Identification of cell signaling pathways based on biochemical reaction kinetics repositories

Gustavo Estrela de Matos

Text presented
to
Institute of Mathematics and Statistics
of the
University of São Paulo
For
The Qualification Exam Of Master Of Science

Field of knowledge: Computer Science

Advisor: Dr. Marcelo da Silva Reis

Center of Toxins, Immune-Response and Cell Signaling (CeTICS)

Special Laboratory of Cell Cycle, Butantan Institute

During the development of this work the author received financial support from FAPESP.

São Paulo, November 23, 2020

Abstract

Cell signaling pathways are composed of a set of biochemical reactions that are associated with signal transmission within the cell and its surroundings. Traditionally, these pathways are identified through statistical analyses on results from biological assays, in which involved chemical species are quantified. However, once generally it is measured only a few time points for a fraction of the chemical species, to effectively tackle this problem it is required to design and simulate functional dynamic models. Recently, it was introduced a method to design functional models, which is based on systematic modifications of an initial model through the inclusion of biochemical reactions, which in turn were obtained from the interactome repository KEGG. Nevertheless, this method presents some shortcomings that impair the estimated model; among them are the incompleteness of the information extracted from KEGG, the absence of rate constants, the usage of sub-optimal search algorithms and an unsatisfactory overfitting penalization. In this project, we propose a new methodology for identification of cell signaling pathways, which will make use of a myriad of public interactome and biochemical reaction kinetics repositories to deal with the incompleteness of a priori information. Moreover, we will use optimal algorithms for model selection, as well as more effective cost functions for overfitting penalization. The new methodology will be tested on artificial instances and also on cell signaling pathways identification in our case study, the Y1 mouse adrenocortical tumor cell line. (AU)

Contents

Chapter 1

Introduction

Cell signaling pathways are cascades of chemical interactions that allow the communication between the cell environment and the cell itself. These pathways are also able to regulate many cell functions, including DNA replication, cell division and death. We can observe the functioning of signaling pathways as a mechanism that can conform the cell behavior with signals that come from the environment conditions in which the cell is placed. The studies of cell signaling pathways can lead to determining how cells can respond to different stimuli; for instance, with the studies of signaling pathways activated by a chemical species, one could determine how an unhealthy cell would respond to a drug containing this species.

It is possible to construct mathematical models to represent a set of chemical reactions and consequently a signaling network. One approach on the modeling of those interactions is based on the law of mass action. This law proposes that the rate of a chemical reaction is proportional to the product of reactants concentrations, i.e. we can calculate the concentration change rate of a species in an interaction by calculating the product of reactants concentrations, up to a multiplying constant. If we consider the set of interactions of a signaling pathway, we can then come up with a system of ordinary differential equations (ODEs) that can model the dynamics of the concentration of each chemical species from the pathway. Generally, these systems are complex and cumbersome, if not impossible, to be solved analytically, therefore we resort on computational tools that apply numerical methods to approximate solutions of these systems.

In this work, we are interested in computational models that can reproduce the behavior of signaling networks, comparing simulations generated by those models to experimental measures, generally based on Western blot data. The Figure 1.1 shows a set of interactions as well as parameters of a model of a signaling network. To create computational models that are able to simulate the behaviour of a signaling pathway, two main tasks need to be accomplished.

The first task one must complete to create a model is to determine a set of interactions that will be considered in the ODE system. Searching for pathway maps on the Kyoto Encyclopedia of Genes and Genomes (KEGG) [KG00] is a good start for this task. The KEGG PATHWAY Database provides manually drawn diagrams that represent signaling networks created with experimental evidences. However, it is possible that there is no pathway on KEGG that is able to correctly represent the biological experiment of interest; for those situations, it is necessary to modify the pathway by adding or removing interactions. One might reason that we should use as many interactions as we can to get a better simulation. Indeed, a model that consider more reactions is more general and might be able to reproduce different sets of experimental data. However, being more general may imply in poor or computationally infeasible models because of two reasons: first, complex models will require more time for a numerical solution computation, which may be infeasible due to limited computational resources; and second,

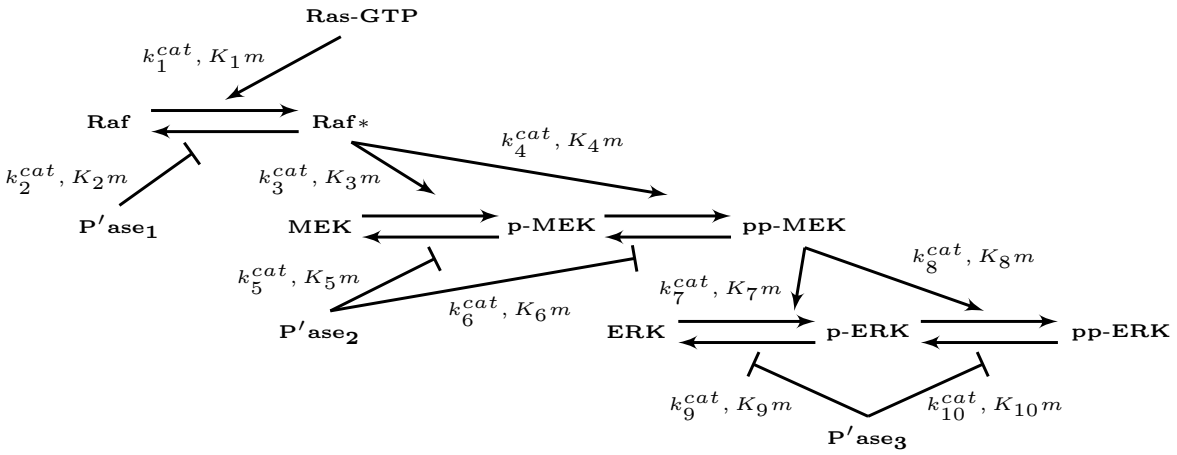


Figure 1.1: The above diagram show a hypothesis for a signaling pathway that flows through Raf-MEK-ERK cascade. Names in bold represent chemical species. Names in italic represent parameters of the ordinary differential equation of each interaction.Horizontal arrows represent phosphorylation when directed from left to right or dephosphorylation when directed in the opposite direction. Other arrows represent positive feedback if they are directed downwards or negative feedback otherwise. Original image of Marcelo S. Reis et al. (2017) [Rei+17a].

when considering many interactions, we are also considering many parameters (constants of the differential equations system), and finding appropriate values for them becomes harder as we increase the number of parameters.

The second task is to find values for all the model parameters. We can highlight two approaches for this task; one can either fetch values for these constants from the literature, or one can find values that make the model output approximate the experimental observations. For the first approach, repositories such as BioModels [LN+06] can be used; for the second approach, statistical and optimization methods are needed. For optimization, it is necessary to define a metric that can evaluate how close a set of parameters bring the simulation to the experimental observation. Only after that it is possible to search for the optimal parameter in the parameter space. Statistical inference, in the other hand, will usually try to maximize some likelihood function (find parameters that makes the data more likely to happen) on a more classical approach, while in a Bayesian approach the goal is usually to compute some posterior distribution for the model parameters (the probability of parameter values given the experimental observation).

After completing both tasks, however, as we mentioned before, we might still not have found a pair of model and parameter values that fairly approximates the biological experiment of interest. That could indicate that the set of chemical interactions chosen for the model is incomplete or has interactions that are not relevant for the biological experiment. Therefore, it is desirable to construct a systematic method of modifying the set of chemical reactions of the model in order to find a good set to represent the signaling network.

With the title "A method to modify molecular signaling networks through examination of interactome databases" [Wu15] Lulu Wu presented in her masters dissertation a methodology to systematically modify computational models of signaling networks to better simulate biological experiments. Starting with a model that does not approximate well the biological data, this methodology proposes to add to the model a set of chemical interactions that are relevant for the biochemical experiment, and consequently approximate the model simulation to experimental data. This set of interactions is a subset of interactions from a database created by Wu, joining

information from many static maps of signaling networks available on KEGG. The choice of this subset can be modeled as a combinatorial optimization problem, the feature selection problem, in which the search space is the set of all possible subsets of interactions (features) to be added. The cost function of this problem, however, is not as simple to define as the search space. Note that points from the search space do not fully define models, because there is still need to define parameter values to produce a simulation. Therefore, to analyze the quality of a model, the cost function must take into account the set of values for the model parameters. As an example, we could define the cost as the minimum distance between experimental and model measures considering all possibilities of parameter values; however, unfortunately, finding the minimizing parameter values is a hard problem.

Since this is a hard problem, the method presented by Wu implements a heuristic version of this cost function, moreover, the algorithm used to traverse the search space is also a heuristics. The cost function heuristics is based on a Simulated Annealing procedure that searches for a set of parameter values trying to minimize (as much as possible) the distance between model and experimental measures. The best found distance times -1 is then considered as the cost of the model. The size of the search space is the number of all possible subsets of interactions to be added, and this number grows exponentially on the number of interactions from the database. That explains the need of a heuristic to traverse the search space. This heuristic is based on the greedy algorithm called Sequential Forward Selection (SFS) [Whi71]. The heuristic implemented by Wu selects a fixed number of interactions from the database and then creates candidate models by adding to the current solution the respective interaction; then, after evaluating the cost function for each model, the algorithm moves to the best candidate.

The results presented on Wu's dissertation show that the method is useful when there are only a few differences between the starting model and a model that closely approximates the biological experiment. This limitation could be explained by the intrinsic difficulty of the problem, which demands fitting complex models with few experimental data; however, we would like to highlight three aspects of the work that contributes to its limitations. The first aspect is that the constructed database could be more nearly complete, adding information from other interactome databases, such as STRING [Szk+10], and also by adding information about model parameters, i.e. chemical reaction constants, that are available in other databases, e.g. the SABIO-RK [Wit+11] database. Second, the search algorithm used to modify the models can only add interactions, therefore, if the algorithm starts with (or add along the search) a spurious interaction on the topology, then the algorithm will not be able to "regret" that interaction even though there might be similar solutions without it with better fit. Third, the cost function does not include a proper penalization of complex models; the used penalization is based on a execution time limit on the simulated annealing procedure, implying on a random penalization for more complex models, which typically demand more execution time. Without a proper penalization, the algorithm is doomed to select overly complex models that, even with a good fit to the experimental data, are not likely to reproduce the same experiment conduced with any kind of perturbation to the biological environment or to the data collected.

We propose on this project to create and test a new method for modifying models of signaling networks, based on the work of Wu, and including a possible solution to the third aspect mentioned on the last paragraph. Although the first and second aspects have potential of improvements, we have not prioritized them and, because of the complexity of the third aspect, we decided that including the first and second aspect would excessively enlarge the scope of this work. Therefore, we limit our work to a few known cell signaling pathways, and we only include simple algorithms for traversing the search space. To the last aspect, which considers the cost function, as our major concern in this work, we intend to use Bayesian approaches to

rank models [VG07] based on the likelihood of them to reproduce the observed data; if we say M is a model with parameter space Θ and D is a set of observations, then we would like to estimate

$$p(\boldsymbol{D}|M) = \int_{\boldsymbol{\theta}\in\Theta} p(\boldsymbol{D}|\boldsymbol{\theta}, M)p(\boldsymbol{\theta}|M)d\boldsymbol{\theta},$$

where $p(\boldsymbol{D}|\boldsymbol{\theta}, M)$ is the likelihood of data $\boldsymbol{D}$, given that the model M with parameters $\boldsymbol{\theta}$ are "correct", which is the same as stating that this model and parameters determine the behaviour of the cell; $p(\boldsymbol{\theta}|M)$ is the prior probability of $\boldsymbol{\theta}$; and finally, $p(\boldsymbol{D}|M)$ is the probability of the data being generated by model M. This cost function has as an advantage the fact that models are not ranked using a single value for parameters, instead, the cost considers all possibilities of parameters, integrating over the parameter space. Another advantage of this cost function is that, since it is based on likelihoods of the model to reproduce data, overly complex models are automatically penalized.

1.1 Objectives

In this work we propose to implement a methodology that allows us to solve the problem of identification of cell signaling pathways as a feature selection problem, using a Bayesian approach for the cost function. This cost function should be able to rank models according to the likelihood of the experimental data being generated by them. After the definition and construction of such cost function, we intend to build small examples of model selection of cell signaling pathways, and run simple searches on their search spaces, using our chosen cost function. Finally, these runs should allow us to get a glance on the surface that the chosen cost function induces over the search space of the model selection problem. To achieves these goals, we should accomplish the following tasks

1. **Study state of the art Bayesian algorithms for signaling network model selection**. We have chosen two methodologies to study: the first, a work of Liepe et al. [Lie+14], uses an Approximate Bayesian Computation approach for ranking models; the second, a work of Xu et al. [Xu+10], uses an estimative of the marginal likelihood $p(\boldsymbol{D}|M)$ to rank models.

2. **Implementation and testing of cost functions**. We propose to compare both the cost functions used by Liepe and Xu. To test the cost function used by Liepe, we should use the software ABC-SMC, and to test the cost function used by Xu, we should implement our own software, SigNetMS. After the preparation of instances, we should test the performance of both software.

3. **Preparation of instances of the identification of cell signaling network problem.** After comparing both cost functions, we intend to construct instances to test our feature selection approach on identification of cell signaling network. These instances must be composed by a base model, a small database of candidate reactions, and a set of measurements to which our candidate models should fit.

4. **Traversal of the search space.** We propose to perform walks and runs of simple search algorithms over the search space of candidate models for the instances we created. These runs will provide us information about the surface that the used cost function induces over the search space. This type of information shall be useful for defining new search algorithms for the identification of cell signaling pathways.

1.2 Organization

TODO

- *Chapter 2 (Fundamental Concepts):* we will give more details about the identification of signaling pathway problem. We will show how to obtain experimental data for signaling pathways and how can chain of chemical reactions can be modeled as a system of differential equations. We close the chapter presenting briefly state of the art methods of model ranking and also the Metropolis-Hastings algorithm, which is a useful tool for methods of model ranking.
- *Chapter 3 (Model Selection Methods):* we will present two methodologies that are the state of the art in model selection. Both methods are Bayesian approaches, where the first is based on the estimation of marginal likelihoods and the second is based on Approximate Bayesian Computation.

Chapter 2

Fundamental Concepts

In this section we provide the concepts that are fundamental to understand the biological and computational problems, methodologies and results that we will present in this work. We start this chapter presenting what is a cell signaling pathway and how can one take measures to identify its activity on the cell. Later we present how it is possible to represent chemical iteractions as differential equations, and how that allows one to model a cell signaling pathway with a system of ordinary differential equations. Then, we present more formally the problem we are trying to solve on this project, the identification of cell signaling pathways, as well as the state of the art methods of model ranking. Finally, we present the basics of posterior distribution sampling, which is a useful tool when working with Bayesian approaches, such as the ones used on this project to rank models.

2.1 Cell Signaling Pathways

Cell signaling pathways are part of the complex cell communication system, and it allows the cell to perceive the conditions of the environment in which it is placed and change its behaviour accordingly. Signaling pathways participate in the regulation of many cell functions, including development, division and cell death [Han17]. The dynamic of a signaling pathway can also be releated to diseases, as in many cases of cancer.

The signal perceived by a cell can come from cells that are close (including the same cell that produced the signal), as in synapses, or it can travel long distances in the organism, as in hormones. When a signal reaches a cell, it binds to a specific receptor in the membrane, and once that happens, the receptor can trigger a sequence of chemical interactions that can include change of conformation of proteins, activation or inactivation of proteins, and change of concentration of chemical species in the cell. Ultimately, this chain of chemical reactions caused by the signal can alter the behaviour of the cell, what is called signal transduction.

Since signaling pathways participate in many of the cell functions, and are also related to diseases, it is important to study those structures in order to get a better understanding of the cell mechanisms and diseases. One approach on the study of the cell signaling pathways is to measure the concentration change of proteins and how they interact to produce those changes.

2.2 Measurements of Proteins in Cell Signaling Pathways

Western blot [TSG79] is a laboratory technique that can indicate the amount of a specific protein that is present in a mixture. This technique shows the presence of a protein in a mixture by "blotting" a membrane where the molecules of interest are located. We can summarize the procedure in the following steps: first a mixture containing a sample of cells of interest must be created; second, proteins from the mixture should be fixed on the blotting membrane; third, an antibody should bind to the target protein molecules; and finally, a method for highlighting the bound antibody should be applied. An image of the resulting membrane can then be analyzed with computer programs to quantify the relative concentration (with respect to some other protein, usually a control protein that has fairly the same concentration during the whole experiment) of the protein of interest.

By repeating this procedure in different times it is possible to create time-course observations of proteins throughout the biological experiment. With this tool, a researcher can choose a set of relevant proteins from a signaling network and gain knowledge about the dynamics of such chemical species during the experiment. For instance, in a signaling network experiment in which it is desired to understand how the change of concentrations of a species at the beginning of the pathway changes the concentration of some species at the end of the cascade, then measurements of both are relevant to understand the biological experiment. Figure 2.1 presents an example of time-course Western blot for an experiment where it is desirable to understand how extracellular signal-regulated kinase (ERK) is activated (phosphorylated) as a function of levels of Rat sarcoma bound to guanosine triphosphate (Ras-GTP).

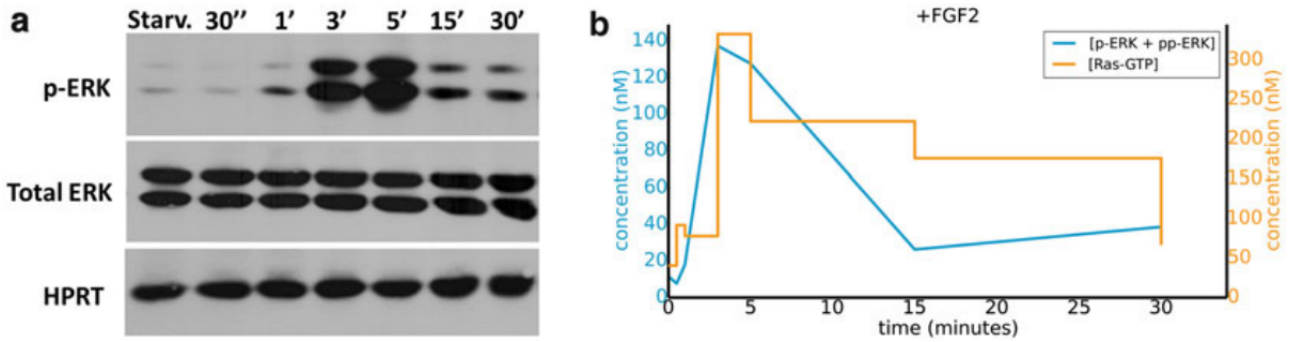


Figure 2.1: Figure **a** *shows time-course measurements of ERK, phosphorylated ERK (p-ERK) and hypoxanthine-guanine phosphoribosyltransferase (HPRT). HPRT is a "loading" protein, that means that its concentration is fairly the same through the experiment, and therefore it is used as a normalizing factor to total ERK concentration. Figure* **b** *shows values of phosphorylated ERK that are obtained after processing Figure* **a**. *Original image of Marcelo S. Reis et al. (2017) [Rei+17a].*

These measurements alone do not always provide means for researchers to understand a cell signaling pathway experiment. However, if we create a computational model for this signaling networks that is able to reproduce experimental data, then we might use this model as a summary of the signaling network, which can provide to researchers evidences of the biological phenomena.

2.3 Dynamic Modeling of Cell Signaling Pathways

One approach onto modeling cell signaling pathways is to model the dynamics of the concentrations of chemical species involved. This can be accomplished when using the law of mass action [VV10]. This law states that, in an elementary reaction, the speed (or rate) of a chemical reaction is proportional to the product of the concentration of all reactants. An elementary reaction is a reaction in which there is no participation or need of an intermediate reaction to describe the first in a molecular level. In practice, it is more common to see two types of elementary reactions, they are first or second order reactions.

2.3.1 Modeling Elementary Reaction Rates

A first order reaction is composed of one reactant only. Suppose A is the only reactant and B is the only product of a reaction, then we can write this reaction as:

$$\mathrm{A} \longrightarrow \mathrm{B}.$$

The reaction rate of this reaction, according to the law of mass action, is

$$k_1[\mathrm{A}],$$

where k_1 is some constant and [A] is the concentration of A. It is import to note that the constant k_1 is a rate coefficient of the reaction and, therefore, it can only assume positive values.

A second order reaction is composed of two reactants. Suppose C and D are both and the only reactants and F is the product of a reaction, then we can write this reaction as:

$$\mathrm{C} + \mathrm{D} \longrightarrow \mathrm{F}.$$

The reaction rate of this reaction is

$$k_2[\mathrm{C}][\mathrm{D}],$$

where k_2 is a (positive) constant and [C] and [D] are the concentrations of C and D, respectively.

Using these two laws to calculate the speed of reactions, we are able to describe how the concentration of chemical species in a system change through time using differential equations. To illustrate this and future concepts of this section, we are going to consider a minimal system composed of a simple enzymatic reaction:

$$\mathrm{E} + \mathrm{S} \underset{k_r}{\overset{k_f}{\rightleftharpoons}} \mathrm{ES} \xrightarrow{k_{cat}} \mathrm{E} + \mathrm{P}, \tag{2.1}$$

where E is an enzyme, S is a substrate, ES is the enzyme-substrate complex, and P is the product.

Each arrow in Equation 2.1 represents one elementary reaction, and the names over or under arrows represent reaction rate constants. All three reactions can be represented by the equations:

$$\mathrm{E} + \mathrm{S} \xrightarrow{k_f} \mathrm{ES} \tag{2.2a}$$

$$\mathrm{ES} \xrightarrow{k_r} \mathrm{E} + \mathrm{S} \tag{2.2b}$$

$$\mathrm{ES} \xrightarrow{k_{cat}} \mathrm{E} + \mathrm{P} \tag{2.2c}$$

and they have, respectively, reaction rates of:

$$k_f[\mathrm{E}][\mathrm{S}]$$
$$k_r[\mathrm{ES}]$$
$$k_{cat}[\mathrm{ES}].$$

Now, to determine a model of the concentration dynamics for Reaction 2.1, we will write a system of ordinary differential equations. To do so, we should take every chemical species and calculate its concentration change rate based on the rate of each reaction that it participates. For instance, the enzyme E is a reactant on Reaction 2.2a and is also a product on reactions 2.2b and 2.2c, then we consider that E changes its concentration over time (t) according to the differential equation:

$$\frac{d[\mathrm{E}]}{dt} = -k_f[\mathrm{E}][\mathrm{S}] + (k_r + k_{cat})[\mathrm{ES}] \tag{2.3}$$

Note that we are adding reation rates in which the species is a product and we are subtracting reaction rates in which the species is a reactant. Repeating this procedure for every other species of the enzymatic reaction leads to the following system of ordinary differential equations:

$$\frac{d[\mathrm{E}]}{dt} = -k_f[\mathrm{E}][\mathrm{S}] + (k_r + k_{cat})[\mathrm{ES}] \tag{2.4a}$$
$$\frac{d[\mathrm{S}]}{dt} = -k_f[\mathrm{E}][\mathrm{S}] + k_r[\mathrm{ES}] \tag{2.4b}$$
$$\frac{d[\mathrm{ES}]}{dt} = k_f[\mathrm{E}][\mathrm{S}] - (k_r + k_{cat})[\mathrm{ES}] \tag{2.4c}$$
$$\frac{d[\mathrm{P}]}{dt} = k_{cat}[\mathrm{ES}]. \tag{2.4d}$$

2.3.2 Simplification of Dynamic Models

The System 2.4 can be simplified if we apply properties of enzymatic reactions together with algebraic simplifications. We will show then how to derive the quasi-steady-state Michaelis-Menten model for enzymatic reactions. With the correct assumptions, this model is able to reproduce the behaviour of an enzymatic reaction without considering the intermediate enzyme-substrate complex, leading to simpler models.

A basic principle we need to apply to our system in order to derive the Michaelis-Menten model is the principle of mass conservation. This principle is valid if we assume that the reactions 2.1 are isolated, meaning that the chemicals on these reactions are not involved in other reactions at the same time. Applying this principle to the enzyme chemical, produces the following equation:

$$[\mathrm{E}_0] = [\mathrm{E}] + [\mathrm{ES}].$$

If we apply this equation to the derivative of the concentration of ES, we will get the following equation:

$$\frac{d[\mathrm{ES}]}{dt} = k_f([\mathrm{E}_0] - [\mathrm{ES}])[\mathrm{S}] - (k_r + k_{cat})[\mathrm{ES}]. \tag{2.5}$$

One more assumption is necessary to derive the simplification. This assumption states that the concentration of substrate-enzyme complex does not change over time, i.e. $\frac{d[\mathrm{ES}]}{dt} = 0$, and it was first proposed in 1925 by Briggs and Haldane [BH25]. Generally, this assumption is

applicable whenever [S] ≫ [E]. If we apply this assumption together with the mass conservation assumption on the Equation 2.5, we get:

$$[\mathrm{ES}](k_r + k_{cat}) = k_f([\mathrm{E_0}] - [\mathrm{ES}])[\mathrm{S}],$$
$$[\mathrm{ES}] = \frac{[\mathrm{E}]_0[\mathrm{S}]}{K_m + [\mathrm{S}]},$$

in which $K_m = \frac{k_{cat}+k_r}{k_f}$ is known as Michaelis constant. Considering this, we can rewrite the rate of [P] as:

$$\frac{d[\mathrm{P}]}{dt} = k_{cat}\frac{[\mathrm{E}]_0[\mathrm{S}]}{K_m + [\mathrm{S}]}. \tag{2.6}$$

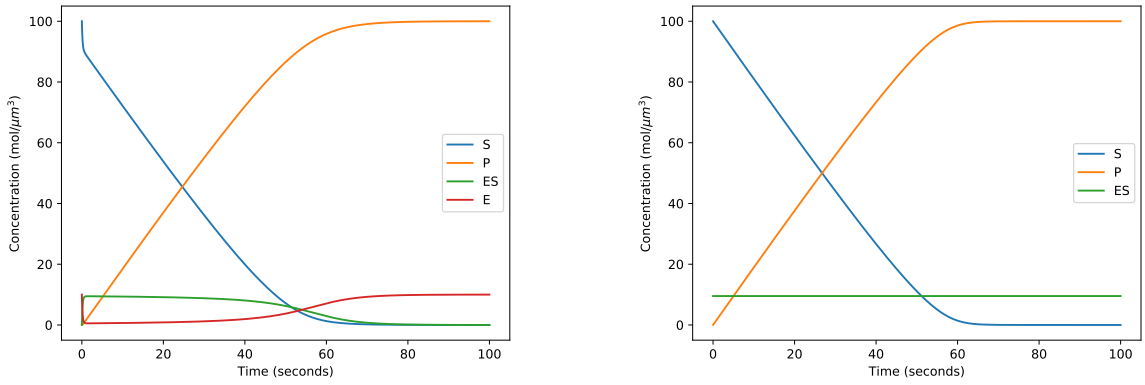
And finally, if we apply mass conservation to the substrate, we will get the following equation:

$$[\mathrm{S_0}] = [\mathrm{S}] + [\mathrm{ES}] + [\mathrm{P}],$$

then, we can differentiate this equation on t and use the quasi-steady-state assumption ($\frac{d[\mathrm{ES}]}{dt} = 0$) to obtain:

$$\frac{d[\mathrm{S}]}{dt} = -\frac{d[\mathrm{P}]}{dt}. \tag{2.7}$$

Now, with equations 2.6 and 2.7 we are able to reproduce the dynamics of the substrate and product of the enzymatic reaction. Therefore, using the Michaelis-Menten model, we could simplify the System2.4 that had four equations and three parameters to a new model that has only two equations and two parameters (k_{cat} and K_m). Figure 2.2 shows a comparison between the dynamics of the complete and Michaelis-Menten models of enzymatic reactions.



(a) (b)

Figure 2.2: An example of the dynamics produced by two models of enzymatic reactions. The Figure 2.2(a) presents the dynamics of the model 2.4 and Figure 2.2(b) presents the dynamics of the Michaelis-Menten simplification to the same model. For this simulations, it is necessary to define initial concentrations of the chemical species involved, and it is used: 10 molecules/$(\mu m)^3$ for the enzyme (E); 100 molecules/$(\mu m)^3$ is used for the substrate (S); and 0 molecules/$(\mu m)^3$ is used for the other species. In addition to this, it is also necessary to define model parameter values, and it is used: 0.06 $(\mu m)^3(molecules*s)^{-1}$ for k_f; 0.1 (s^{-1}) for k_r; 0.2 (s^{-1}) for k_{cat}; and, following the Michalis-Menten model, 5 molecules/$(\mu m)^3$ is used for K_m.

2.4 Identification of Cell Signaling Pathways

Identification of signaling pathways is the problem of finding the components of a signaling pathway and how they interact in order to reproduce a cell behaviour that has been previously

measured experimentally. The input of this problem is usually a description of the biological experiment, containing previous information about the signaling pathway, such as known reactions and parameters, and a set of measurements, commonly Western blot data, which is the only type of measurement we consider in this work. The output to this problem is then composed of a set of interactions that are actively controlling the behaviour of interest of the cell, and also the set of parameter values that should be used on these interactions to create a model that approximates the experimental observations; it is possible to output a single value for each parameter, as it was presented in [Wu15], or output information about these values, using a posterior (to the experimental data) distribution, as it was presented in [Lie+14] and [Xu+10].

Two main tasks must be completed to produce such output. The first task is to find candidate topologies for the pathway model, i.e. different set of interactions that are relevant to the pathway of interest. The second task is to rank those models according to their ability to simulate the pathway, approximating the experimental measurements.

The second task is also known as the model selection problem and even though it is a broad area, there are works on the literature that treat specifically the problem in a biochemical context. Solutions to this problem should be able to choose model candidates according to their ability to reproduce observed data, penalizing overly complex models to avoid overfitting.

One approach on setting the score of a model is to search for the set of parameter values that make the model measurements the closest to the experimental data, and then define a distance between these two measurements; then it is possible to use this distance, plus a penalty for complexity, to create a ranking of the models. Note that a penalty function is needed because complex and more general models can overfit, producing models that cannot represent the same signaling network with small perturbations to biological experiment. We can write this scoring function as:

$$score_1(M) = -min_{\{\boldsymbol{\theta} \in \Theta' \subset \Theta\}} dist(\phi(M, \boldsymbol{\theta}), \boldsymbol{D}) + R(M),$$

where M is the model, Θ is the parameter space for model M, Θ' is the subset of the parameter space where the search for the best parameter values was conduced, $\boldsymbol{D}$ is a list of experiment measurement (each measurement is a time-course observation of the biological phenomena of interest), ϕ is a function that determines the simulated measurement on the model, and R is a regularization function that penalizes model complexity. This approach was implemented on the work of Lulu Wu [Wu15], using a Simulated Annealing procedure to search for parameter values that minimize the distance between simulation and experiment. The Simulated Annealing procedure used is available on the software SigNetSim (Signaling Network Simulator), which is based on the of the work of Chu on [CDR99]. However, Wu's methodology showed limitations when testing the ability to reconstruct models from experiments, and this could be related to the used penalization term. In fact, choosing a good regularization function is crucial to the performance of this methodology.

Another approach to setting the model score is to consider the model parameters as random variables and then marginalize the probability of the model and parameters to reproduce the observed data, i.e. estimate (because calculating is usually hard) the marginal probability:

$$p(\boldsymbol{D}|M) = \int_{\Theta} p(\boldsymbol{D}|M, \boldsymbol{\theta}) p(\boldsymbol{\theta}|M) d\boldsymbol{\theta}, \tag{2.8}$$

where $\boldsymbol{D}$ is the observed data, M is the model and Θ is the parameter space for model M. The function $p(\boldsymbol{\theta}|M)$ is the prior probability function of the parameter $\boldsymbol{\theta}$ on model M. The function $p(\boldsymbol{D}|M, \boldsymbol{\theta})$ is the likelihood of observing the data $\boldsymbol{D}$ when simulating the model M using parameters θ. Sometimes, however, the likelihood function is unknown or computationally intractable; for these cases, it is possible to use an alternative Bayesian approach, called

Aproximate Bayesian Computation (ABC), to estimate the probability $p(M|\boldsymbol{D})$ and use it as a ranking score [Ton+09].

For the first task of identification of signaling pathways we described (the creation of model candidates), there are not as many works on the literature as there are for the second task. Commonly, researchers must resort on their own knowledge on the biological experiment and consult interactome maps available on repositories such as KEGG and BioModels to construct manually the hypothesis of models for a signaling pathway. That enlightens the importance to create a methodology that systematically creates candidate models of signaling networks, as we propose on this project.

2.5 State of the Art in Selection of Biochemical Models

The state of the art in biochemical model selection is based on Bayesian inference. The Bayesian approaches provide the benefits of ranking models with statistical formalism, and automatically penalizes overly complex models. More than that, through the prior distribution of the parameters, the researcher is able to input prior knowledge about interactions constants; this type of information can facilitate the parameter inference of models since it tends to concentrate the search.

Bayesian approaches consider that model parameters are random variables, instead of fixed unknown constants. We can argue that this modeling is fair to the reality in biochemical processes because interactions constants can vary depending on the cell conditions. Therefore, in a biological experiment in which there might be perturbations to the cell environment, it should be more adequate to rank models integrating the models score over a probability space of parameters instead of fitting the model to data using a single point of the parameter space. We will now present the basic concept of two methods that use this idea for model selection, the Annealing-Melting Integration (AMI) [VG07] and Approximate Bayesian Computation (ABC) [Ton+09].

The Annealing-Melting Integration is a method that estimates the integral 2.8 using concepts of thermodynamics. With thermodynamic integration, it is possible to write the logarithm of this integral as:

$$\ln p(\boldsymbol{D}|M) = \int_0^1 \mathbb{E}_{q_\beta(\boldsymbol{\theta})}[\ln p(\boldsymbol{D}|M,\theta)]d\beta \tag{2.9}$$

where $p(\boldsymbol{D}|M,\theta)$ is the likelihood function (the probability of observing the data $\boldsymbol{D}$ when M is the correct model, with parameters $\boldsymbol{\theta}$); and $\mathbb{E}_{q_\beta(\boldsymbol{\theta})}$ is an expectation taken over the probability space of $q_\beta(\boldsymbol{\theta}) \propto p(\boldsymbol{D}|M,\boldsymbol{\theta})^\beta p(\boldsymbol{\theta}|M)$. The variable β works in the integral as a temperature term, determining the probability functions $q_\beta(\boldsymbol{\theta})$, $\beta \in [0,1]$; note that when $\beta = 0$ then

$$q_0(\boldsymbol{\theta}) = p(\boldsymbol{\theta}|M),$$

the prior distribution of the parameters; also, and when $\beta = 1$ then

$$q_1(\theta) = \frac{p(\boldsymbol{D},\boldsymbol{\theta}|M)}{\int_\Theta p(\boldsymbol{D},\boldsymbol{\theta}|M)} = \frac{p(\boldsymbol{\theta}|\boldsymbol{D},M)p(\boldsymbol{D}|M)}{p(\boldsymbol{D}|M)} = p(\boldsymbol{\theta}|\boldsymbol{D},M),$$

the posterior distribution of parameters. Therefore, the integral 3.6 takes the expected value of the likelihood function of $\boldsymbol{D}$ over a sequence of probability distributions that is a "bridge of distributions" connecting the prior and posterior distributions of parameters. Calculating this integral is usually infeasible, hence in practice it is needed to estimate this integral using samples of a finite number of tempered distributions, $q_\beta(\boldsymbol{\theta})$.

As we mentioned before, the likelihood function $p(\boldsymbol{D}|M, \boldsymbol{\theta})$ may be very hard to calculate if not impossible. For those cases it is possible to use a parameter inference approach that is likelihood-free, called Approximate Bayesian Computation (ABC). This method has the goal of producing a sample of parameters that brings the model simulations close to the observed data. A generic ABC algorithm starts proposing a candidate parameter $\boldsymbol{\theta}^*$ from a proposal distribution; then, a simulation, $\phi(M, \boldsymbol{\theta}^*)$ of the model using the candidate parameters is produced; if, for some distance function d, it is true that $d(\phi(M, \boldsymbol{\theta}^*), \boldsymbol{D}) < \epsilon$, then we accept $\boldsymbol{\theta}^*$ as part of the sample. If ϵ is sufficiently small, then the produced samples approximates well the posterior distribution. To use ABC methods for model selection it is enough to add a model indicator parameter to the parameter array, then it is possible to extract model distribution from the accepted parameters.

2.6 Metropolis-Hastings to Generate Samples

Both methods for model selection, based on ABC or using thermodynamics integration, need to generate samples of probability distributions. Generating samples of distributions is simple for many well known distributions, however, for some other distributions this task may not be as simple. Metropolis-Hastings algorithms are capable of generating a sample that has some probability distribution p, which is called the target distribution. In fact, the method can be used even when it is not possible to access the target function directly; for those cases, a distribution that is proportional to the target is necessary.

Being able to generate a sample of a target distribution without accessing the probability function itself is useful for our applications in Bayesian model ranking. Consider that we need to create a sample of the parameters posterior distribution $p(\boldsymbol{\theta}|M, \boldsymbol{D})$. Calculating this probability function is very hard because

$$p(\boldsymbol{\theta}|M, \boldsymbol{D}) = \frac{p(\boldsymbol{D}|\boldsymbol{\theta}, M)p(\boldsymbol{\theta}|M)}{p(\boldsymbol{D}|M)},$$

and this equation has the term $p(\boldsymbol{D}|M)$, which is only known (through some estimation) at the end of the model ranking. However, if we can access the likelihood function $p(\boldsymbol{D}|\boldsymbol{\theta}, M)$ and the prior $p(\boldsymbol{\theta}|M)$, then we can calculate the product of these two, which is proportional to the posterior distribution (since $p(\boldsymbol{D}|M)$ is only a constant because it does not depend on $\boldsymbol{\theta}$). Therefore, using the likelihood and the prior distributions together with a Metropolis-Hastings algorithm provides a mean to generate a sample of the posterior.

A generic Metropolis-Hastings algorithm that creates a sample of a target distribution $p(\lambda)$ proceeds as follows:

1. Choose some starting point λ_0 for which $p(\lambda_0)$ is not zero. Also set $t = 1$.

2. Sample a candidate point λ^* from a proposal (or jumping) distribution with probability $J_t(\lambda^*|\lambda^{t-1})$.

3. Calculate the ratio:
$$r = \frac{p(\lambda^*)J_t(\lambda^{t-1}|\lambda^*)}{p(\lambda^{t-1})J_t(\lambda^*|\lambda^{t-1})} \tag{2.10}$$

4. With probability $min(1, r)$ set $\theta^t = \theta^*$ and set $\theta^t = \theta^{t-1}$ otherwise.

5. Increase t by one and, if not reached limit number of iterations, go back to step 2.

Note that if the target $p(\lambda)$ is not available, and rather another function $q(\lambda) = \frac{1}{c}p(\lambda)$ is available, then the ratio 2.10 can be calculated as:

$$r = \frac{p(\lambda^*)J_t(\lambda^{t-1}|\lambda^*)}{p(\lambda^{t-1})J_t(\lambda^*|\lambda^{t-1})} = \frac{(q(\lambda^*)c)J_t(\lambda^{t-1}|\lambda^*)}{(q(\lambda^{t-1})c)J_t(\lambda^*|\lambda^{t-1})} = \frac{q(\lambda^*)J_t(\lambda^{t-1}|\lambda^*)}{q(\lambda^{t-1})J_t(\lambda^*|\lambda^{t-1})}$$

More than that, if the proposal distribution is symmetric ($J_t(\lambda_a|\lambda_b) = J_t(\lambda_b|\lambda_a)$ for all λ_a, λ_b and t), the produced algorithm is called Metropolis algorithm and has the ratio $r = p(\lambda^*)/p(\lambda^{t-1})$.

Different implementations of the Metropolis-Hastings algorithm are possible. The possible changes include the choice of starting point, the choice of proposal distributions and number of iterations. As an example, some algorithms are adaptive in the sense that they can change the proposal distribution according to the acceptance rate of proposed points [Gel+13]. These types of algorithms are handy when we are not sure of how spread the proposal distribution should be: with larger variance, we are likely to stay in a local minima for a longer time, rejecting most of the proposed jumps; in the other hand, if the proposal distribution has small variance, then the algorithm needs more time to explore the parameter space.

Chapter 3

Model Selection Methods

In this chapter we will define the model selection problem in the context of this work and we will also present two state of the art methodologies that can be used to rank models, both of them are Bayesian approaches. These methods create estimatives of $p(\boldsymbol{D}|M)$ or $p(M|\boldsymbol{D})$ and use these values as model scores.

The first approach is to estimate the marginal likelihood of the data $\boldsymbol{D}$ being reproduced by a model M, $p(\boldsymbol{D}|M)$. To use this method, it is necessary to define a likelihood function, $p(\boldsymbol{D}|M, \boldsymbol{\theta})$, and prior distributions of model parameters $p(\boldsymbol{\theta}|M)$. The estimation of this probability is done by taking samples of tempered posteriors (to experimental data) of model parameters. These tempered posterior distributions construct a bridge of distributions connecting the prior and posterior parameter distributions, allowing an estimation of the marginal likelihood.

Another method is to use Approximate Bayesian Computation. ABC methods are used to estimate posterior (also to experimental data) distributions of parameters. On this method, a sequence of populations of parameters is created, and for each generation, the distance between experimental data and data simulated by the population is decreased, leading to a population that tends to be closer to a posterior distribution. In our case, we add to $\boldsymbol{\theta}$ a model indicator parameter M, and then we create a sample of $p(\boldsymbol{\theta}, M|\boldsymbol{D})$. If we marginalize this sample, we can calculate an estimative of the probability $p(M|\boldsymbol{D})$. ABC methods allow simpler algorithms, and they also allow selecting models without the neeed of a likelihood function, altough defining the prior distribution of parameters is still needed.

3.1 Elements of model selection

The model selection problem we consider in this work consists in the selection of a biochemical model that can best simulate the dynamics of concentrations of chemical species, measured previously on a biological experiment. The models we consider are mathematically represented by a system of ordinary differential equations that contains parameters related to reaction rates and experimental errors. We can list three main entities of this problem: the set of models, the parameter space of each model, and the biological experimental data. We shall then define how we represent these entities through this text.

We indicate a model with a random variable that assume natural values, $M : \Omega \to \mathbb{N}$. We consider that a model is not only bound to a set of chemical reactions and its mathematical representation, but it is also bound to initial concentrations of chemical species and a prior distribution of model parameters, which we represent as $p(\boldsymbol{\theta}|M)$.

Since we are using a Bayesian approach, we consider that model parameters is a random

vector $\boldsymbol{\theta}|M : \Omega \to \mathbb{R}^n_+$, where d equals to the number of parameters in M plus one, so we can also represent one parameter related to experimental error. This parameter represents the standard deviation of errors in the observation of the experiment. It is important to note that all parameters are positive, since reaction rates and standard deviations can only be positive. That is an important information to choose a prior distribution for model parameters.

The experimental data is a vector of values that describe the dynamics of some measurement made on the biological phenomena of interest. We represent data as $\boldsymbol{D} \in \mathbb{R}^d$. As we described in Section 2.2, these measurements are often created with Western Blot experiments, and the values of $\boldsymbol{D}$ usually represent the ratio of concentrations of proteins.

3.2 Model ranking using Marginal Likelihood

The marginal likelihood of an experiment measurement $\boldsymbol{D}$ being reproduced by a model M, $p(\boldsymbol{D}|M)$, can be used as a model ranking metric as it determines which model makes the experimental observations more likely to happen. Before defining how to calculate the marginal likelihood, we must define some likelihood function. To understand our chosen likelihood function $p(\boldsymbol{D}|M, \boldsymbol{\theta})$, we must understand that conditioning the observation to a model and a set of parameters means that, in the probability space from which $\boldsymbol{D}$ is taken, the model M is the "real" model and it has the parameter values of $\boldsymbol{\theta}$; i.e. the model M with parameters $\boldsymbol{\theta}$ controls the behaviour of the system from which $\boldsymbol{D}$ was observed. Now, after understanding in which probability space we are, we can assume that:

$$p(\boldsymbol{D}|M, \boldsymbol{\theta}) = p_{\mathcal{N}_{(\vec{0},\Sigma)}}(\phi(M, \boldsymbol{\theta}) - \boldsymbol{D}), \tag{3.1}$$

where $\phi(M, \boldsymbol{\theta}) \in \mathbb{R}^d$ is the experimental measurement on the simulation generated by the model M with parameters $\boldsymbol{\theta}$, and $p_{\mathcal{N}_{(\vec{0},\Sigma)}}(\cdot)$ is the probability density function of a Multivariate Normal random variable with mean $\vec{0}$ and covariance matrix Σ. Note that we are considering that simulations of a model are deterministic, as ϕ is function, and therefore the randomness of the model arises from observation experimental error. We consider this random error follows a Gaussian distribution, similarly to Xu and Vyshemirsky [Xu+10][VG07].

As it is done in the work of Xu et al. (2010), we can consider that the observation error is independent for each time step [Xu+10], therefore we can simplify 3.1 to:

$$p(\boldsymbol{D}|M, \boldsymbol{\theta}) = \prod_{i=1}^{d} p_{\mathcal{N}_{(0,\sigma^2)}}(\phi_i(M, \boldsymbol{\theta}) - \boldsymbol{D}_i). \tag{3.2}$$

The σ^2 used in Equation 3.2 is also a parameter of the model, and as we explained on the last section, this parameter is incorporated into $\boldsymbol{\theta}$, which means that, for some $k \in \{1, ..., n\}$, $\boldsymbol{\theta}_k = \sigma^2$.

Now that we defined the likelihood function, we can write the marginal likelihood. As the name suggests, this value comes from the marginalization of the likelihood function, over the parameter space. We can write as:

$$p(\boldsymbol{D}|M) = \int_{\Theta} p(\boldsymbol{D}|M, \boldsymbol{\theta}) p(\boldsymbol{\theta}|M) d\boldsymbol{\theta}. \tag{3.3}$$

However, calculating this integral analytically is only possible in very special cases and, usually, it would depend on knowing models for the distributions associated to these probability functions, which is generally not possible in our case.

Even though this integral is very hard to be calculated, there are methods that allow us to estimate its value. A straight forward method to estimate this integral value is using Importance Sampling Estimators [NR93]. This method uses the Monte Carlo integral estimation method that can estimate integrals of the form $\int g(\lambda)p(\lambda)d\lambda$ using the estimator:

$$\hat{I} = \sum_{i=1}^{m} w_i g(\lambda_i) / \sum_{i=1}^{m} w_i,$$

where $w_i = p(\lambda)/p^*(\lambda)$, and $p^*(\cdot)$ is known as the importance sampling function. If we set $\lambda = \boldsymbol{\theta}|M$ and use the prior ($p(\boldsymbol{\theta}|M)$) or the posterior ($p(\boldsymbol{\theta}|M, \boldsymbol{D})$) as importance sampling functions, then we would get respectively the estimators:

$$\frac{1}{m}\sum_{i=1}^{m} p(\boldsymbol{D}|M, \boldsymbol{\theta}^{(i)}) \quad (\text{with } \boldsymbol{\theta}^{(i)} \sim p(\boldsymbol{\theta}|M)),$$

$$\left(\frac{1}{m}\sum_{i=1}^{m} p(\boldsymbol{D}|M, \boldsymbol{\theta}^{(i)})^{-1}\right)^{-1} \quad (\text{with } \boldsymbol{\theta}^{(i)} \sim p(\boldsymbol{\theta}|M, \boldsymbol{D})).$$

However, as showed by Vyshemirsky et al. (2007), these estimators might produce very large variances and may not perform well for biochemical model selection applications. Hence, new methods with ideas of thermodynamics were proposed. These methods are based on rewriting the marginal likelihood equation using intermediate distributions of parameters between the prior and posterior distributions [FP08].

3.2.1 Thermodynamic Integration for Marginal Likelihood

The Thermodynamic Integration is a method that proposes to rewrite the Integral 3.3 using ideas of thermodynamics, providing new estimators for the marginal likelihood. This method is able to rewrite the marginal likelihood integral in a way that it marginalizes the likelihood through many intermediate probability spaces of parameters, bridging the prior and posterior distributions of parameters. These distributions are also called tempered distributions or power posteriors [FP08].

Given a parameter prior distribution $p(\boldsymbol{\theta}|M)$ and a posterior distribution $p(\boldsymbol{\theta}|\boldsymbol{D}, M)$, then we define a power posterior distribution with "temperature" β as:

$$p_\beta(\boldsymbol{\theta}) = \frac{p(\boldsymbol{D}|\boldsymbol{\theta}, M)^\beta p(\boldsymbol{\theta}|M)}{z(\beta)},$$

where

$$z(\beta) = \int_\Theta p(\boldsymbol{D}|\boldsymbol{\theta}, M)^\beta p(\boldsymbol{\theta}|M) d\boldsymbol{\theta}.$$

Note that when $\beta = 0$, then $p_{\beta=0} = p(\boldsymbol{\theta}|M)$, the prior distribution of the parameters; also, when $\beta = 1$, then

$$p_{\beta=1}(\boldsymbol{\theta}) = \frac{p(\boldsymbol{D}|\boldsymbol{\theta}, M)p(\boldsymbol{\theta}|M)}{z(\beta)} = \frac{p(\boldsymbol{D}, \boldsymbol{\theta}|M)}{\int_\Theta p(\boldsymbol{D}, \boldsymbol{\theta}|M)d\boldsymbol{\theta}} = \frac{p(\boldsymbol{\theta}|\boldsymbol{D}, M)p(\boldsymbol{D}|M)}{p(\boldsymbol{D}|M)} = p(\boldsymbol{\theta}|\boldsymbol{D}, M),$$

the posterior distributions of the parameters. Therefore, if we vary the value of β from 0 to 1 we can create a path of probability distributions that connect the prior and posterior distributions.

We will explain now, how this sequence of distributions are applied to calculate the marginal likelihood.

Consider the derivative of $\ln z(\beta)$.

$$\begin{aligned}
\frac{d}{d\beta}\ln z(\beta) &= \frac{1}{z(\beta)}\frac{d}{d\beta}z(\beta) \\
&= \frac{1}{z(\beta)}\frac{d}{d\beta}\int_\Theta p(\boldsymbol{D}|\boldsymbol{\theta},M)^\beta p(\boldsymbol{\theta}|M)d\boldsymbol{\theta} \\
\text{(using that } \tfrac{d}{dx}c^x = c^x\ln c\text{)} \\
&= \frac{1}{z(\beta)}\int_\Theta p(\boldsymbol{D}|\boldsymbol{\theta},M)^\beta p(\boldsymbol{\theta}|M)\ln p(\boldsymbol{D}|\boldsymbol{\theta},M)d\boldsymbol{\theta} \\
&= \int_\Theta \frac{p(\boldsymbol{D}|\boldsymbol{\theta},M)^\beta p(\boldsymbol{\theta}|M)}{z(\beta)}\ln p(\boldsymbol{D}|\boldsymbol{\theta},M)d\boldsymbol{\theta} \\
&= \int_\Theta p_\beta(\boldsymbol{\theta})\ln p(\boldsymbol{D}|\boldsymbol{\theta},M)d\boldsymbol{\theta} \\
&= \mathbb{E}_{p_\beta(\boldsymbol{\theta})}[\ln p(\boldsymbol{D}|\boldsymbol{\theta},M)]. \qquad (3.4)
\end{aligned}$$

And it is not hard to see that:

$$\begin{aligned}
z(0) &= \int_\Theta p(\boldsymbol{\theta}|M)d\boldsymbol{\theta} = 1 \\
z(1) &= \int_\Theta p(\boldsymbol{D},\boldsymbol{\theta}|M)d\boldsymbol{\theta} = p(\boldsymbol{D}|M)
\end{aligned} \qquad (3.5)$$

Using equations 3.5 and Equality 3.4 we can write that:

$$\begin{aligned}
\int_0^1 \mathbb{E}_{p_\beta(\boldsymbol{\theta})}[\ln p(\boldsymbol{D}|\boldsymbol{\theta},M)]d\beta &= \int_0^1 \frac{d}{d\beta}\ln z(\beta)d\beta \\
&= \Big[\ln z(\beta)\Big]\Big|_0^1 \\
&= \ln p(\boldsymbol{D}|M). \qquad (3.6)
\end{aligned}$$

Then we have written an expression for the logarithm of the marginal likelihood. This expression is still hard to be calculated analytically, however from this equation we will be able to find estimators for the logarithm of the marginal likelihood, and consequently for the likelihood. To calculate these estimators, we will need to generate samples for a series of power posteriors of parameters, and this will be explained in the next section.

3.2.2 Estimation of the Marginal Likelihood

There are multiple approaches on estimating the Integral 3.6 and, usually it is necessary to find samples of $p_{\beta_t}(\boldsymbol{\theta}|M,\boldsymbol{D})$ for a sequence of values of β_t that vary from zero to one [Xu+10; VG07; FP08]. The methods that create such samples can take different approaches on the choice of the sequence $\beta_1 < \beta_2 < \ldots < \beta_T$, on the Metropolis-Hastings (MH) algorithms used, and finally on the estimator.

We are now going to show one possible methodology to estimate parameter values. First, we assume that there is a chosen sequence of $\beta_1 < \beta_2 < \ldots < \beta_T$, so we can explain how to generate samples of power posterior distributions with these values of β. After that we will explain how to choose the sequence of β values and, finally, present two possible estimators for $p(D|M)$.

Power posteriors sampling

Given a sequence of β values, the method we present to sample from the power posteriors has three different steps, and all of them use Metropolis-Hastings algorithms, similarly to what is done by Xu et al. (2010). For each of the T β values, the first two phases are run independently, generating T chains that are samples of the power posteriors. Then, on the third phase, each chain continues to grow, but not independently, because two random chains will have their last accepted points swapped. This exchange leads to a mixture of chains of samples of different power posteriors; this approach is called Populational Monte Carlo Markov Chain (Populational MCMC) [FP08].

The first step of sampling is run independently for each temperature and we call it **naive burn-in**. The Metropolis-Hastings performed on this step is started by taking a sample from the parameter priors. For each step, the proposal distribution has to generate positive values and the covariance matrix should be a diagonal covariance matrix, i.e. parameters are proposed independently. This covariance matrix is updated after every predefined number of iterations to adapt the proposal distribution according to the current sample. This type of MCMC algorithm, that updates its rules according to its trace is also called adaptive MCMC algorithm. This update is performed as proposed in Gelman et al. (2013):

- if the acceptance rate of parameter points in the last iterations is greater than 0.44, then increase the variance of the jump;
- if the acceptance rate is lower than 0.23, then decrease the variance of the jump.

This rule is a golden rule for Metropolis-Hastings calibration since the work of Roberts et al. [RGG97]. The rationale behind this rule is that chains generated with small variance tend to need more steps to cover the space, while chains generated with larger variance tend to reject more proposed parameters, staying at the same point of the chain for many iterations.

Given that we are taking a sample from a power posterior with temperature β_t, the target function is $p_{\beta_t}(\boldsymbol{\theta})$. Hence, if the current point is $\boldsymbol{\theta}^{(t,i)}$, the probability of accepting a proposed parameter $\boldsymbol{\theta}^* \sim J_{(t,i)}(\boldsymbol{\theta}^*|\boldsymbol{\theta}^{(t,i)})$, is $\min(1, r)$ with

$$r = \frac{p_{\beta_t}(\boldsymbol{\theta}^*)}{p_{\beta_t}(\boldsymbol{\theta}^{(t,i)})} \frac{J_{(t,i)}(\boldsymbol{\theta}^{(t,i)}|\boldsymbol{\theta}^*)}{J_{(t,i)}(\boldsymbol{\theta}^*|\boldsymbol{\theta}^{(t,i)})},$$

with $J_{(t,i)}$ being the proposal jump distribution on iteration i on chain of power posterior β_t. Since we do not have direct access to the $p_t(\boldsymbol{\theta}|M, \boldsymbol{D})$ distribution, and considering the definition of power posterior we presented in the beginning of the chapter, that $p_{\beta_t}(\boldsymbol{\theta}) \propto p(\boldsymbol{D}|M, \boldsymbol{\theta})^{\beta_t} p(\boldsymbol{\theta}|M)$, we can rewrite this equation as:

$$r = \frac{p(\boldsymbol{D}|M, \boldsymbol{\theta}^*)^{\beta_t}}{p(\boldsymbol{D}|M, \boldsymbol{\theta}^{(t,i)})^{\beta_t}} \frac{p(\boldsymbol{\theta}^*|M)}{p(\boldsymbol{\theta}^{(t,i)}|M)} \frac{J_{(t,i)}(\boldsymbol{\theta}^{(t,i)}|\boldsymbol{\theta}^*)}{J_{(t,i)}(\boldsymbol{\theta}^*|\boldsymbol{\theta}^{(t,i)})}. \tag{3.7}$$

After a pre-determined number of iterations, we stop the naive burn-in to move to the next sampling algorithm. By the end of the naive burn-in, we expect to have generated a sample that nearly approximates a sample of the posterior distribution. However, the sampled points might still be very correlated. To reduce this correlation and improve the posterior approximation, we will need the next two steps of sampling. We call the sampling algorithm that follows naive burn-in as **posterior shaped burn-in**.

The posterior shaped burn-in has this name because the used proposal distribution has a covariance matrix similar to an estimative of the covariance of the posterior distribution. To

create this estimative, we use, for each chain, the current sample of the posterior generated. On the first iteration of posterior shaped burn-in, we estimate the covariance of the posterior using the last half points sampled in the naive burn-in (we discard the first half of the chain because we assume that these points can have very low probability on the posterior distribution). For the second and further iterations of the posterior shaped burn-in, we estimate the covariance of the posterior considering the second half of sampled points of the first iterations plus all the points accepted after that.

The proposal distribution chosen on the posterior shaped burn-in must be multivariate with a covariance that is similar to the one estimated from the current sample of the posterior, differently from the naive burn-in, in which pairs of parameters $(\boldsymbol{\theta}_i, \boldsymbol{\theta}_j)$, $i, j \in \{1, \ldots, n\}$, $i \neq j$, were sampled independently. Similarly to the first sampling phase, we also sample each chain (for each β_t value) separately, and we also scale the covariance matrix according to the number of accepted parameters (as the golden rule for MCMC suggests). The probability of accepting a proposed parameter on this sampling phase is the same as it was on the first phase, as described by Equation 3.7.

By the end of the posterior shaped burn-in it is expected that the created sample is more likely to be a sample of the posterior distribution. However, there might still exist big correlations between sampled points of the same chain. To solve this problem, we need the third and last sampling phase: the **Populational Monte Carlo Markov Chain**. This algorithm continues the chains for each value of β_t, and after each step, a couple of chains have their last sampled value exchanged. This procedure mixes the chains and reduces correlation of sampled points.

At the begin of the Populational MCMC phase, we will have T chains of sampled parameters, containing the selected parameters of the first and second steps. For each of these chains there is a covariance matrix that was being used to determine the covariance of the proposal distribution of the respective temperature. On the third phase, we fix all of these matrices and, for each iteration, two steps are performed. The first step is to iterate all chains, using the respective fixed covariance matrix, identically to how it was done in the posterior shaped burn-in. The second step consists in choosing two different chains β_i and β_j, with $i \neq j$ and $i, j \in \{1, \ldots, T\}$, and exchange the last sampled parameters of these chains with a certain probability. The value of i is chosen randomly and uniformly over $\{1, \ldots, T\}$. Once i is chosen, the second chain, of temperature β_j, is chosen following a Discrete Laplacian distribution given i, with probability function $p(j|i) \propto e^{|i-j|/2}$. This algorithm, as we described, was proposed by Friel et al. (2008) and can be summarized in the following steps:

1. For each temperature β_t, $t \in \{1, \ldots, T\}$, update the t-th chain using MCMC with a proposal distribution that has the same covariance matrix used on the last iteration of the same chain, on the last sampling phase.

2. Choose uniformly i from $\{1, \ldots, T\}$. Then, choose j with probability density function $p(j|i) \propto e^{|i-j|/2}$. Finally, swap the last sampled points of the chains of power posteriors β_i and β_j, $\boldsymbol{\theta}^{(i,k)}$ and $\boldsymbol{\theta}^{(j,k)}$, with probability $\min\{1, r\}$, where:

$$r = \frac{p_{\beta_i}\left(\boldsymbol{\theta}^{(j,k)}\right)}{p_{\beta_i}\left(\boldsymbol{\theta}^{(i,k)}\right)} \frac{p_{\beta_j}\left(\boldsymbol{\theta}^{(i,k)}\right)}{p_{\beta_j}\left(\boldsymbol{\theta}^{(j,k)}\right)} \frac{p(j|i)}{p(i|j)}$$

   which can be simplified to:

$$r = \left[\frac{p\left(\boldsymbol{D}|\boldsymbol{\theta}^{(j,k)}\right)}{p\left(\boldsymbol{D}|\boldsymbol{\theta}^{(i,k)}\right)}\right]^{\beta_i} \left[\frac{p\left(\boldsymbol{D}|\boldsymbol{\theta}^{(i,k)}\right)}{p\left(\boldsymbol{D}|\boldsymbol{\theta}^{(j,k)}\right)}\right]^{\beta_j} \frac{p(j|i)}{p(i|j)}$$

After finishing the third step of sampling, the samples obtained on the first two phases are discarded and only the samples from the last phase are used to estimate the marginal likelihood.

Estimators of the Marginal Likelihood

The sampling procedure explained on the Section 3.2.2 produces samples of the power posteriors $p_{\beta_1}(\boldsymbol{\theta}), \ldots, p_{\beta_T}(\boldsymbol{\theta})$, and these samples can be used to estimate logarithm of the marginal likelihood:

$$\ln p(\boldsymbol{D}|M) = \int_0^1 \mathbb{E}_{p_\beta(\boldsymbol{\theta})}[\ln p(\boldsymbol{D}|\boldsymbol{\theta}, M)]d\beta. \tag{3.8}$$

This can be achieved using both numerical integration or creating an unbiased estimator. The choice of either approach of estimation will imply in a method for choosing the sequence of temperatures $\beta_1, \ldots, \beta_T$.

The method proposed by Friel et al. (2008), uses a numerical integration method to estimate the Integral 3.8. Given that T power posteriors, $0 = \beta_1 < \beta_2 < \ldots < \beta_T = 1$, were selected and samples of its respective distributions posteriors were generated, then using trapezoidal rule over the β values allows us to estimate the integral as:

$$\log p(\boldsymbol{D}|M) \approx \sum_{t=0}^{T-1}(\beta_{t+1} - \beta_t)\frac{\mathbb{E}_{p_{\beta_{t+1}}(\boldsymbol{\theta})}[\log p(\boldsymbol{D}|M, \boldsymbol{\theta})] + \mathbb{E}_{p_{\beta_t}(\boldsymbol{\theta})}[\log p(\boldsymbol{D}|M, \boldsymbol{\theta})]}{2}$$

and if the sample of power posterior of temperature β_t has L_t parameter points, then we can rewrite this equation as:

$$\log p(D|M) \approx \sum_{t=0}^{T-1}(\beta_{t+1} - \beta_t)\frac{\frac{1}{L_{t+1}}\sum_{i=1}^{L_{t+1}} \log p(\boldsymbol{D}|M, \boldsymbol{\theta}^{(t+1,i)}) + \frac{1}{L_t}\sum_{i=1}^{L_t} \log p(\boldsymbol{D}|M, \boldsymbol{\theta}^{(t,i)})}{2} \tag{3.9}$$

According to the work of Friel et al. (2008), a good temperature schedule for $\beta_1, \ldots, \beta_T$ that can be used in this approach is:

$$\beta_t = \left(\frac{t-1}{T-1}\right)^c,$$

with $t \in 1, \ldots, T$; with better results achieved when T is between 20 and 100 and c is between 3 and 5.

Another method, proposed by Xu et al. (2010), considers that β can be treated as a random variable, and therefore we can rewrite Integral 3.8 as:

$$\mathbb{E}_{p(\beta)}\left[\frac{\mathbb{E}_{p_\beta(\boldsymbol{\theta})}[\ln p(\boldsymbol{D}|\boldsymbol{\theta}, M)]}{p(\beta)}\right] \tag{3.10}$$

The author uses this ideas to derive the following estimator. First, the interval $[0, 1]$ is discretized into $S-1$ disjoint intervals $\Delta\beta_i = [t_{i+1}, t_i]$ such that $\sum_{i=1}^{S-1}(t_{i+1} - t_i) = 1$. Then, for each interval $\Delta\beta_i$, T_i temperatures are taken randomly from the uniform distribution on the interval $[t_{i+1}, t_i]$. Finally, the estimator of the logarithm of the marginal likelihood is given by:

$$\sum_{s=1}^{S-1}\frac{|\Delta\beta_k|}{T_k}\sum_{i=1}^{M_k}\log p(\boldsymbol{D}|M, \boldsymbol{\theta}^{(\beta_{k,i})}) \tag{3.11}$$

where $\beta_{k,i}$ is the i-th sampled temperature from the interval $\Delta\beta_k$; $\boldsymbol{\theta}^{(\beta_{k,i})}$ is a parameter sampled from the power posterior $p_{\beta_{k,i}}(\boldsymbol{\theta})$; and $|\Delta\beta_k| = t_{k+1} - t_k$. However, Xu et al. do not provide information about the discretization method of the interval $[0, 1]$.

3.3 Approximate Bayesian Computation

Approximate Bayesian Computation (ABC) is an approach that allows generating samples from the posterior distribution without accessing the likelihood function. Adding a model indicator parameter to parameters being sampled also allows us to create an approximate sample of the posterior $p(\boldsymbol{\theta}, M|\boldsymbol{D})$, and from this sample it is possible to estimate $p(M|\boldsymbol{D})$, which can be used as a model ranking metric. The main idea of ABC methods is to generate a sample from posterior by generating parameter points that, when plugged into a model, generates simulations that differ from the experimental measurements at most by some small ϵ.

A generic ABC method that generates a sample of the posterior $p(\boldsymbol{\theta}, M|\boldsymbol{D})$ is composed of the following steps:

1. Sample a parameter candidate $(\boldsymbol{\theta}^*, M^*)$ from some proposal distribution.
2. Simulate the model M^* with parameter values $\boldsymbol{\theta}^*$, generating simulated measurements $\phi(M^*, \boldsymbol{\theta}^*) = D^*$.
3. Calculate, for some distance function d, the value of $d(D^*, D)$. If $d(D^*, D) < \epsilon$ for some previously specified ϵ, then add $(\boldsymbol{\theta}^*, M^*)$ to the sample.
4. Repeat until some iteration limit.

The simplest ABC algorithm is the ABC Rejection, and it goes very similarly to the generic algorithm we just presented, with the specification that on step 1 the proposal distribution is the prior distribution. The output of this algorithm is a sample of the distribution $p(\boldsymbol{\theta}, M|d(\phi(M, \boldsymbol{\theta}), \boldsymbol{D}) \leq \epsilon)$. When $\epsilon \to \infty$ this is then a sample of the prior distribution, and as $\epsilon \to 0$ then this sample tends to be a sample of the posterior distribution [Pri+99]. This algorithm, however, does not perform well when the posterior distribution is very different from the prior distribution. For that reason, new ABC methods using Monte Carlo Markov Chains were created [Mar+03]. The ABC MCMC method proposed by Marjoram et al. (2003) produces a Markov Chain whose stationary distribution is $p(\boldsymbol{\theta}, M|d(\phi(M, \boldsymbol{\theta}), \boldsymbol{D}) \leq \epsilon)$. Nonetheless, this algorithm might still suffer from correlation in samples or even get stuck in regions of local peaks of probability. For that reason, the ABC sequantial Monte Carlo (ABC SMC)method was proposed [Ton+09].

The ABC SMC method creates a sequence of samples with the goal of getting closer to a posterior sample in each step. Let we simplify the notation by saying that $d(\phi(M, \boldsymbol{\theta}), \boldsymbol{D})$ is just $\rho_{M,\boldsymbol{\theta}}$. From a predefined sequence $\epsilon_1, \ldots, \epsilon_T$ the algorithm generates a sequence of samples that represents the distributions $p(\boldsymbol{\theta}, M|\rho_{M,\boldsymbol{\theta}} \leq \epsilon_1), p(\boldsymbol{\theta}, M|\rho_{M,\boldsymbol{\theta}} \leq \epsilon_2), \ldots, p(\boldsymbol{\theta}, M|\rho_{M,\boldsymbol{\theta}} \leq \epsilon_T)$. At the first generation, a sample of $p(\boldsymbol{\theta}, M|\rho_{M,\boldsymbol{\theta}} \leq \epsilon_1)$ is created by proposing points from the prior distribution. Then, for next generations the candidates to the sample are proposed based on the points of the last generation and their weight, plus some noise determined by a perturbation Kernel; the weight of a point $(\boldsymbol{\theta}^*, M^*)$ on generation t is an estimative of $p(\boldsymbol{\theta} = \boldsymbol{\theta}^*, M = M^*|\rho_{M,\boldsymbol{\theta}} \leq \epsilon_t)$. The pseudo-code 1 presents the ABC SMC algorithm.

The ABC SMC algorithm is implemented in Python language in a software called ABC-SysBio [Lie+14].

ABC SMC $(\mathcal{M}, D)$

Define the sequence $\epsilon_1, \ldots, \epsilon_T$.
Define N, the sample size for each generation.
Sample $\{(\theta^{(1,1)}, M^{(1,1)}), (\theta^{(1,2)}, M^{(1,2)}), \ldots, (\theta^{(1,N)}, M^{(1,N)})\}$ from $p(\theta, M|D)$.
Set $w^{(1,i)} = 1, \forall i \in 1, \ldots, N$.
for $t \in \{1, \ldots, T\}$ **do**
    $i \leftarrow 1$
    while $i \leq N$ **do**
        Sample M^* from $p(M|D)$, the model prior.
        Sample $(\theta^{(t-1,k)}, M^*)$ from the last generation with weight $w^{(t-1,k)}$.
        Create (θ^*, M^*) by perturbing $\theta^{(t-1,k)}$; $\theta^* \propto K^t(\theta|\theta^{(t-1,k)})$.
        if $p(\theta^*|M^*) = 0$ **then**
            Continue to next iteration.
        end if
        $D* \leftarrow \phi(M^*, \theta^*)$
        if $d(D^*, D) \leq \epsilon$ **then**
            $i \leftarrow i + 1$
            $(\theta^{(t,i)}, M^{(t,i)}) \leftarrow (\theta^*, M^*)$
        end if
    end while
    Calculate the weights of the population: $w^{(t,i)} = \frac{p(\theta^{(t,i)}|M^{(t,i)})}{\sum_{j=1}^{N} w^{(t-1,j)} p_{K^t}(\theta^{(t-1,j)}, \theta^{(t,i)})}$
end for
return

Algorithm 1: Pseudo-code of ABC SMC.

Chapter 4

Development of SigNetMS, a Software for Model Ranking

In this chapter we present the development process and implementation details of the SigNetMS software, which stands for **Sig**naling **Net**work **M**odel **S**election. This software is capable of producing an estimative of the marginal likelihood of a model given experimental data, using the ideas of Thermodynamic Integration. Besides implementation details, in this chapter, we also present the main difficulties of creating a computationally efficient implementation of such program, and also how we tackled these difficulties.

A few experiments are shown in this chapter, in order to support some of our implementation choices. All the experiments of this chapter were conduced in a server with an Intel Xeon E5-2690 CPU, and 252GB of RAM memory.

4.1 The SigNetMS Software

SigNetMS is a Python program that can be used as a tool for model selection, producing an estimative of the marginal likelihood of a model given experimental observations. The source code is available on Github[1] and it is open source, under the GNU General Public License.

This program expects as the input: a signaling pathway model, represented by a Systems Biology Markup Language (SBML) [Huc+03b] file, with the definition of reactions, kinetic laws and initial concentrations of chemical species; an Extensible Markup Language (XML) file with experimental data, including time series measurements of the biological phenomena of interest; another XML file with definitions of prior distributions of reaction rate constants; and, finally, a set of parameter values that determine the sampling process of model parameters. There are also optional parameters on SigNetMS, used to control random number generator seeds, number of execution threads, and verbose runs.

The output of the program is composed by an estimative of the marginal likelihood of the model, given experimental data, $p(\boldsymbol{D}|M)$ and a list of parameter values $(\boldsymbol{\theta}^1, \boldsymbol{\theta}^2, \ldots \boldsymbol{\theta}^l)$ that represent a sample of the distribution $p(\boldsymbol{\theta}|M, \boldsymbol{D})$. If one simulates the model M with parameter values from the sample, we expect that, the higher the marginal likelihood, the closer the created simulation is to the experimental data.

To calculate the marginal likelihood estimative, we use the ideas of Thermodynamic Integration we presented on section 3.2. Before implementing SigNetMS we also considered using another software, called BioBayes [VG08], that has a very similar approach to produce an es-

[1] https://github.com/gustavoem/SigNetMS

timative of the marginal likelihood. However, we did not proceed to use this software because there was only a graphical user interface version of the software, which made the process of running instances and collecting results cumbersome. We have also tried contacting the authors to obtain the source code, however they could not provide us that.

4.2 Creating an estimative of the marginal likelihood

To create estimates of the marginal likelihood using the Thermodynamic Integration approach, we need to define a likelihood function $p(\boldsymbol{D}|M, \boldsymbol{\theta})$ and also samples power posterior distributions $p_\beta(\boldsymbol{\theta})$ for a sequence of values of β. After samples are produced, we use them to produce the estimative, and on SigNetMS, we use the trapezoidal rule to approximate the marginal likelihood, given by the equation 3.9. Inspired by the work of Friel et al., we discretize the interval $[0, 1]$ of power posteriors using the $\beta_i = \left(\frac{i-1}{T-1}\right)^c$, where $T = 20$, $c = 4$ and $i \in \{1, 2, \ldots, T\}$. In the following sections, we explain how we implemented the likelihood function, and how samples of power posteriors are produced.

4.2.1 Implementing the likelilhood function

The likelihood function we used is the same as we defined in equation 3.1:

$$p(\boldsymbol{D}|M, \boldsymbol{\theta}) = p_{\mathcal{N}_{(\vec{0},\Sigma)}}(\phi(M, \boldsymbol{\theta}) - \boldsymbol{D}),$$

where $\boldsymbol{D}$ is the experimental measurement, M is the model, $\boldsymbol{\theta}$ is a set of parameter values, Σ is the variance of the error of experimental observations, and, finally, ϕ is a function that calculates an approximation of the results of the experiment that produced $\boldsymbol{D}$, applied to the simulated environment of model M with parameter values $\boldsymbol{\theta}$. This simulation of the model is created by deriving a system of ordinary differential equations, and then numerically integrating this system, over the time steps defined by the experimental measurements. To reproduce the same experiment that generated $\boldsymbol{D}$, SigNetMS expects that the XML file containing experimental data also contains a mathematical representation of which quantity was measured, in terms of concentrations of chemical species of the system.

The numerical integration of the system is alone a hard problem, and therefore, we used third-party software to produce such integrations. The most popular software available for this problem conduce iterative algorithms that, step by step, approximate the state of the system for a time interval. It is important to know that some instances of the problem can be stiff, meaning that they may make the integration algorithm be unstable, since it may need consecutive iterations with really small steps. Since we did not have time to go in details of when such cases occur, we chose third-party software that can adapt the used algorithm according to the stiffnes of the instance.

After the implementation of the function ϕ, most of the work to implement the likelihood function is done. The remaining work is to calculate the value of the probability density function $p_{\mathcal{N}_{(\vec{0},\Sigma)}}$ and that can easily be accomplished using statistical packages such as SciPy [Vir+20].

4.2.2 Sampling parameters from power-posteriors

After implementing the likelihood function, we can move to the creation of samples of power posteriors. The methodology we used to generate such samples is identical to the one we presented on section 3.2.2. And for this reason, we divided the sampling process in three phases:

naive burn-in, adaptive burn-in and Populational MCMC; all of them are types of a Metropolis-Hastings procedure. The number of iterations of each phase is determined by SigNetMS's arguments, and each phase has a different scheme to determine the proposal distribution.

On the first phase, we start the sample of every chain (for each value of β) with a random draw from the prior distribution of parameters. Before the first iteration, we also create an estimative of the variance of the logarithm of each model parameter, independently. These estimates compose the first covariance matrix of the jumping distribution; we use a diagonal matrix where the diagonal elements are set as the estimated variance of the logarithm scaled sample of the associated parameter. This matrix is rescaled according to the acceptance rate, as described in section 3.2.2, after a number of iterations that is defined in one of the arguments of SigNetMS. For each iteration, we determine that the jumping distribution is a multivariate log-normal $(\boldsymbol{\mu}, \Sigma)$ distribution, with covariance matrix as explained before, and with $\boldsymbol{\mu} = \log_e(\boldsymbol{\theta}^t)$, where $\boldsymbol{\theta}^t$ is the current sample point, i.e., we take a sample $\boldsymbol{X}$ of the multivariate normal $\mathcal{N}(\boldsymbol{\mu}, \Sigma)$ and then we set our sampled value as $\boldsymbol{Y} = \exp(\boldsymbol{X})$, which is a standard procedure to produce samples of lognormal distributions.

On the second phase, the posterior shaped burn-in phase, we also use a lognormal as the proposal distribution, however the covariance matrix is not diagonal. Half of the sample produced in the first phase is discarded, and for each step of the second phase, we calculate the covariance of the log-scaled current sample (the latter half of samples of the first phase plus the current sample of the second phase), producing the matrix used as the covariance matrix for the proposal distribution. Similarly to the first phase, the proposal distribution also uses $\boldsymbol{\mu} = \log_e(\boldsymbol{\theta}^t)$. It is important to remember that up to the end of this sampling phase, each power posterior sample is created independently.

At the third and last phase, a Populational Monte Carlo Markov Chain procedure is performed. In this procedure, we iterate each chain of power posterior samples using the same algorithm of the second phase (except we do not update the covariance matrix anymore), followed by an exchange of the last sampled points on two random selected power posteriors. At the end of this phase, we discard parameters sampled on previous phases and we set the actual sample as all the parameters sampled in this phase. Note that, differently from the first two phases, the third phase depends on a synchronism of different power posteriors in order to mix different power posterior samples.

4.3 Fast system integration and parameter sampling

The numerical integration of a system of ordinary differential equations is one of the most computationally expensive processes for the estimation of the marginal likelihood. Besides being computationally expensive, the integration is also performed numerous times; for each calculation of the likelihood function, one integration is necessary. Note that the likelihood function is evaluated for all proposed jumps in the sampling, for each of the power posterior distributions. Even in basic experiments, we may evaluate this function hundreds of thousands of times. This is the main reason why our first implementation of SigNetMS did not cope even with toy models. Therefore, we needed to add optimizations to our software before experimenting with more complex instances.

Since most of the computational time spent by SigNetMS is concentrated in the generation of samples of power posteriors, we decided to focus optimizations in two different aspects of the sampling. The first aspect, is to reduce the required time to integrate a model, exploring different approaches to represent the model; the second aspect, is to take advantage of the independence of the sampling of two different power posteriors, in the first two phases of

sampling.

4.3.1 Optimizing the integration of a system of ordinary differential equations

The first consideration that was taken into account when optimizing the integration of the model was to use different algorithms, because, in general, algorithms prepared for non-stiff instances are less time consuming. However, because of the different nature of instances of the problem, we could not choose between algorithms prepared for stiff or non-stiff instances, and then we chose to continue using the `odeint` integrator, offered by SciPy, which can adapt according to the stiffness of the instance. The integrator `odeint` is actually a wrapper to the `LSODA` integrator, which is part of the Fortran package called `ODEPACK` [Hin82] language.

One common approach to reduce the computational time needed on the integration (and improve accuracy) is to provide the Jacobian matrix of the system. This matrix is essential to many integration methods to determine the next value of the unknown function (with a certain accuracy), and if this matrix is not provided, algorithms like `LSODA` will approximate this function, decreasing the computational efficiency. Considering the types of chemical reactions we consider on this work, we could implement a function that produces the Jacobian function of a system of ordinary differential equations that represents the model of interest.

However, after implementing such matrix derivation, and providing it to the integrator, the computational time required to produce integrations increased. That led us to believe that our computational representation of functions, the one that represents the system and the Jacobian, were not efficient. At this point, our representation of both system and Jacobian were an array of strings (since there is one function for each chemical species, to determine concentration change); then, the evaluation of these functions was equivalent to interpreting the strings of all the functions and then evaluating them. Even though that was the simplest implementation to evaluate functions, it was clear for us that some pre-processing was necessary to create a better representation.

To improve the computational time needed to evaluate these functions, we considering using SymPy, a Python package that allows symbolic mathematics. The idea was to provide the same strings of the functions to SymPy and create objects that represent such functions symbolically. After that, we could also remove our differentiation method, since SymPy offers one. However, the ‘odeint‘ package is not prepared to receive a SymPy object to perform integrations, instead, it expects to receive a Python function. Fortunately, SymPy offers methods to symbolic functions that allows code generation, and more specifically, Python code generation.

Then, we experimented two ways of using SymPy function objects to produce Python functions to represent the system and its Jacobian. The first one is to use `lambdify`, which creates a Python function that represents the SymPy object. The second one, is to generate C language code, and then use Cython to compile, import and wrap it as a Python function. Fortunately, again, SymPy provides an utility function that does all the work, called `autowrap`. We proceeded comparing all of these approaches on representing the model with the repeated integration of a simple model, composed by five reactions and five chemical species. On this experiment, we performed multiple integrations of the model; it is important to note that our goal is to reduce the overall time of many integrations, therefore, comparing the time of one integration only is not enough to determine the best choice. This is important because the last two approaches have a pre-processing stage where the system function is created, wrapped or even compiled, and this stage is only necessary to be ran once, before the first integration.

As we can see on Table 4.1, the approach where a C code is generated is the best option for

Number of Integrations	Average time (seconds) to perform a sequence of integrations		
	String Evaluation	`sympy.lambdify`	`sympy.autowrap`
10	2.98	0.8	0.9
100	35.3	5.9	6.6
200	72.1	15.8	13.1
400	139.1	33.1	26.9

Table 4.1: The average time spent by different approaches on a sequence of integrations of a signaling pathway model containing five reactions and five chemical species. Each entry is an average taken after 50 runs of this experiment. It is important to state that the integrations using `autowrap` *produce the C code and its Python wrapper only once (in a single experiment repetition), on the first integration. The String Evaluation column represents the string evaluation approach, without providing the Jacobian matrix, since we previously found out that in this approach, providing such matrix slows down the procedure. Note that with 10 integrations, the execution time with 'sympy.lambdify' and 'sympy.autowrap' are similar, however, with 400 hundred integrations, the compilation overhead is washed off making the C code approach faster.*

our application, where multiple integrations are needed. If the number of integrations needed are very small, the simpler `sympy.lambdify` should be the best fit, since it does not have the C code compilation overhead as `sympy.autowrap` has. Finally, this table also shows how bad string evaluation is, compared to generating a Python function to represent the system.

4.3.2 Parallel sampling of power posteriors

Parallel processing is a common approach to promote a better use of computational resources and, consequently, to reduce an algorithms execution time. Parallelization is usually successful when there are multiple tasks to be done, with few communication and synchronization. In the case of SigNetMS, during the first two phases of sampling, naive burn-in and posterior shaped burn-in, the sampling process occurs independently between different power posteriors. In the last phase, however, different power posteriors need to be synchronized every iteration, because of the mixing procedure that takes the last sampled parameter of two different chains.

Therefore, we proceeded to create a parallelization of both first and second phases of sampling. We achieved parallelization using the map pattern, where a function, passed to a map framework, is applied to a list of elements. In our application, the list of elements is a list of power posteriors values to be sampled, and the function is the first two phases of the sampling procedure. The parallelization arises when we consider that there is a pool of workers, each one in a different process, that can be assigned to apply the sampling function to a power posterior. The size of this pool is defined by the user as one of the SigNetMS parameters; if the user does not define a value, the algorithm sets the number of workers as the number of CPU cores available on the machine.

To understand how this parallelization affects the execution time of SigNetMS, we designed an experiment where a sequence of 500 sampling steps (of the first phase) are performed, for 40 power posteriors, for a model with 5 chemical reactions and 5 chemical species. The results are shown on Figure 4.1, and as we can see, we could reduce the execution time as we increased the number of workers. There is however, some saturation in time execution improvement after a number of workers. That is explained by the number of jobs that the pool receives to be done, which is the number of power posterior values to be sampled, 40 in our experiment (and also on SigNetMS). Even though we have this limitation, we believe that this simple parallelization is enough, and more up to standard solutions can be proposed in future works.

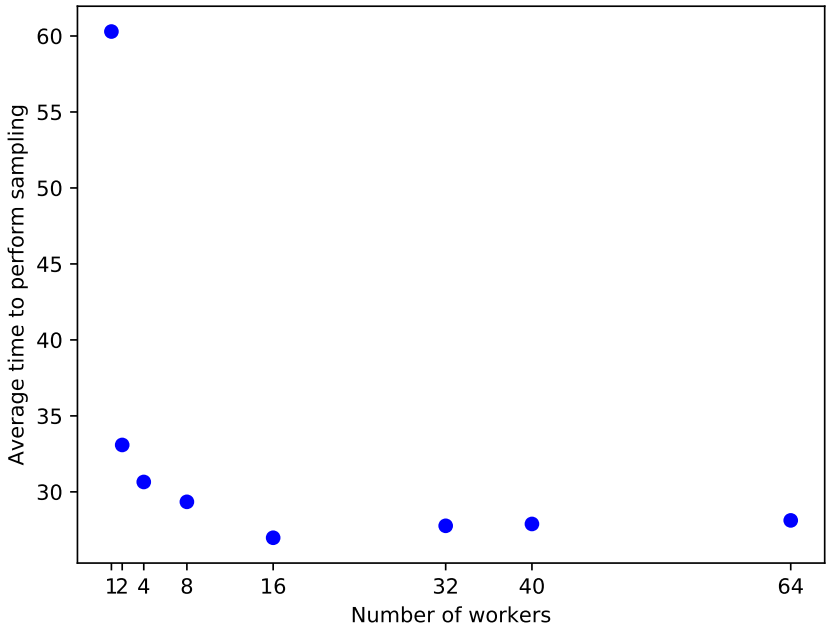


Figure 4.1: Average execution time of 500 *steps of the burn-in sampling, with varying number of workers. The average values are taken over 50 repetitions of the experiment. The model being sampled on this experiment contains 5 chemical reactions and 5 chemical species.*

Chapter 5

Experiments and Results

5.1 Choosing a Software for Model Selection

To choose between SigNetMS and ABC-SysBio, we performed a model selection experiment. This experiment, originally performed on the work of Vyshemirsky and Girolami [VG07], consists in creating artificial experimental data from a model of cell signaling pathway, and then selecting between four different models, including the correct one. Using SigNetMS and ABC-SysBio we should be able to create a ranking of the four models, in which we expect to see as the best, the model we used to create the experimental data. More than that, we should analyze the produced results to check if simpler models are preferred over complex models; we should also check if the simulations produced by the models, with the estimated sample of the posterior distribution of parameters, approximates experimental data.

5.1.1 A simple instance of the model selection problem

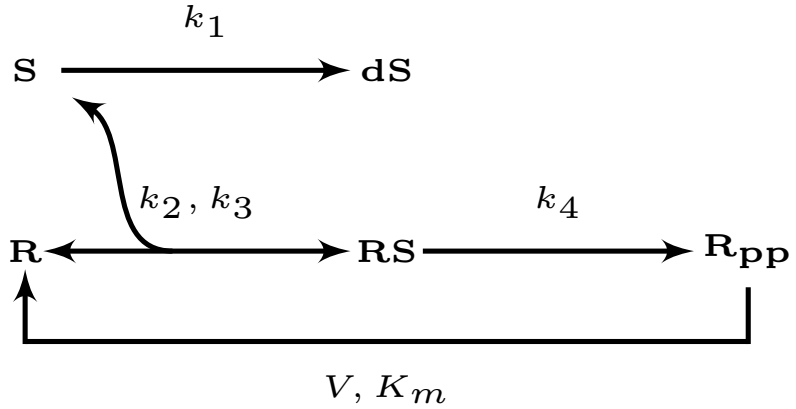


Figure 5.1: A diagram that represents the correct model our simple model selection experiment. This model represents a common motif, and has S and Rpp as input and output, respectively. This model contains five reactions: the decay of S to dS with k_1 as reaction rate constant; the reversible reaction $\mathrm{S} + \mathrm{R} \underset{k_2}{\overset{k_1}{\rightleftharpoons}} \mathrm{RS}$*; the first order reaction* $\mathrm{RS} \xrightarrow{k_4} \mathrm{Rpp}$*; and the Michaelis-Menten reaction* $\mathrm{A} \xrightarrow{V, K_m} \mathrm{B}$.

We start our model selection problem with the correct model, which is a signalling pathway composed by five reactions and five chemical species. Figure 5.1 shows a diagram with this

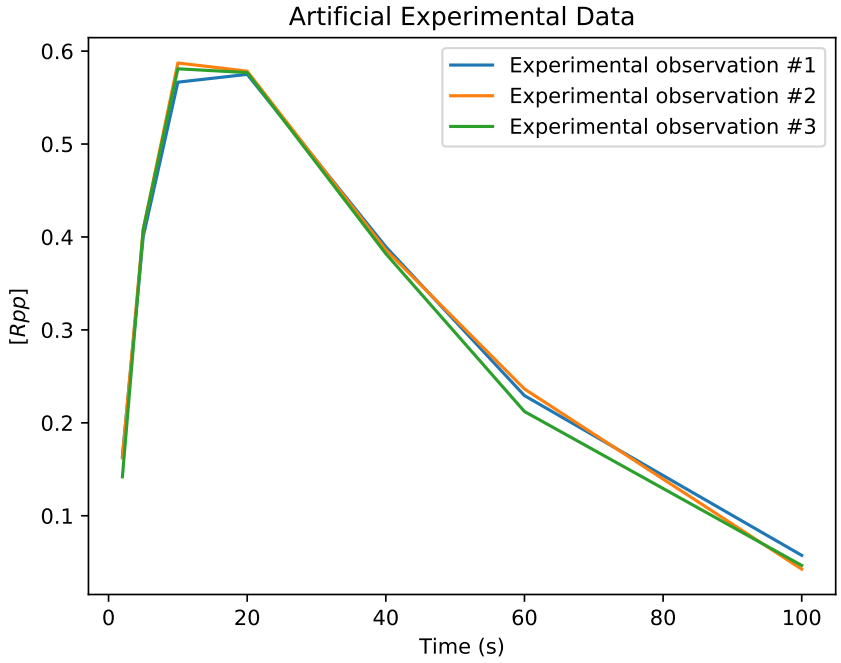


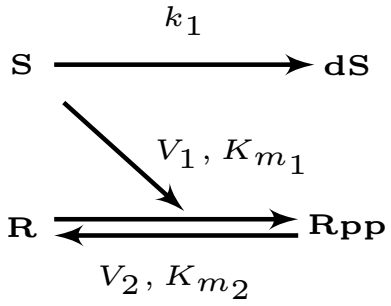
Figure 5.2: The dynamics produced by the correct model, with pre-defined reaction rate constants plus a small Gaussian error, for each time point. The measurement taken from the model is the concentration of the Rpp species, which we denote as [Rpp]. We linearly interpolate the experimental measure points to produce a continuous dynamics from 2s to 100s.

model. This model represents a common motif, and it has as the input signal the chemical species S, and as the output the chemical species Rpp; the experimental measurement used is the concentration of the output chemical species, which we donote as [Rpp].

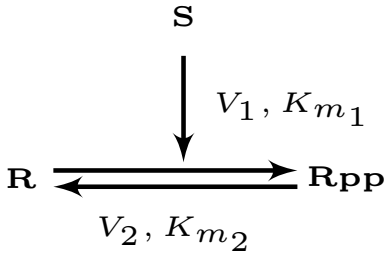
In this experiment, for the sake of simplicity, we neglect the units of reaction rates constants and initial concentrations. The initial concentrations used are: S = 1, R = 1, dS = 0, RS = 0, $R_{pp} = 0$. To create the experimental data, the reaction rate constants we used have the values: $k_1 = 0.07$, $k_2 = 0.6$, $k_3 = 0.05$, $k_4 = 0.3$, $V = 0.017$, and $K_m = 0.3$. It is important to remember that we discard reaction rate constant values during model selection; initial concentrations, however, are still provided during this phase. To generate experimental data, we simulate the dynamics of this model, using these parameter values, on the time steps of: 2s, 10s, 20s, 40s, 60s and 100s. Three simulations are created, and to each one of them we add, for each time measurement, a Gaussian error with mean 0 and standard deviation 0.01. A representation of the three experiment repetitions are showed on figure 5.2.

To assess the ranking produced by each of the model selection software, we compare the first model with three other models (based on the correct model): a simplified model; an overly simplified model, which should not be able to generate the observed dynamics; and, finally, a generalization (more complex) model. Figure 5.3 shows diagrams that represent the three alternative models.

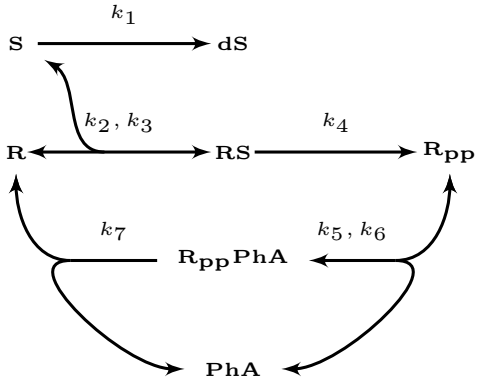
Before talking about results produced by different software, we should note that this choice of candidate models are made so we can analyze more than the ability of the software to correctly rank the correct model as the best model. First, consider that we introduced a spurious model, represented on figure 5.3(b) which neglects a crucial reaction, making it impossible to reproduce the experimental data; we expect this model to be ranked last between all models. Then, there



(a) simplified model



(b) overly simplified model



(c) generalization model

Figure 5.3: The diagrams of three other candidate models, based on the correct model that was presented before on Figure 5.1. The model 5.3(a) is a simplification where we neglect the chemical species RS, and we use the Michaelis-Menten to represent the reaction R $\longrightarrow$ Rpp *with S working as a catalyst. Model 5.3(b) is the over simplified model, as it neglects the decay of S; we do not expect this model to reproduce experimental data, since the constant concentration level of S tend to continuously produce Rpp, a species that, after 20 seconds, has a monotonic decreasing concentration. Finally, Model 5.3(c) is a generalization of the correct model, as it generalizes the reaction* Rpp $\longrightarrow$ R*, as instead of using the Michaelis-Menten kinetics, we use the enzymatic reaction* Rpp+PhA $\rightleftharpoons$ RppPhA $\longrightarrow$ R+PhA*; even though we expect this model to be able to reproduce observed dynamics, we also expect that the complexity of this model gets penalized.*

are two options to the correct model, one a simplification, and the other a generalization. For these models, we expect that the experimental dynamics are possible, however, we should be observant of how they are ranked according to their complexity. That is important because, one of the goals on using a Bayesian approach for model selection is that these approaches tend to automatically penalize overly complex models.

5.1.2 Solving a simple model selection instance using ABC-SysBio and SigNetMS

After defining the candidate models and producing the artificial experimental data, we proceeded to perform the experiment of model selection. The instance related information provided to SigNetMS and ABC-SysBio is the same: a model, with predefined initial concentrations of chemical species; a set of experiments, with the same time steps, with measurements of the concentration of Rpp; and a file containing prior distributions for each one of the model parameters. It is important to remember that the output produced by each software is different.

SigNetMS produces an estimative of $p(\boldsymbol{D}|M)$ and also a sample of the posterior distribution of parameters $p(\boldsymbol{\theta}|M, \boldsymbol{D})$ which is, in fact, composed by samples of all power posterior distributions $p_\beta(\boldsymbol{\theta})$ with values as we described on 4.2. ABC-SysBio, on the other hand, produces an estimatives of $p(\boldsymbol{\theta}, M|\boldsymbol{D})$ that might be closer to this target distribution on each iteration. Note that in this experiment, we need to run SigNetMS for every model, while on ABC-SysBio we only need to run the software once for all four candidate models.

The prior distribution of parameters are the same as used by Vyshemirsky and Girolami [VG07]. All model parameters priors are Gamma(1, 3), where the first and second arguments are shape and scale. Gamma and Lognormal distributions are often used as prior for parameters because they have a zero probability density for negative values.

For ABC-SysBio we decided to use its feature of automatically choosing the schedule of threshold values, which is based on the acceptance of produced individuals on each iteration. For SigNetMS, we used the following parameters values: 15000 iterations of the naive burn-in, and 5000 iterations of the posterior shaped burn-in, with 1000 iterations between covariation matrix rescales, and 3000 iterations of the Populational MCMC. We used an empiric approach to determine these parameter values, observing similar results when the number of iterations are greater than these.

The ranking produced by ABC-SysBio and SigNetMS

The ABC-SysBio run created 26 populations of parameter values, each of them with 100 individual parameters values. At the last iteration, the algorithm stopped with $\epsilon = 1$ and the following estimates:

- $\hat{p}(M = \text{Correct Model}|\boldsymbol{D}, \epsilon = 1) = 0.005$;
- $\hat{p}(M = \text{Simplified Model}|\boldsymbol{D}, \epsilon = 1) = 0.014$;
- $\hat{p}(M = \text{Incorrect Model}|\boldsymbol{D}, \epsilon = 1) = 0.976$;
- $\hat{p}(M = \text{Generalization Model}|\boldsymbol{D}, \epsilon = 1) = 0.003$.

These estimates induces the ranking 3, 2, 1, 4.

After running the SigNetMS software four times, one for each model, we were able to get the following estimates:

- $\log \hat{p}(\boldsymbol{D}|M = \text{Correct Model}) = 26$
- $\log \hat{p}(\boldsymbol{D}|M = \text{Simplified Model}) = 21$
- $\log \hat{p}(\boldsymbol{D}|M = \text{Incorrect Model}) = -1$
- $\log \hat{p}(\boldsymbol{D}|M = \text{Generalization Model}) = 19$

Comparing the ranking produced by ABC-SysBio and SigNetMS

Before comparing the model ranking produced by ABC-SysBio and SigNetMS, we should state that the ranking achieved by Vyshemirsky and Girolami [VG07], on the original work that introduced this instance, is: Correct Model $\prec$ Generalization Model $\prec$ Simplified Model $\prec$ Incorrect Model. On this work, a methodology similar to SigNetMS was used.

On ABC-SysBio results, we see that the Correct Model was not ranked first, and, surprisingly, the Incorrect Model was ranked first. More than that, when the algorithm stopped, other

candidate models were considered with low probability of being the "true" model, and therefore we cannot strongly state a ranking between the other three candidates.

On SigNetMS results, we see that the Correct Model was ranked first and the Incorrect Model is ranked last as expected. For these two models, SigNetMS results are equal to the results of Girolami and Vyshemirsky, and for the other two models, the ranking is the opposite. On SigNetMS, we ranked the Simplified Model as better than the Generalization Model. It is important to note here that, in fact, the Generalization Model, which is more complex, was actually ranked worse than the Correct Model; that is an evidence that this approach does penalize the complexity of models.

Analyzing the posterior distributions produced by ABC-SysBio and SignetMS

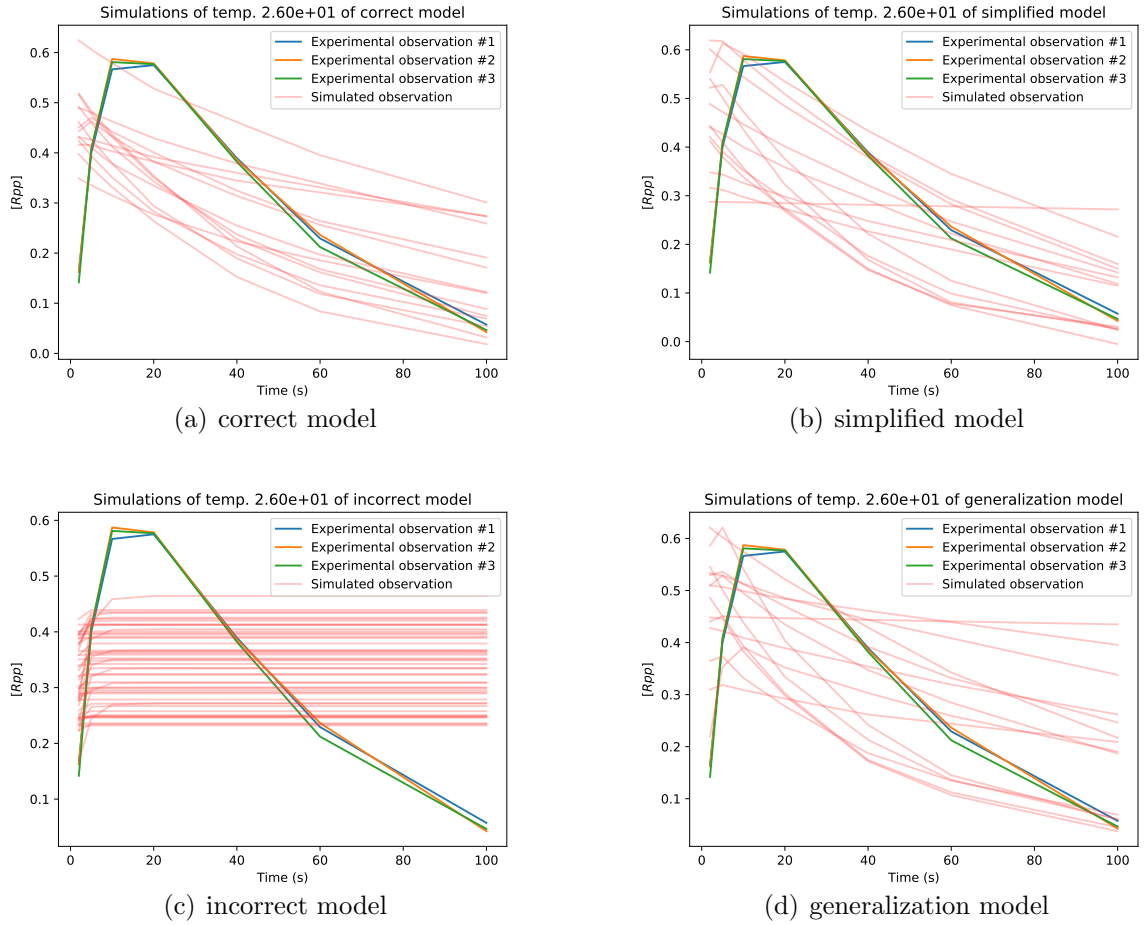
If we consider only the ranking produced, there are indications that SigNetMS is a better choice for our application. However, we should also take into account other output information produced by both software, relative to the distribution of model parameters. ABC-SysBio algorithm produces in every iteration a population of parameters that, when applied to an specific model, creates a simulation that is at most epsilon distant to the experimental measurements, with decreasing epsilon as the iteration number grows. SigNetMS, on the other hand, produces samples of forty power posterior distributions $p_\beta(\boldsymbol{\theta})$, and although there is no threshold like there is on ABC-SysBio, we expect that the closer the value of β is to 1, the closer should be the produced simulation to the experimental measurements; this is explained by the fact that the power posterior distributions $p_0(\boldsymbol{\theta})$ and $p_1(\boldsymbol{\theta})$ are, respectively, the prior and posterior distribution of parameters.

A possible approach to analyze the produced parameters is to simulate models with those parameter and create simulations to be compared with the experimental measurements. For ABC-SysBio, in a population of 100 parameters, including the model indicator as one of the parameters, we are able to simulate and visualize the generated experimental measurements for all individuals; on SigNetMS, on the other hand, the number of parameter values produced is much greater, than a randomly chosen subset of parameters should be enough. With such experiment, we are then able to identify what dynamics were created on the candidate models according to the estimated posterior distribution of parameters.

On figure 5.4 we present the dynamics of sampled parameters of the last iterations of the ABC-SysBio run. We can see on this figure that ABC-SysBio could not produce a set of parameter values that allows the model to represent the dynamics observed on the experiment. More than that, we can see that the incorrect model had the best fit, and the dynamics produced by the sampled parameters induces a nearly stationary dynamics of $[Rpp]$, with intermediary values of concentration. With these results, we can understand that the ranking produced by ABC-SysBio is incorrect because the software could not find suitable parameter values that allow models to approximate the dynamics observed on experiments.

On figure 5.5 we present the dynamics of a subset of parameters of the posterior distribution (or power posterior of $\beta = 1$), for all four candidate models. We can see on this figure that SigNetMS could not find parameter values that allow the incorrect model to reproduce experimental observations, which is expected, and that the three other models candidate models could closely reproduce the experimental observations. It is also interesting to observe the dynamics produced by sampled parameters for other values of β, which is shown on figure 5.7, for the correct model only. Remember that from $\beta = 0$ to $\beta = 1$, a sequence of power posterior distributions is constructed by SigNetMS, bridging the prior and posterior distributions.

The ranking and the simulations indicate that SigNetMS is more appropriate for our applications. To further investigate the results produced by this software, we also created plots



(a) correct model

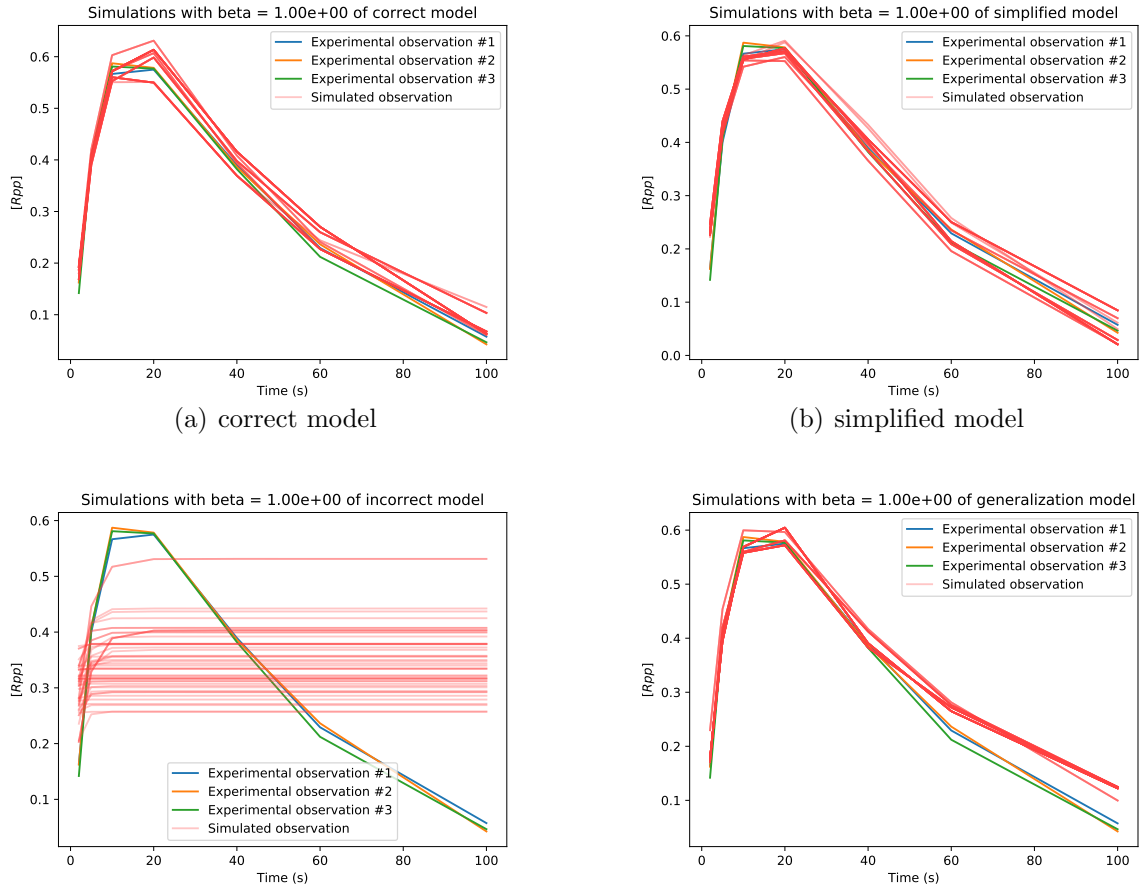
(b) simplified model

(c) incorrect model

(d) generalization model

Figure 5.4: The simulated dynamics of the four candidate models, using parameters generated on the ABC-SysBio software. There are 100 individuals generated on each iteration of the algorithm, and each individual consists of a list of parameter values and a model indicator. Because of this, the number of produced simulations is not equal between models, in fact, the better the fit of a model, the higher the number of individuals representing such model, and therefore, the higher the number of simulations shown. Each red line represent an individual simulation, and stronger red lines represent overlapping simulations. Lines with blue, yellow and green color represent experimental observations.

of approximations of the produced power posterior samples, presented on figure 5.7. These density function estimates were created using the `distplot` function of the Seaborn Python package, which uses Gaussian Kernel Densinty Estimate (KDE) to provide an estimation of the density function given a sample of such distribution. The presented figure shows estimated power posterior distributions, with different values of β, for the k_1 parameter of the correct model, which had value 0.07 when artificial experimental data was created. We can see that as we increase the value of β, the posterior distribution concentrates on values around the "true" value of the parameter. It is also interesting to see that although the estimated posterior is not necessarily centered on the "true" value, the created simulations do approximate the experimental observations.



(a) correct model

(b) simplified model

(c) incorrect model

(d) generalization model

Figure 5.5: The simulated dynamics of the four candidate models, using randomly chosen subsets of parameters from the sample produced by SigNetMS of the power posterior distribution of $\beta = 1$, which is the posterior distribution of parameters, $p(\boldsymbol{\theta}|M, \boldsymbol{D})$. Red translucent lines represent the simulated dynamics, using the sampled parameters, and stronger lines represent overlapping simulations. Lines with blue, yellow, and green color represent experimental observations.

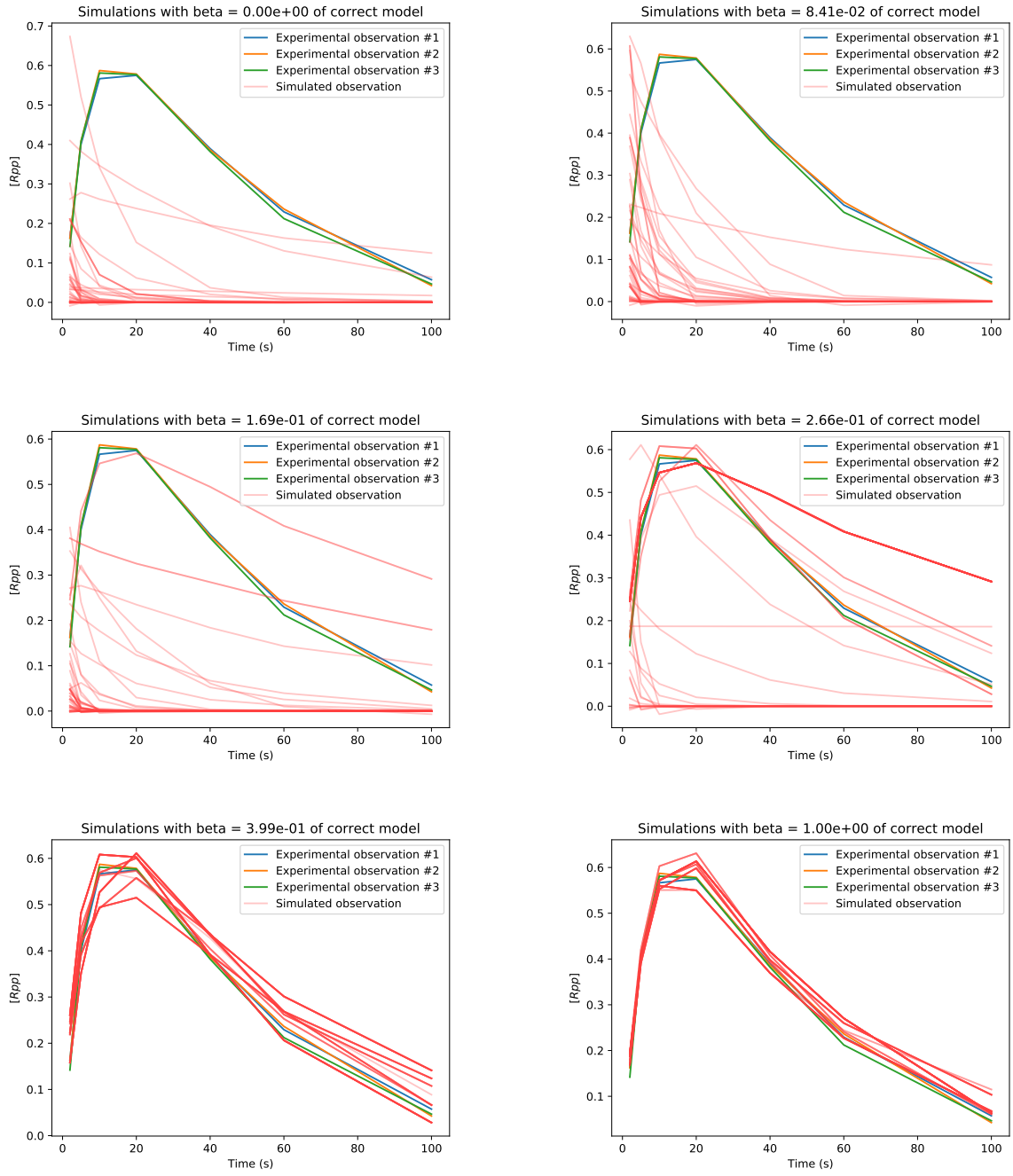


Figure 5.6: The dynamics induced by samples of different power posterior distributions of parameters of the correct model. The value of β increases from left to right and from top to bottom. We can observe how the curve of simulations progressively fits the curve of experimental observations.

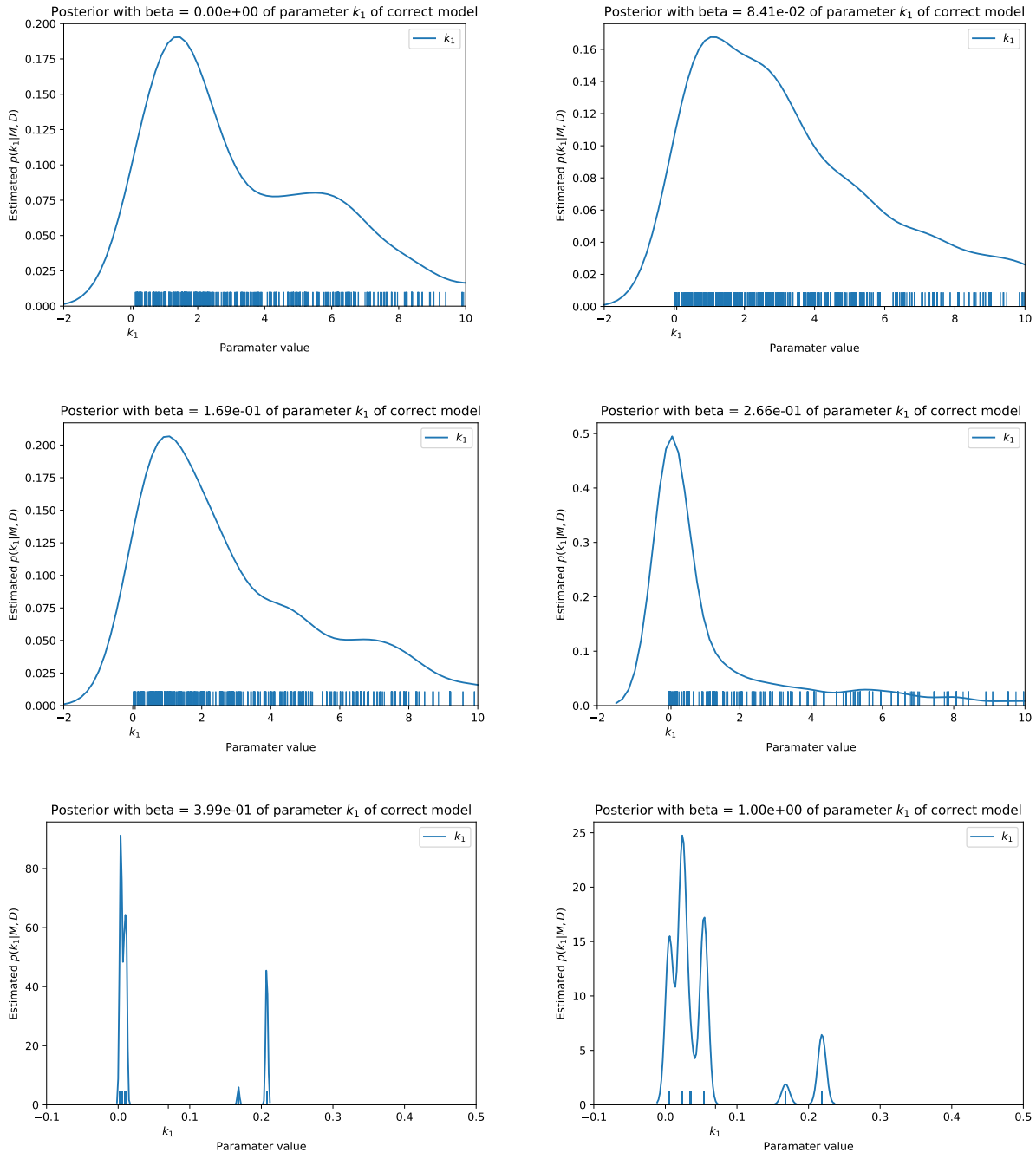


Figure 5.7: Approximation of the power posterior distribution of the samples created by SigNetMS of parameter k_1 of the correct model. We show on this graph the estimated distribution of six different power posteriors, with increasing value of β from left to right, from top to bottom. The approximation of this graph is created using `distplot` *function of Seaborn package. The value used for this parameter on the creation of the experimental data is* 0.07*, and it is represented on the axis of the plots.*

5.2 Model Selection as a Feature Selection Problem

After defining that SigNetMS is our software choice for model selection, we are now able to experiment and analyze the approach of solving a model selection problem as a feature selection problem, using a Bayesian approach to define a cost function. To accomplish this, we created another simple instance of model selection. Although this instance is still a toy model, we will be able to access the space of possible solutions and get a glance of the cost surface induced by SigNetMS over this space.

A feature selection instance can be defined by a pair (S, c) where S is a set of features and c is a cost function that evaluates subsets of S. The space of solution is usually the power set of S, $\mathcal{P}(S)$, and the cost function usually takes values from this space to positive real values, $c : \mathcal{P}(S) \to \mathbb{R}$. An optimal solution is a subset $X \in \mathcal{P}(S)$ such that $c(X) \leq c(Y), \forall Y \in \mathcal{P}(S)$, however, it is important to notice that the size of the search space grows exponentially with the number of features and, in practice, with time consuming cost functions, it is computationally unfeasible use optimal search algorithms. That is exactly our case, since the cost function we propose to use, based on SigNetMS, depends on the estimation of multiple power posteriors of parameters and includes numerous numerical integrations of a system of ordinary differential equations.

It is often useful to represent the search space with a boolean lattice, which is defined by the power set $\mathcal{P}(S)$ and the partial order relation $\subseteq$. More than an aid to represent the search space, the boolean lattice provides a structure that is useful for search algorithms to define paths and to take advantage of surface of the search space, as it is done on algorithms for the U-Curve problem, a special case of the feature selection problem where the cost function describes a u-shaped curve on every chain of the boolean lattice. The U-Curve problem is still an NP-hard problem as the feature selection problem is [Rei12], however there are heuristics and optimal algorithms that can be used on the U-Curve problem to produce a quality answer (optimal or close to be optimal) with a feasible computational time. Moreover, one can produce good results using U-Curve algorithms to solve a feature selection instance that is not necessarily U-Curve, but does reproduce u-shaped curves with a few oscillations on chains of the search space.

In our application, we are going to convert a model selection problem into a feature selection problem. To do so, we define a set of candidate reactions S and a base model. Then, we consider that a set of feature $X \in \mathcal{P}(S)$ represents a candidate model composed by the base model plus the reactions from X. The cost function we use is the logarithm of the marginal likelihood produced by SigNetMS. Figure 5.8 shows an example of feature selection instance that represents a model selection instance.

After defining the feature selection instance, we will traverse the search space in two ways. First, we will traverse chains from the empty set to the complete set, to understand the shape of the cost function as we increase the number of reactions. Then, we will run the Sequential Forward Search algorithm, a heuristic for the feature selection problem, to try to find a good solution for our model selection problem.

5.2.1 Defining the Feature Selection Instance

The instance we prepared is based on a Ras switch pathway. Ras represents a family of proteins that are common on signaling pathways that participate on cell growth and differentiation. Because of this participation, Ras proteins that are constantly switched on can play a part on some types of cancer. The model we consider for generating experiments, which we also call

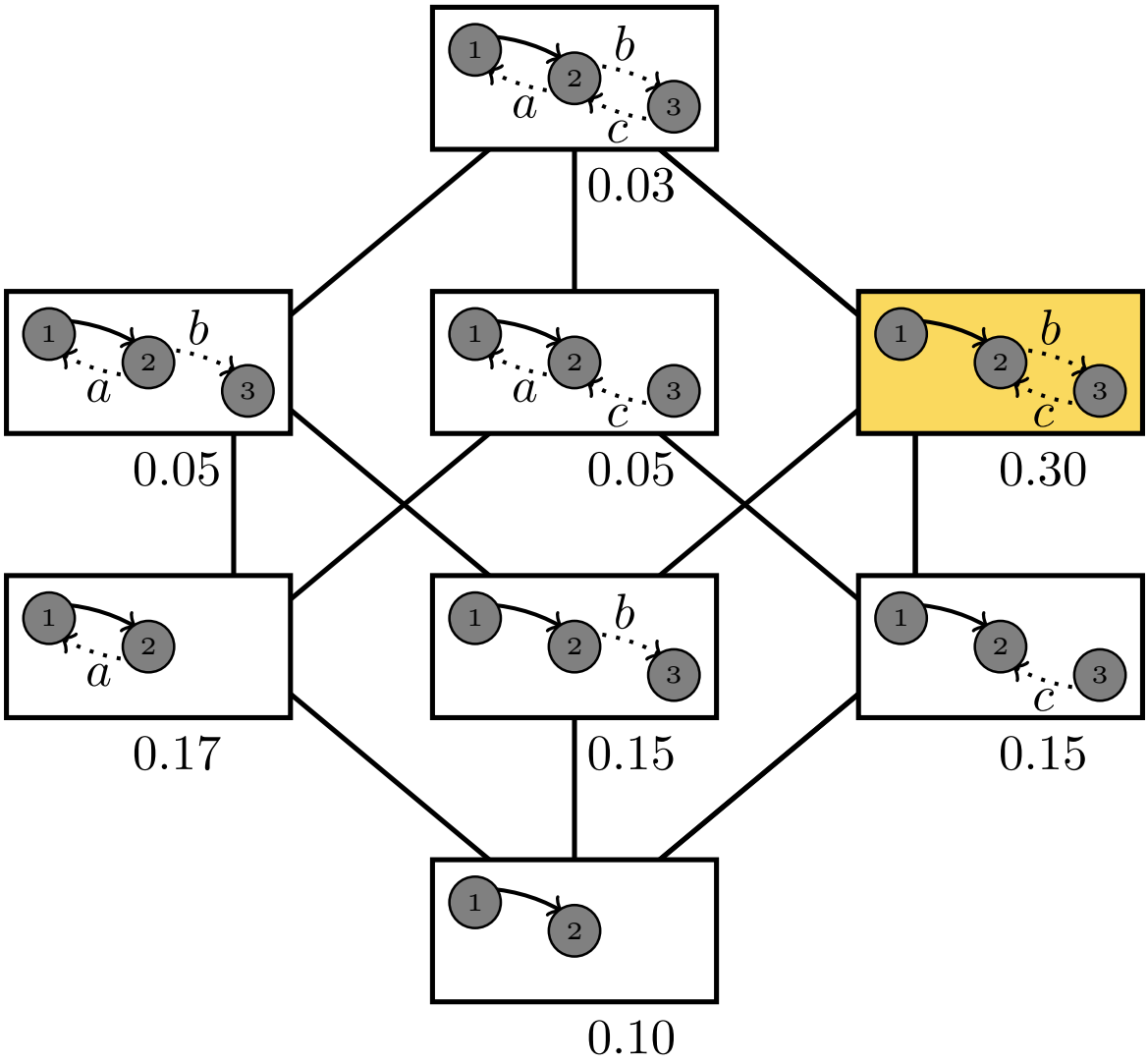


Figure 5.8: A diagram with the search space and costs of a feature selection problem that represents a model selection problem. Each rectangle represents a subset of the power set of features, were reactions from such subset are drawn with dots. Links between rectangles represent the $\subseteq$ relation; that is, two rectangles linked represent two subsets of reactions X and Y such that $X \subseteq Y$ (or $Y \subseteq X$). The set of features is composed by three reactions, $\{a, b, c\}$, inducing the search space with eight elements. The base model is composed by two chemical species, 1 and 2 and a reaction between those species, we can see this model at the base of the diagram, on the rectangle that represents the empty set (no reaction are drawn with dots). Each rectangle also represents a candidate model, which is composed by the base model plus the reactions drawn with dots. The number below each rectangle represents the score (minus one times the cost) of each model. The optimal subset is drawn in yellow and it the subset of reactions $\{b, c\}$. Note that this instance is a U-Curve instance, if we consider the cost as minus one times the score: for every chain of rectangles, the cost of the model describes a u shaped curve. As an instance, consider the chain $\emptyset, \{c\}, \{a, c\}, \{a, b, c\}$, which has respectively the costs of -0.10, -0.15, -0.30, and -0.03.

the "correct" model is shown on Figure 5.9. This model shows a pathway that decides the state of a Ras protein as switched on, represented by RasGTP, and as switched off, represented by RasGDP. Five other chemical species are present on this model, and they have the following

initial concentration: 200 for SOS; 0 for SOS_allo_RasGDP and SOS_allo_RasGTP; 900 for RasGDP; 100 for RasGTP; 200 for GEF; and 125 for GAP. Once again, we omit concentration and reaction rate units for the sake of simplificty. These chemical species interact in eight different chemical reactions:

- SOS + RasGDP $\xrightarrow{k_2}$ SOS_allo_RasGDP;
- SOS_allo_RasGDP $\xrightarrow{d_2}$ SOS + RasGDP;
- SOS + RasGTP $\xrightarrow{k_1}$ SOS_allo_RasGTP;
- SOS_allo_RasGTP $\xrightarrow{d_1}$ SOS + RasGTP;
- RasGTP $\longrightarrow$ RasGDP, with SOS_allo_RasGTP as a catalyst, and $k3cat$ and $K3m$ as catalytic constant and Michaelis constant, respectively;
- RasGTP $\longrightarrow$ RasGDP, with SOS_allo_RasGDP as a catalyst, and $k4cat$ and $K4m$ as catalytic constant and Michaelis constant, respectively;
- RasGTP $\longrightarrow$ RasGDP, with GEF as a catalyst, and $k6cat$ and $K6m$ as catalytic constant and Michaelis constant, respectively;
- RasGDP $\longrightarrow$ RasGTP, with GAP as a catalyst, and $k5cat$ and $K5m$ as catalytic constant and Michaelis constant, respectively.

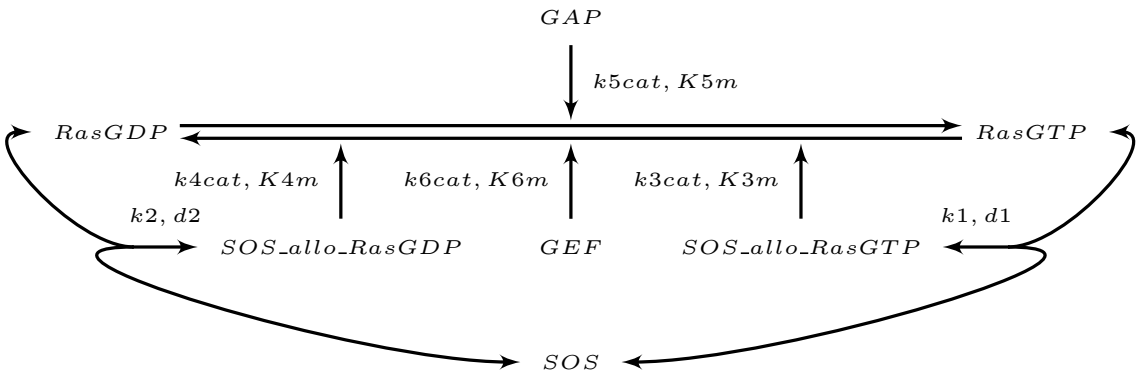


Figure 5.9: A representation of a Ras switch pathway that we consider as the correct model for our model selection experiment. This model contains seven chemical species, and eight different chemical reactions. Reversible reactions are represented with arrows poiting both directions, with reaction rate parameters over the reaction, with the forward reaction parameter first and then the parameter of the reverse reaction. Michaelis-Menten reactions are represented with parameters to the left of the arrow from starting at the enzyme, with the catalystic parameter first, and then the Michaelis constant.

We used this model to generate an artificial experiment where the concentration of activated Ras was measured at the time steps of 30, 60, 90, 120, 150, 180, 210, and 240 seconds. Figure 5.10 shows a graph of such experimental measurements. Similarly to the experiment of the previous section, those observations were created by simulating the correct model and then adding a Gaussian error of mean zero and standard deviation of 0.01. The reaction rate

parameters used to create these simulations are: k1 = $1.8e-4$, d1 = 3, k2 = $1.7e-4$, d2 = 0.04, k3cat = 3.8, K3m = $1.64e3$, k4cat = 0.003, K4m = $9.12e3$, k5cat = 0.1, K5m = $1.07e2$, k6cat = 0.01, and K6m = 1836 (once again, remember that we are omitting units for the sake of simplicity).

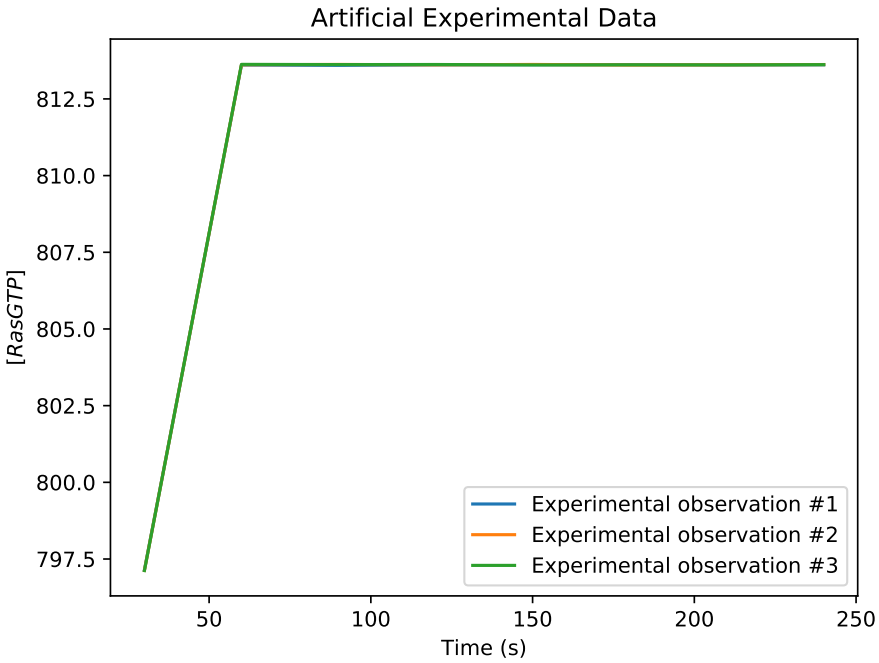


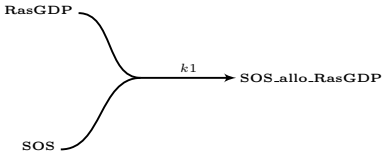
Figure 5.10: Experimental data generated with our correct model for the Ras switch pathway. There is a set of three observations being represented, however, the time points observations of [RasGTP] are close enough that the lines of each experiment overlap. The time points considered are 30, 60, 90, 120, 150, 180, 210, and 240 seconds; the lines are produced with a linear interpolation of measurements on these points.

To construct our search space, we considered as the base of our search space a trivial model, containing no reactions. For the set of features, we considered all reactions from the correct model (i.e. the correct model is present on the search space), plus two other reactions. A complete list of candidate reactions is shown of Figure 5.11. The complete list of candidate reactions is saved in a JSON file, which also stores information about rate parameters for each reaction, including its name and prior. In this experiment we defined Uniform prior distributions, using our prior knowledge about reactions. As a matter of fact, in our first approach we used Gamma distributions, however, due to numerical instabilities we decided Uniform distributions for its simplicity and restrictiveness. (TODO: should I include prior distributions on the text?)

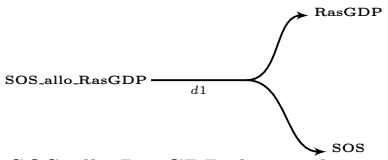
Figure 5.11 also presents an arbitrary order we choose for these reactions. This order, from (a) to (j) is useful to enumerate points of the search space, something that search algorithms can rely on when traversing the search space. With this order, we can introduce the concept of characteristic vector, which is a vector composed by boolean flags that represents a point of the search space, and, that is, a subset of features. If the i-th element of the characteristic vector is a 0, then the i-th feature is not present on the subset represented, if the i-th elemnt is 1, then this feature is present on the subset. Considering the order we choose for this Ras switch

instance, we could say that the subset represented by the characteristic vector 0101010001 has the reactions: SOS_allo_RasGDP decomplexation, SOS_allo_RasGTP decomplexation, Ras activation by SOS_allo_RasGTP, and GAP activation by RasGTP.

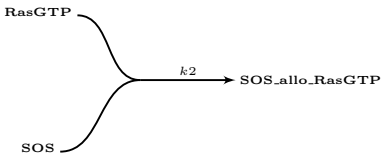
After defining our set of features, a base model, the rule for defining the model associated to a subset of features, and the cost function, we are now able to experiment solving the model selection problem as a feature selection problem. It is important to remember that our cost function, the output of SigNetMS, is computationally expensive and, since we have 10 features, the search space has the size 2^{10}. That makes it unfeasible to optimally search on the space, and therefore we choose a heuristic to search for a good solution.



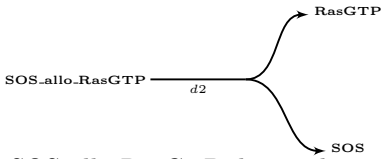
(a) SOS_allo_RasGDP complexation



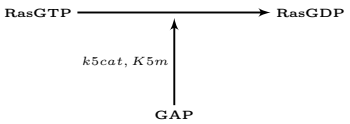
(b) SOS_allo_RasGDP decomplexation



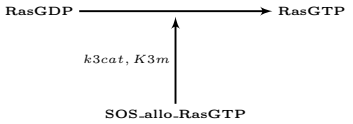
(c) SOS_allo_RasGTP complexation



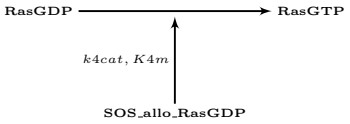
(d) SOS_allo_RasGTP decomplexation



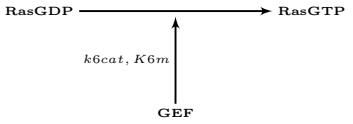
(e) Ras inactivation by GAP



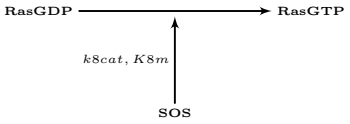
(f) Ras activation by SOS_allo_RasGTP



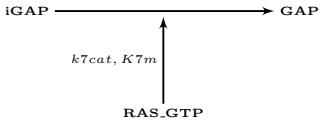
(g) Ras activation by SOS_allo_RasGDP



(h) Ras activation by GEF



(i) Ras activation by SOS



(j) GAP activation by RasGTP

Figure 5.11: All candidate reactions, or the set of features for our feature selection problem. Reactions (a) to (h) are all the reactions of the considered "correct" model.

5.2.2 Using Sequential Forward Selection to Select a Model

As discussed in the previous section, the feature selection instance we have in hands has a search space and cost function that makes it unfeasible to use an optimal algorithm of feature selection. Therefore, we use the Sequential Forward Selection (SFS) [Whi71] heuristics to search for a solution.

The Sequential Forward Selection heuristics is a greedy algorithm that starts with the empty set of features and then, for each iteration, decides to add the "best" remaining feature to the current solution. Although it is very easy to find instances in which this algorithm is not optimal (including U-Curve instances), it can generally find fairly good solutions in many cases, making it a good choice to explore spaces with unknown surfaces, such as in our case. Pseudocode 2 presents the dynamics of the SFS heuristics.

SFS (S, c)

```
1: X ← ∅
2: did_impove ← True
3: while did_improve do
4:     s* ← NIL
5:     for s ∈ S and s ∉ X do
6:         if c(X ∪ {s}) < c(X ∪ {s*}) then
7:             s* ← s
8:         end if
9:     end for
10:    if s* is NIL then
11:        did_improve ← False
12:    end if
13: end while
14: return X
```

Algorithm 2: A pseudocode representing the SFS algorithm.

Results of the search

The SFS search was conduced in a server running with an Intel Xeon E5-2690 CPU, and 252GB of RAM memory. The total time to conduce the experiment was about 26 hours, with 42 calls to the cost function, that is, 42 points of the search space were visited. The chain of subsets the algorithm traversed, from the base to the found solution, is shown on Table 5.1.

The best model found has the characteristic vector 0011010100 and had the logarithm of the marginal likelihood of 10.8. This model has the following reactions: SOS_allo_RasGTP complexation, SOS_allo_RasGTP decomplexation, Ras activation by SOS_allo_RasGTP and Ras activation by GEF.

Characteristic Vector	Score	Cost function time (seconds)
0000000000	330721.05	669.79
0010000000	245681.93	889.42
0010010000	211.98	3408.28
0011010000	5.30,	2974.71
0011010100	-10.89	3756.48

Table 5.1: The trace of the SFS run on the Ras switch instance. The characteristic vector determines the set of reactions present on the model; the Score is minus one times the marginal likelihood of the model; and, the Cost function time is the total time used by SigNetMS to calculate the marginal likelihood.

Even though the resulting subset is not the correct model, the marginal likelihood indicates that the model reasonably approximates the experimental measurements. To verify this, we

took the sample of the posterior of model parameters produced by SigNetMS and created plots of simulations using those sampled parameters. Figure 5.12 shows simulations created using different power posterior samples. We can see that as β increases, the simulation approximates better the experimental results, to the point where the simulation curves overlap with the experimental curves. That indicates that, in fact, model 0011010100 is able to reproduce the dynamics observed on experimental observations.

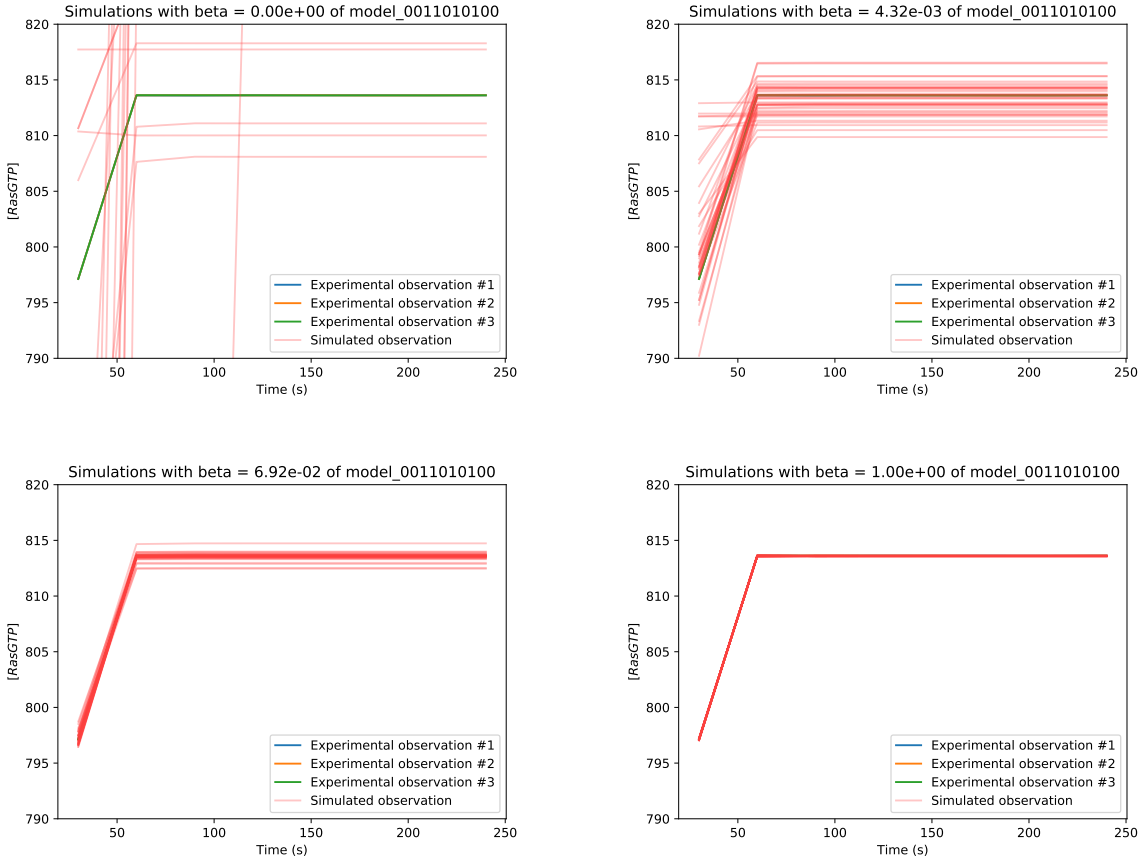


Figure 5.12: Simulations created with power posterior samples of Ras switch model with characteristic vector 0011010100*, which is the model found by the SFS search. The model parameters used to create these simulations were sampled when the SFS algorithm evaluated the cost function for the model; when the cost function was called, SigNetMS produced both the estimation of the log of the marginal likelihood and samples of different power posteriors of model parameters. There were 40 different power posteriors sampled, and we show only four of them on this figure. Since the range of the plot area is fixed, it may seem that the first figure has less lines than the other ones; that only indicate that some simulations did not appear on the plot area.*

Bibliography

[BH25] George Edward Briggs and John Burdon Sanderson Haldane. "A Note on the Kinetics of Enzyme Action". In: *Biochemical Journal* 19.2 (1925), pp. 338–339. DOI: 10.1042/bj0190338. URL: https://doi.org/10.1042/bj0190338.

[CDR99] King-Wai Chu, Yuefan Deng, and John Reinitz. "Parallel Simulated Annealing by Mixing of States". In: *Journal of Computational Physics* 148.2 (Jan. 1999), pp. 646–662. DOI: 10.1006/jcph.1998.6134. URL: https://doi.org/10.1006/jcph.1998.6134.

[FP08] N. Friel and A. N. Pettitt. "Marginal likelihood estimation via power posteriors". In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 70.3 (July 2008), pp. 589–607. DOI: 10.1111/j.1467-9868.2007.00650.x. URL: https://doi.org/10.1111/j.1467-9868.2007.00650.x.

[Gel+13] Andrew Gelman, John B. Carlin, Hal S. Stern, David B. Dunson, Aki Vehtari, and Donald B. Rubin. *Bayesian Data Analysis, Third Edition*. Chapman and Hall/CRC, Nov. 2013. DOI: 10.1201/b16018. URL: https://doi.org/10.1201/b16018.

[Han17] John T. Hancock. *Cell Signalling*. Oxford University Press, 2017. ISBN: 019965848X.

[Hin82] Alan Hindmarsh. "ODEPACK, A Systemized Collection of ODE Solvers". In: *Lawrence Livermore National Laboratory* (1982).

[Huc+03a] M. Hucka et al. "The systems biology markup language (SBML): a medium for representation and exchange of biochemical network models". In: *Bioinformatics* 19.4 (Mar. 2003), pp. 524–531. DOI: 10.1093/bioinformatics/btg015. URL: https://doi.org/10.1093/bioinformatics/btg015.

[Huc+03b] Michael Hucka, Andrew Finney, Herbert M. Sauro, Hamid Bolouri, John C. Doyle, Hiroaki Kitano, Adam P. Arkin, Benjamin J. Bornstein, Dennis Bray, Athel Cornish-Bowden, et al. "The systems biology markup language (SBML): a medium for representation and exchange of biochemical network models". In: *Bioinformatics* 19.4 (2003), pp. 524–531.

[KG00] Minoru Kanehisa and Susumu Goto. "KEGG: kyoto encyclopedia of genes and genomes". In: *Nucleic Acids Research* 28.1 (2000), pp. 27–30.

[Lie+10] J. Liepe, C. Barnes, E. Cule, K. Erguler, P. Kirk, T. Toni, and M. P. H. Stumpf. "ABC-SysBio–approximate Bayesian computation in Python with GPU support". In: *Bioinformatics* 26.14 (June 2010), pp. 1797–1799. DOI: 10.1093/bioinformatics/btq278. URL: https://doi.org/10.1093/bioinformatics/btq278.

[Lie+14] Juliane Liepe, Paul Kirk, Sarah Filippi, Tina Toni, Chris P Barnes, and Michael P H Stumpf. "A framework for parameter estimation and model selection from experimental data in systems biology using approximate Bayesian computation". In: *Nature Protocols* 9.2 (Jan. 2014), pp. 439–456. DOI: 10.1038/nprot.2014.025. URL: https://doi.org/10.1038/nprot.2014.025.

[LN+06] Nicolas Le Novere, Benjamin Bornstein, Alexander Broicher, Melanie Courtot, Marco Donizelli, Harish Dharuri, Lu Li, Herbert Sauro, Maria Schilstra, Bruce Shapiro, et al. "BioModels Database: a free, centralized database of curated, published, quantitative kinetic models of biochemical and cellular systems". In: *Nucleic Acids Research* 34.suppl_1 (2006), pp. D689–D691.

[Mar+03] P. Marjoram, J. Molitor, V. Plagnol, and S. Tavare. "Markov chain Monte Carlo without likelihoods". In: *Proceedings of the National Academy of Sciences* 100.26 (Dec. 2003), pp. 15324–15328. DOI: 10.1073/pnas.0306899100. URL: https://doi.org/10.1073/pnas.0306899100.

[Mil+09] Ron Milo, Paul Jorgensen, Uri Moran, Griffin Weber, and Michael Springer. "BioNumbers—the database of key numbers in molecular and cell biology". In: *Nucleic Acids Research* 38.suppl_1 (Oct. 2009), pp. D750–D753. DOI: 10.1093/nar/gkp889. URL: https://doi.org/10.1093/nar/gkp889.

[NR93] Michael A. Newton and Adrian E. Raftery. "Approximate Bayesian Inference with the Weighted Likelihood Bootstrap". In: *Journal of the Royal Statistical Society. Series B (Methodological)* 56.1 (1993), pp. 3–48.

[Pri+99] J. K. Pritchard, M. T. Seielstad, A. Perez-Lezaun, and M. W. Feldman. "Population growth of human Y chromosomes: a study of Y chromosome microsatellites". In: *Molecular Biology and Evolution* 16.12 (Dec. 1999), pp. 1791–1798. DOI: 10.1093/oxfordjournals.molbev.a026091. URL: https://doi.org/10.1093/oxfordjournals.molbev.a026091.

[Rei+17a] Marcelo S. Reis, Vincent Noël, Matheus H. Dias, Layra L. Albuquerque, Amanda S. Guimarães, Lulu Wu, Junior Barrera, and Hugo A. Armelin. "An Interdisciplinary Approach for Designing Kinetic Models of the Ras/MAPK Signaling Pathway". In: *Methods in Molecular Biology*. Springer New York, 2017, pp. 455–474. DOI: 10.1007/978-1-4939-7154-1_28. URL: https://doi.org/10.1007/978-1-4939-7154-1_28.

[Rei+17b] Marcelo S. Reis, Gustavo Estrela, Carlos Eduardo Ferreira, and Junior Barrera. "featsel: A framework for benchmarking of feature selection algorithms and cost functions". In: *SoftwareX* 6 (2017), pp. 193 –197. ISSN: 2352-7110. DOI: https://doi.org/10.1016/j.softx.2017.07.005. URL: http://www.sciencedirect.com/science/article/pii/S2352711017300286.

[RGG97] G. O. Roberts, A. Gelman, and W. R. Gilks. "Weak convergence and optimal scaling of random walk Metropolis algorithms". In: *The Annals of Applied Probability* 7.1 (Feb. 1997), pp. 110–120. DOI: 10.1214/aoap/1034625254. URL: https://doi.org/10.1214/aoap/1034625254.

[Sch04] I. Schomburg. "BRENDA, the enzyme database: updates and major new developments". In: *Nucleic Acids Research* 32.90001 (Jan. 2004), pp. 431D–433. DOI: 10.1093/nar/gkh081. URL: https://doi.org/10.1093/nar/gkh081.

[Szk+10] D. Szklarczyk et al. "The STRING database in 2011: functional interaction networks of proteins, globally integrated and scored". In: *Nucleic Acids Research* 39.Database (Nov. 2010), pp. D561–D568. DOI: 10.1093/nar/gkq973. URL: https://doi.org/10.1093/nar/gkq973.

[Ton+09] Tina Toni, David Welch, Natalja Strelkowa, Andreas Ipsen, and Michael P.H Stumpf. "Approximate Bayesian computation scheme for parameter inference and model selection in dynamical systems". In: *Journal of The Royal Society Interface* 6.31 (Feb. 2009), pp. 187–202. DOI: 10.1098/rsif.2008.0172. URL: https://doi.org/10.1098/rsif.2008.0172.

[TSG79] H. Towbin, T. Staehelin, and J. Gordon. "Electrophoretic transfer of proteins from polyacrylamide gels to nitrocellulose sheets: procedure and some applications." In: *Proceedings of the National Academy of Sciences* 76.9 (Sept. 1979), pp. 4350–4354. DOI: 10.1073/pnas.76.9.4350. URL: https://doi.org/10.1073/pnas.76.9.4350.

[VG07] Vladislav Vyshemirsky and Mark A. Girolami. "Bayesian ranking of biochemical system models". In: *Bioinformatics* 24.6 (Dec. 2007), pp. 833–839. DOI: 10.1093/bioinformatics/btm607. URL: https://doi.org/10.1093/bioinformatics/btm607.

[VG08] V. Vyshemirsky and M. Girolami. "BioBayes: A software package for Bayesian inference in systems biology". In: *Bioinformatics* 24.17 (July 2008), pp. 1933–1934. DOI: 10.1093/bioinformatics/btn338. URL: https://doi.org/10.1093/bioinformatics/btn338.

[VV10] Donald Voet and Judith G. Voet. *Biochemistry, 4th Edition*. Wiley, 2010. ISBN: 0121822117.

[Whi71] A.W. Whitney. "A Direct Method of Nonparametric Measurement Selection". In: *IEEE Transactions on Computers* C-20.9 (Sept. 1971), pp. 1100–1103. DOI: 10.1109/t-c.1971.223410. URL: https://doi.org/10.1109/t-c.1971.223410.

[Wit+11] U. Wittig et al. "SABIO-RK–database for biochemical reaction kinetics". In: *Nucleic Acids Research* 40.D1 (Nov. 2011), pp. D790–D796. DOI: 10.1093/nar/gkr1046. URL: https://doi.org/10.1093/nar/gkr1046.

[Xu+10] Tian-Rui Xu et al. "Inferring Signaling Pathway Topologies from Multiple Perturbation Measurements of Specific Biochemical Species". In: *Science Signaling* 3.113 (2010), ra20–ra20. ISSN: 1945-0877. DOI: 10.1126/scisignal.2000517. eprint: http://stke.sciencemag.org/content/3/113/ra20.full.pdf. URL: http://stke.sciencemag.org/content/3/113/ra20.

[Rei12] M. S. Reis. "Minimization of decomposable in U-shaped curves functions defined on poset chains – algorithms and applications". PhD thesis. University of Sao Paulo, 2012.

[Vir+20] Pauli Virtanen et al. "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python". In: *Nature Methods* 17 (2020), pp. 261–272. DOI: https://doi.org/10.1038/s41592-019-0686-2.

[Wu15] Lulu Wu. "Um método para modificar vias de sinalização molecular por meio de análise de banco de dados de interatomas". MA thesis. University of Sao Paulo, 2015.