.org/10.1073/pnas.1615660114

Sun, J., Garibaldi, J. M., & Hodgman, C. (2012). Parameter estimation using metaheuristics in systems biology: A comprehensive review. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, *9*(1), 185–202. https://doi.org/10.1109/TCBB.2011.63

Szklarczyk, D., Gable, A. L., Lyon, D., Junge, A., Wyder, S., Huerta-Cepas, J., Simonovic, M., Doncheva, N. T., Morris, J. H., Bork, P., Jensen, L. J., & Mering, C. von. (2019). STRING v11: protein-protein association networks with increased coverage, supporting functional discovery in genome-wide experimental datasets. *Nucleic Acids Research*, *47*(D1), D607–D613. https://doi.org/10.1093/nar/gky1131

Tiger, C. F., Krause, F., Cedersund, G., Palmér, R., Klipp, E., Hohmann, S., Kitano, H., & Krantz, M. (2012). A framework for mapping, visualisation and automatic model creation of signal-transduction networks. *Molecular Systems Biology*, *8*(578), 1–20. https://doi.org/10.1038/msb.2012.12

Tiwari, K., Kananathan, S., Roberts, M. G., Meyer, J. P., Sharif Shohan, M. U., Xavier, A., Maire, M., Zyoud, A., Men, J., Ng, S., Nguyen, T. V. N., Glont, M., Hermjakob, H., & Malik-Sheriff, R. S. (2021). Reproducibility in systems biology modelling. *Molecular Systems Biology*, *17*(2), 1–7. https://doi.org/10.15252/msb.20209982

Van Rossum, G., & Drake, F. L. (2009). *Python 3 Reference Manual*. CreateSpace.

Vanlier, J., Tiemann, C. A., Hilbers, P. A. J., & van Riel, N. A. W. (2012). An integrated strategy for prediction uncertainty analysis. *Bioinformatics*, *28*(8), 1130–1135. https://doi.org/10.1093/bioinformatics/bts088

Vasilescu, D., Greene, J., Schaff, J. C., Moraru, I. I., & Blinov, M. L. (2018). Molecular Process Diagram: A precise, scalable and compact visualization of rule-based models. *BioRxiv*. https://doi.org/10.1101/503359

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S. J., Brett, M., Wilson, J., Millman, K. J., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., Larson, E., … Contributors, S. 1. . (2020). SciPy 1.0: fundamental algorithms for scientific computing in Python. *Nature Methods*, *17*(3), 261–272. https://doi.org/10.1038/s41592-019-0686-2

Voit, E. O., Martens, H. A., & Omholt, S. W. (2015). 150 Years of the Mass Action Law. *PLoS Computational Biology*, *11*(1), 1–7. https://doi.org/10.1371/journal.pcbi.1004012

Vrugt, J. A., & Ter Braak, C. J. F. (2011). DREAM(D): An adaptive Markov Chain Monte Carlo simulation algorithm to solve discrete, noncontinuous, and combinatorial posterior parameter estimation problems. *Hydrology and Earth System Sciences*, *15*(12), 3701–3713. https://doi.org/10.5194/hess-15-3701-2011

Vrugt, Jasper A. (2016). Markov chain Monte Carlo simulation using the DREAM software package: Theory, concepts, and MATLAB implementation. *Environmental Modelling and Software*, *75*, 273–316. https://doi.org/10.1016/j.envsoft.2015.08.013

Waltemath, D., Adams, R., Bergmann, F. T., Hucka, M., Kolpakov, F., Miller, A. K., Moraru, I. I., Nickerson, D., Sahle, S., Snoep, J. L., & Le Novère, N. (2011). Reproducible computational biology experiments with SED-ML - The Simulation Experiment Description Markup Language. *BMC Systems Biology*, *5*(1), 198. https://doi.org/10.1186/1752-0509-5-198

Weerts, H. H. M., Van den Hof, P. M. J., & Dankers, A. G. (2018). Identifiability of linear dynamic networks. *Automatica*, *89*, 247–258. https://doi.org/10.1016/j.automatica.2017.12.013

Westphal, D., Kluck, R. M., & Dewson, G. (2014). Building blocks of the apoptotic pore: how Bax and Bak are activated and oligomerize during apoptosis. *Cell Death & Differentiation*, *21*(2), 196–205. https://doi.org/10.1038/cdd.2013.139

Wilkinson, D. J. (2007). Bayesian methods in bioinformatics and computational systems biology. *Briefings in Bioinformatics*, *8*(2), 109–116. https://doi.org/10.1093/bib/bbm007

Xia, T., Van Hemert, J., & Dickerson, J. A. (2011). CytoModeler: A tool for bridging large-scale network analysis and dynamic quantitative modeling. *Bioinformatics*, *27*(11), 1578–1580. https://doi.org/10.1093/bioinformatics/btr150

Yang-Yen, H.-F. (2006). Mcl-1: a highly regulated cell death and survival controller. *Journal of Biomedical Science*, *13*(2), 201–204. https://doi.org/10.1007/s11373-005-9064-4

Yao, J., Pilko, A., & Wollman, R. (2016). Distinct cellular states determine calcium signaling response. *MSB*, 92093. https://doi.org/10.1101/059287

YIN, X.-M. (2000). Signal transduction mediated by Bid, a pro-death Bcl-2 family proteins, connects the death receptor and mitochondria apoptosis pathways. *Cell Research*, *10*(3), 161–167. https://doi.org/10.1038/sj.cr.7290045

Yoon, S. O., Park, D. J., Ryu, J. C., Ozer, H. G., Tep, C., Shin, Y. J., Lim, T. H., Pastorino, L., Kunwar, A. J., Walton, J. C., Nagahara, A. H., Lu, K. P., Nelson, R. J., Tuszynski, M. H., & Huang, K. (2012). JNK3 perpetuates metabolic stress induced by Aβ peptides. *Neuron*, *75*(5), 824–837. https://doi.org/10.1016/j.neuron.2012.06.024

Zhan, X., Kaoud, T. S., Kook, S., Dalby, K. N., & Gurevich, V. V. (2013). JNK3 enzyme binding to arrestin-3 differentially affects the recruitment of upstream mitogen-activated protein (MAP) kinase kinases. *Journal of Biological Chemistry*, *288*(40), 28535–28547. https://doi.org/10.1074/jbc.M113.508085

Zhan, X., Kook, S., Gurevich, E. V, & Gurevich, V. V. (2014). Arrestin-dependent activation of JNK family kinases. *Handbook of Experimental Pharmacology*, *219*, 259–280. https://doi.org/10.1007/978-3-642-41199-1_13

Zheng, L.-S., Zhang, Y.-Y., Wu, J.-W., Wu, Z., Zhang, Z.-Y., & Wang, Z.-X. (2012). A continuous spectrophotometric assay for mitogen-activated protein kinase kinases. *Analytical Biochemistry*, *421*(1), 191–197. https://doi.org/10.1016/j.ab.2011.11.018

Zhou, H.-X., Rivas, G., & Minton, A. P. (2008). Macromolecular Crowding and Confinement: Biochemical, Biophysical, and Potential Physiological Consequences. *Annual Review of Biophysics*, *37*(1), 375–397. https://doi.org/10.1146/annurev.biophys.37.032807.125817

Zhou, H. X. (2010). Rate theories for biologists. *Quarterly Reviews of Biophysics*, *43*(2), 219–293. https://doi.org/10.1017/S0033583510000120

Zhou, P., Qian, L., Kozopas, K. M., & Craig, R. W. (1997). Mcl-1, a Bcl-2 family member, delays the death of hematopoietic cells under a variety of apoptosis-inducing conditions. *Blood*, *89*(2), 630–643. http://www.ncbi.nlm.nih.gov/pubmed/9002967

Zou, H., Yang, R., Hao, J., Wang, J., Sun, C., Fesik, S. W., Wu, J. C., Tomaselli, K. J., & Armstrong, R. C. (2003). Regulation of the Apaf-1/caspase-9 apoptosome by caspase-3 and XIAP. *Journal of Biological Chemistry*, *278*(10), 8091–8098. https://doi.org/10.1074/jbc.M204783200