TAMPERE UNIVERSITY OF TECHNOLOGY

ANTTI HÄKKINEN

SIMULATING EVOLUTIONARY PROCESSES OF STOCHASTIC GENETIC NETWORKS IN VOLATILE ENVIRONMENTS

Master of Science Thesis

Examiners: Professor Olli Yli-Harja,
Assistant Professor Andre Ribeiro
Examiners and topic approved in
the Computing and Electrical Engineering Faculty Council meeting
on November 9th 2011

TIIVISTELMÄ

TAMPEREEN TEKNILLINEN YLIOPISTO
Tietoliikenne-elektroniikan koulutusohjelma
ANTTI HÄKKINEN: Evoluutiomekanismien simuloiminen stokastisissa geeniverkoissa muuttuvien ympäristöolosuhteiden alla
Diplomityö, 66 sivua
Marraskuu 2011
Pääaine: Signaalinkäsittely
Tarkastajat: Professori Olli Yli-Harja, Yliassistentti Andre Ribeiro
Avainsanat: Evoluutio, geeninsäätelyverkot, stokastiset mallit

Tässä työssä esitellään simulaattori, joka mahdollistaa evoluution alla olevien solupopulaatioden mallintamisen. Soluissa olevien geeninsäätelyverkkojen dynamiikkaa mallinnetaan stokastisen simulointialgoritmin avulla yksittäisten molekyylien ja yksittäisten tapahtumien tasolla. Tämän lisäksi työssä käsitellään monimutkaisten järjestelmien, kuten biologisten geeninsäätelyverkkojen, mallinnusta. Simulaattoria voidaan käyttää geneettisten piirien mallintamiseen käyttäen geneettisiä operaattoreita, kuten lisääntymistä, mutaatiota sekä geneettisen materiaalin siirtymistä, ja solupopulaatiota voidaan mallintaa stokastisissa ympäristöolosuhteissa, jotka lisäksi muuttuvat ajan funktiona, mahdollistaen odottamattomien ympäristöolosuhteiden vaikutusten tutkimisen. Työssä esitellään myös kaksi biologisiin järjestelmiin liittyvää tutkimusta, jotka havainnollistavat simulaattorin sovellettavuutta. Ensimmäisessä esimerkissä tutkitaan odottamattomien ympäristön muutosten vaikutuksia solupopulaatioiden monimuotoisuuteen sekä soluissa tapahtuvien mutaatioiden nopeuksiin. Tämän lisäksi esimerkki paljastaa, kuinka evoluutio muodostaa monimutkaisia fenotyyppisiä jakaumia mutaatioden luonteesta riippuen. Toinen esimerkki paljastaa, että pienet muutokset evoluution reunaehdoissa saattavat johtaa hyvin erilaisten soluprossesien kehittymiseen, jotka voivat poiketa ainoastaan niissä olevan stokastisuuden määrässä. Esimerkki paljastaa myös kuinka pienet muutokset näissä prosesseissa johtavat sellaisten fenotyyppien kehittymiseen, jotka ovat huomattavasti soveltuvampia tietynlaisiin ympäristöolosuhteisiin.

ABSTRACT

TAMPERE UNIVERSITY OF TECHNOLOGY
Master's Degree Programme in Communications Electronics
ANTTI HÄKKINEN: Simulating Evolutionary Processes of Stochastic Genetic Networks in Volatile Environments
Master of Science Thesis, 66 pages
November 2011
Major: Signal Processing
Examiners: Professor Olli Yli-Harja, Assistant Professor Andre Ribeiro
Keywords: Evolution, gene regulatory networks, stochastic models

This work presents a simulator for modeling evolving cell populations. The gene network dynamics are simulated using a delayed stochastic simulation algorithm at single event and single molecule level. Moreover, modeling strategies of such complex systems are discussed. The simulator can be used to implement genetic circuits using typical genetic operators such as reproduction, mutations, and exchange and deletion of genetic material, in arbitrary fashion, and the evolving populations can be modeled in transient stochastic environments, enabling studies of the pathways of evolution in such unpredictable conditions. To demonstrate its applicability, two biologically relevant examples are presented. In the first example, the effects of environmental changes to the phenotypic diversity and mutation rates are studied. Moreover, it is shown that evolution can generate complex distributions of phenotypes, depending on the nature of the mutations. Using the second example, it is shown that small changes in the evolutionary constraints can drive a population to favor different levels of stochasticity in their cellular processes, and how small changes in the details of these processes will lead to generation of phenotypes with significant evolutionary advantage.

PREFACE

The topic of this thesis is related to the tasks that I have been performing while working as a research assistant in the Department of Signal Processing in Tampere University of Technology, mostly in early 2011. For this possibility, and for support and fruitful discussions, I want to thank my supervisors professor Olli Yli-Harja and Dr. Andre Ribeiro, and all the numerous colleagues I have had pleasure to work with.

Also, I want to thank Dr. Andre Ribeiro and Jason Lloyd-Price for the pieces of code that were borrowed from their stochastic simulator, SGN Sim.

The work presented here has resulted to the following publications in scientific journals:

> A. Häkkinen, F. G. Biddle, O.-P. Smolander, O. Yli-Harja, and A. S. Ribeiro, "Evolutionary dynamics of a population of cells with a toxin suppressor gene", in *Transactions on Computational Systems Biology XIII* (C. Priami, R. J. Back, I. Petre, and E. de Vink, eds.), vol. 6575 of *Lecture Notes in Computer Science*, pp. 1–12, Heidelberg, DE: Springer Berlin, 2011.
>
> A. S. Ribeiro and A. Häkkinen, "Probing stochastic phenotype switching as a survival strategy in fluctuating environments", *International Journal of Computational Intelligence in Control*, to appear, 2011.

and the following publication in proceedings of a conference:

> A. Häkkinen, O. Yli-Harja, and A. S. Ribeiro, "Evolving the kinetics of single gene expression", *The 11th SocBiN Conference*, Helsinki, FI, on May 10–12, 2011.

Finally, the simulator presented in this work is currently used in teaching activities in the course SGN-6236 "Modeling techniques for stochastic gene regulatory networks" at Tampere University of Technology.

CONTENTS

TERMS AND SYMBOLS

C++ – A compiled programming language

CME, chemical master equation – An equation governing the time-evolution of probabilities of states of a chemical system

DNA, deoxyribonucleic acid – A double-stranded polymer containing the genetic information of an organism

DSSA, delayed stochastic simulation algorithm – An extension of SSA allowing non-Markovian dynamics

genotype – The set of heritable traits of an individual

Lua – A lightweight scripting language

mRNA, messenger RNA – An RNA that is transcribed from DNA and used to carry genetic information

ODE, ordinary differential equation – An equation involving rates of change of variables that are functions of a single variable

phenotype – The observable set of traits of an individual

RE, reaction equation – A representation of the molecules involved in a chemical reaction

RNA, ribonucleic acid – A single-stranded nucleic acid polymer

RNAP, RNA polymerase – An enzyme that is responsible for producing RNA

RRE, reaction rate equation – An equation relating the reaction rate to different conditions, such as numbers of reactant molecules

SSA, stochastic simulation algorithm – A Monte Carlo method for simulating chemical kinetics

1. INTRODUCTION

Since prehistoric times, the fact that organisms inherit their traits from their ancestors have been exploited by humans, for example, using selective breeding to improve the yield of livestock and crop plants. However, prior to the pioneering works of Charles Darwin [1] and Gregor Mendel [2], the scientific understanding of such phenomena was completely non-existent.

The modern theory of evolution builds on traits, features that are characteristic to an individual. Some of these traits are encoded in deoxyribonucleic acid (DNA), the genetic material of the individuals. As a consequence, when an offspring is produced, the traits of the parents are propagated to their descendants. This set of heritable traits is what is collectively known as the genotype.

The change in the distribution of these traits in a population of individuals over time is what is commonly known as evolution [1]. For evolution to occur, it is fundamental that there are processes generating variability in the set of traits that a population of individuals possess, and that the traits can be inherited [3]. These two factors allow the distribution of traits to evolve over time.

The genotypic variability is generated by means such as mutations, genetic recombination, and gene flow, which constantly generate unforeseen combinations of genotypic traits. However, since the processes that only generate variability would lead to fluctuations without a drift in the genotype distribution, it is required that there are mechanisms that act on the variability and drive the direction of the evolution. A well known such mechanism is natural selection. Natural selection is the process where some of the traits provide advantage over the others, making the individuals possessing them more likely to survive or reproduce, increasing the frequency of the advantageous traits over time. Another kind of such mechanisms are provided by genetic drift effects in small populations, which are random sampling effects that can make drastic changes in the frequencies of traits, or even cause certain traits to disappear, by chance. Furthermore, mechanisms such as biased mutations have been identified, with similar effects of introducing a drift to evolution [4].

However, not all of the traits are heritable. Some traits arise from the genotypic traits as a consequence of interactions with the environment. In contrast to genotype, the observable set of traits that an individual possesses is called as its phenotype. The distinction between genotypic and phenotypic traits was noted to be of

importance [5], since instead of the genotype it is the phenotype, which determines the fitness of an individual, and on which the natural selection acts on [6], while it is the genotype that gets inherited. The influence of the environment is a fact that can make two individuals with identical genotypes to appear to be different.

The genotype is encoded in genes, which are long runs of DNA carrying the genetic information in a form of sequences of nucleobases [7]. The process of gene expression is the most fundamental process, which makes the genotype to give raise to the phenotype. This process is used by all known living organisms to generate the polymers that are essential to life from the genes encoding them. This process is known to be highly complex, consisting of a series of time-consuming subprocesses, where several steps can be modulated, the non-protein-coding regions of DNA carrying the instructions of regulatory structures. Since the phenotype is what is under selection, the dynamics of the process of gene expression is under evolutionary pressure, and since the regulatory parameters encoded in the DNA are inherited, the process of gene expression can be refined by evolution.

In live cells, the genes do not function as independent units but are organized into networks of complex pathways of interactions, where the elements are coupled in intricate manner. This is achieved by the expression products of a gene acting as regulatory molecules of other genes, or themselves, either directly or via feedforward and feedback loops of chains of regulatory motifs, such as switches [8, 9]. The details of these processes must be understood to understand how life functions.

The networks of biological systems and other complex networks often feature nonlinear cause and effect relationships between the elements. In most cases, the response is a non-linear function of the inputs, where there is a very limited range of control and outside of which the effects of the inputs quickly saturate to a constant level. The modeling of these phenomena in genetic networks have been attempted, first using boolean networks [10] to capture the threshold-like response, and later using various alternative methods such as linear and non-linear differential equations [11, 12], Bayesian networks [13], and neural networks [14], each of them possessing their advantages and disadvantages.

More recently, advances in experimental measuring techniques in molecular biology have brought available novel data giving insight about the dynamics of cellular processes with level of detail greater than the mean expression levels. Single-molecule measurements in live cells have shown that indeed correlations and stochastic fluctuations can play a major role in gene regulation [15–18], and that these features are under evolutionary pressure [19]. These findings have promoted the usage of stochastic chemical kinetic models [20–22] to accurately capture such dynamical features that were found to be present in the cellular systems, and are neglected by deterministic models. Moreover, these models allow independent regulation of both

the mean expression level and the strength of the fluctuations [23], which has also been found to be present in live cells [24].

Since many important genes are rarely expressed [25], the molecules involved in regulation of genetic circuits are present in low numbers. This makes the fluctuations and correlations in their numbers to play crucial role in the control. While it is known that substantial part of the variability in the individuals is caused by different genotypes [26], the stochasticity that obscures the mapping between genotypes and phenotypes plays a significant role [27, 28]. Moreover, an additional layer of memory is provided by the molecules that are inherited, for example, at the event of cell division, to directly make the phenotypic traits heritable.

Several studies have also investigated the effects of the environmental changes to the evolvability of a population [29]. Depending on the environmental conditions, the noisy nature of gene expression can be exploited a population of cells. From the point of view of the population, it can provide the flexibility necessary for survival that enables the cells to adapt to environments that are rapidly changing [30].

Also, it is known that the rates of mutation depend on the environmental conditions [31], can be regulated [32], and that these properties are heritable [33]. In addition to the noise inherently present in the gene expression, the mutation rates can be used to control the generation of the population variability, on which evolutionary mechanisms such as natural selection act upon. It is likely that the presence of higher mutation rates are promoted in highly transient environments, and this control can turn out to be vital to enable the survival of a species, which has to cope with multiple types of environmental conditions.

An inherent problem in making predictions about evolution is the curse of dimensionality, that is, the explosion in number of states the dimensionality increases. The number of possible evolution paths of even the simplest organisms is vastly beyond the total theoretical processing capacity on Earth [34]. Moreover, since the process of evolution is thought to be gradual, its path is likely to depend highly on the initial conditions and the imposed constraints [35], and tend towards solutions that are only locally optimal.

There are methods for studying high-dimensional problems like evolution. For two reasons, methods that are based on random sampling are well suitable for generation of possible trajectories in the problem domain. First, they avoid dealing with the inverse problem, which is usually much harder than the problem itself, and second, they only work on a randomly sampled subspace of the forward problem domain. If the forward problem can be formulated, a set of possible evolutionary trajectories can be generated, from which conclusions can be drawn, with a confidence that grows with the number of runs.

In this work, a new simulator for studying these phenomena is presented, along

with two biologically relevant examples of its usage. The dynamics of the simulations are controlled by delayed stochastic simulation algorithm, allowing the modeling of genetic circuits in a single molecule, single event level. With the simulator it is possible to model the evolution of genetic circuits, using typical genetic operators such as cell division, mutations, crossover, and gene deletions, in large populations of cells, the operators acting in either synchronous or asynchronous fashion. Simulation of cell lineages and populations over many generations is possible. Cell death and division can be based on the assessment of fitness of each individual, regarding any desired combination of features of the evolving system. Also, the environmental conditions can be modeled as arbitrarily complex stochastic chemical processes, and cells can be made in contact with an external environment and assessed in terms of their fitness regarding the interaction with the environment.

The simulator is written in C++ [36], using algorithms with optimal or low asymptotic complexities, to maximize speed and minimize memory usage allowing simulations to be done in sufficiently large scale. The simulator features a built-in Lua interpreter [37] that allows evaluation of arbitrary expressions and execution of user-provided scripts at runtime. On hardware with multiple processors or cores, parallelization is used to accelerate the simulations.

2. BACKGROUND

2.1 Simulating chemical kinetics

2.1.1 Chemical reactions

Chemical reactions are used to describe processes where a set of substances is transformed into another set of substances. These set of substances are typically called reactants and products, respectively. The reactions can be classified as spontaneous or non-spontaneous reactions, depending on, for example, if some type of energy is required for the reaction to take action. The reactions that cannot be further divided into intermediate steps, and thus describe the behavior of the system in most detail, are called elementary reactions.

A system of chemical reactions is typically represented by a set of chemical reaction equations (RE). In this representation, the reactants and products, which are the substances consumed and produced by the reaction, are placed on the left- and right-hand sides of an reaction arrow. The reaction arrow represents the type and direction of the reaction equation. Reaction equations 2.1 to 2.5 are typical examples of chemical reactions:

$$\emptyset \rightarrow A \tag{2.1}$$

$$A \rightarrow B \tag{2.2}$$

$$A + B \rightleftharpoons AB \tag{2.3}$$

$$2A \rightarrow C \tag{2.4}$$

$$A \rightarrow \emptyset \quad . \tag{2.5}$$

In equations 2.1 through 2.5 there are various substances involved, namely A, B, AB, and C. Equation 2.1 represents a spontaneous creation of the substance A, whereas equation 2.2 represents the transformation of A to B. The equation 2.3 in fact represents two reactions, first of which describes the formation of complex AB by the reaction of A and B, and the second describes the reverse reaction, that is, the disassociation of of this complex back to A and B. This kind of reaction equations are commonly used, since many chemical reactions are reversible by nature. Furthermore, equation 2.4 describes the reaction of two instances of substance A

yielding one instance of C, and equation 2.5 describes the destruction of a molecule of the chemical species A.

2.1.2 Deterministic chemical kinetics

Typically, the kinetics of the chemical reactions are represented in terms of reaction-rate equations (RRE). Reaction rate equations specify how fast the reaction occurs per unit time, as a function of the concentrations of the substances in the system. It is usually the case that only the reactants, or some of the reactants, contribute to the reaction rate of the equation.

Let the system consist of n chemical species $S_1, \cdots, S_n$, which are the different possible types of substances. A general form for the rate equation is:

$$r = k\,[S_1]^{\nu_1} \cdots [S_n]^{\nu_1} \quad , \tag{2.6}$$

where r is the reaction rate, $[S_i] \in \mathbb{N}_0$ denotes the molecular concentration of the species S_i, and k and ν_1 through ν_n are some constants that have to determined experimentally. Specifically, k is the rate constant, which usually depends on the conditions, such as the temperature and the reacting surface areas of the molecules. The sum of the coefficients ν_1 through ν_n determines the reaction order, which is often, but not necessarily taken to be a non-negative integer. In the case of elementary reactions, the factors ν_1 through ν_n are the stoichiometric coefficients, and are thus integers representing counts of whole molecules.

Most chemical kinetic models involve a set of reactions that are of order zero, one, or two. Zeroth-order reactions, that are, reactions with rate equation of the form $r = k$, are useful in the case where the system involving the reaction channel is saturated of all of the affecting reactants and the reaction rate no longer varies as a function of the number of reacting molecules.

Similarly, first-order reactions, with reaction rate equation of the form $r = k\,[S_a]$, are characterized by depending only on a concentration of a single molecular species. They are also called unimolecular reactions, and are useful representing the spontaneous formation of a molecules of a certain species through another, or in situations, where the concentrations of the other reactants do not significantly affect the overall reaction rate. The reactions of the latter form are sometimes called pseudo-first-order reactions.

Second-order reactions, or bimolecular reactions, are reactions that have a reaction rate equation of the form of $r = k\,[S_a][S_b]$. These can be further classified into reactions where $a \neq b$, and $a = b$, which is important when the concentrations are low. Again, higher order reactions can be represented as second-order reactions, where some of concentrations of the reactant molecules do not play a significant role

in determining the reaction rate.

Let us now consider a closed system of volume $|V|$, with the aforementioned n chemical species $S_1, \cdots, S_n$. We use $\boldsymbol{y}(t) = (y_1(t), \cdots, y_n(t))$ denote the concentration of the species, at time moment t, $y_i = [S_i]$ denoting that of the species S_i. The m reaction channels $R_1, \cdots, R_m$ are each characterized by the number of consumed molecules $\boldsymbol{\nu}_j^- = (\nu_{j,1}^-, \cdots, \nu_{j,n}^-)$, the number of produced molecules $\boldsymbol{\nu}_j^+ = (\nu_{j,1}^+, \cdots, \nu_{j,n}^+)$, and the reaction rate equation $r_j(\boldsymbol{y}; k_j, \boldsymbol{\nu}_j^-)$ of the form of equation 2.6.

In the infinite volume limit $|V| \to \infty$, where the numbers of the molecules tend to infinity while their concentrations approach some finite numbers $\boldsymbol{y}$, we can obtain the so-called deterministic formulation of the chemical kinetics. The derivation will be discussed briefly later. Due to law of conservation of mass, we can arrive to the conclusion that the behavior of the deterministic system in this limit is completely characterized by the following equation:

$$\frac{\partial}{\partial t}\boldsymbol{y}(t) = \boldsymbol{f}(\boldsymbol{y}(t)) = \sum_{j=1}^{m} r_j(\boldsymbol{y}(t))\,\boldsymbol{\nu}_j \quad , \tag{2.7}$$

which is a set of ordinary differential equations (ODE) of the first order. Note that, when modeling, the constants $\boldsymbol{\nu}_j = \boldsymbol{\nu}_j^+ - \boldsymbol{\nu}_j^-$ are often conflated into the reaction rate constants k_j. From this equation, it is of our interest to study the time evolution of $\boldsymbol{y}(t)$, which requires that the system of equations 2.7 is solved for $\boldsymbol{y}(t)$.

In the case where $\boldsymbol{f}(\boldsymbol{y})$ is a linear or an affine function of $\boldsymbol{y}$ and constant over time, that is $\boldsymbol{f}(\boldsymbol{y}) = \boldsymbol{A}\boldsymbol{y} + \boldsymbol{b}$, general analytical solutions are readily available regardless of the number of the molecular species involved [38]. In this case, each of the reaction channels would be either of order zero or one. However, this is often not the case, since bimolecular reactions are common in chemical systems, making $\boldsymbol{f}(\boldsymbol{y})$ at best quadratic. If the system is non-linear, and no analytical solution [39] can be found, various numerical methods can be applied to estimate $\boldsymbol{y}(t)$ for a given initial value of $\boldsymbol{y}(0)$ [40].

However, the problem with this approach might not be the tractability of solving $\boldsymbol{y}(t)$, but rather the infinite volume limit assumption. The behavior of the system in this limit is a good approximation of the behavior of the system in the case where the number of molecules of each of the species is sufficiently high, but it fails to capture crucial features of systems where some of the species are present in low copy numbers.

2.1.3 Stochastic chemical kinetics

To obtain the stochastic formulation of the chemical kinetics, we start by considering a system of n interacting chemical species $S_1, \cdots, S_n$, which interact through m elementary reaction channels $R_1, \cdots, R_m$. Furthermore, we make the assumption that at all moments of time t, the following shall hold:

1. the system is well stirred and of constant volume $|V|$, and

2. the system is in thermal equilibrium at constant temperature T.

The stochastic approach was developed to correctly account for the low copy numbers of molecules and the correlations betweens them, whose effects the deterministic approach fails to capture [41]. The formulation provided here builds on the works of Gillespie [42–45].

In mathematical terms, the first assumption is taken to mean that for each molecular species, the position $\boldsymbol{r}$ of an individual molecule is uniformly distributed in the reaction space V and independent of the positions of the other molecules. That is, the probability of finding a molecule in any subregion $V' \subseteq V$ with volume of $|V'|$ of the space is given by:

$$\pi_{\boldsymbol{r}}(\boldsymbol{r}; |V'|) = |V'| \, |V|^{-1} \quad . \tag{2.8}$$

Similarly, the second assumption is to assert that the components of the velocity of a molecule with mass m are independently normally distributed in the three-dimensional space with mean of zero and variance of $k_B T m^{-1}$, where k_B is the Boltzmann constant. Consequently, this will result to the speeds $||\boldsymbol{v}||$ being Maxwell-Boltzmann distributed, and in general, the probability that the velocity of a molecule with mass m is in the infinitesimal region of size $\delta^3 \boldsymbol{v}$ about $\boldsymbol{v}$ is obtained from the three-dimensional normal density:

$$\pi_{\boldsymbol{v}}(\boldsymbol{v}; m) = \left(2\pi k_B T m^{-1}\right)^{-3/2} \exp\left(-\frac{||\boldsymbol{v}||^2}{2 k_B T m^{-1}}\right) \delta^3 \boldsymbol{v} \quad . \tag{2.9}$$

These assumptions are expected to hold for any constant-temperature dilute gas systems, in which nonreactive molecular collisions are much frequent than the reactive ones. The advantage provided by these assumptions is that it allows us to omit the representation of positions and velocities of individual molecules, and assess them in a probabilistic manner. That is, we have converted the problem of explicit modeling of molecular dynamics into a probabilistic problem. Following this, we opt to represent the system using a state vector $\boldsymbol{x}(t) = (x_1(t), \cdots, x_n(t))$, where x_i denotes the number of molecules of the chemical species S_i in the system at time

moment t. Next, since our system is probabilistic, the fundamental question we are trying to address is not to find $\boldsymbol{x}(t)$, but rather the probability density of $\boldsymbol{x}(t)$.

We start by considering a bimolecular reaction $S_a + S_b \rightarrow \cdots$. For now, it is irrelevant if $a = b$ or not. Suppose the molecules of chemical species S_a and S_b having masses of m_a and m_b and radii of r_a and r_b, respectively. The reaction can take place when the distance between the centers of the two molecules reaches $r' = r_a + r_b$. Using the tools of classical mechanics, we can formulate this two-body problem in terms of a single body problem involving the reduced mass $m' = m_a m_b (m_a + m_b)^{-1}$ and the relative velocity vector $\boldsymbol{v}' = \boldsymbol{v}_a - \boldsymbol{v}_b$. Clearly the reaction can only take place when the center of the molecule of the chemical species S_b lies within the cylindrical volume swept by the molecule of species S_b, moving with relative velocity $\boldsymbol{v}'$ in an infinitesimally small time window of $[t, t + \delta t)$. In terms of equations 2.8 and 2.9, we can write:

$$\pi'_{R_j}(\delta t; t) = \iiint_{\boldsymbol{v}'} \pi_{\boldsymbol{r}}(\boldsymbol{r}'; ||\boldsymbol{v}'||\, \delta t\, \pi r'^2)\, \pi_{\boldsymbol{v}}(\boldsymbol{v}'; m') \tag{2.10}$$

$$= |V|^{-1} \left(8\pi^{-1} k_B T m'^{-1}\right)^{1/2} \pi\, r'^2\, \delta t \quad , \tag{2.11}$$

which denotes the probability that two molecules, one of each species, will collide in the infinitesimal time window δt around time moment t.

However, typically not all collisions are in fact reactive. It is reasonable to assume that given that a collision has occurred, the reaction occurs immediately in a probabilistic manner with some probability p_j, that is independent of δt. Thus, the probability for two molecules to collide and react, is given by the product of the probabilities of the two conditions, since the former is a condition for the latter:

$$\pi_{R_j}(\delta t; t) = p_j\, \pi'_{R_j}(\delta t; t) = c_j\, \delta t \quad , \tag{2.12}$$

where c_j is the stochastic rate constant, that is characteristic to the reaction channel R_j. Note that all the constants that are involved in equation 2.11 along with p_j, are now packed in c_j. It will turn out that it is essential that this value of c_j is a constant with respect to δt, which certainly appears to be true at least in the case of a bimolecular reaction.

Similar arguments exist to support the hypothesis that not only in the case of bimolecular reactions it is possible to express the probability $\pi_{R_j}(\delta t; t)$ in the form of equation 2.12. For example, in the case of unimolecular reaction channels $S_a \rightarrow \cdots$, we would expect the reaction mechanism to be a quantum mechanical mechanism analogous to the nuclear decay, in which case we could represent the probability $\pi_{R_j}(\delta t; t)$ in a form of $\alpha_j\, \delta t$, where α_j is a constant, both respect to δt and the volume $|V|$ of the reaction space.

The modeling of higher order reactions $S_a + S_b + \cdots \to \cdots$ is somewhat controversial. One could argue that the higher order reactions never appear as elementary reactions, but are composed of a set of lower order reactions. In this case the justification is not necessary, and the modeling should be done using the system of lower order reactions. On the other hand, under some conditions the contribution of some of the reactants vanish except up to a constant rate factor, and the reaction is therefore well approximated by a single lower order reaction with this constant rate factor. This is exactly what the earlier discussion of the ambiguity of the reaction order was about. Whatever is the case, we should expect that if such reactions are used, for a reaction of order k, there exists $c_j \propto |V|^{k-1}$, or at least a good approximate, that fulfills our criteria.

Now that the form of the probability in equation 2.12, characteristic to the reaction channel R_j, is established we proceed with the formulation. Again, we let $\boldsymbol{\nu}_j^- = (\nu_{j,1}^-, \cdots, \nu_{j,n}^-)$ to denote the number of molecules consumed by the reaction channel R_j, $\boldsymbol{\nu}_j^+ = (\nu_{j,1}^+, \cdots, \nu_{j,n}^+)$ to denote the number of molecules produced by the reaction, and $\boldsymbol{\nu}_j = \boldsymbol{\nu}_j^+ - \boldsymbol{\nu}_j^-$ denote the change in the number of molecules when the reaction takes place. Furthermore, let $h_j(\boldsymbol{x})$ be the number of combinations of the reactant molecules of the reaction channel R_j. This is provided by the combinatorial expression:

$$h_j(\boldsymbol{x}) = \prod_{i=1}^{n} \binom{x_i}{\nu_{j,i}^-} = \prod_{i=1}^{n} \frac{x_i(x_i-1)\cdots(x_i-\nu_{j,i}^-+1)}{\nu_{j,i}^-(\nu_{j,i}^--1)\cdots 1} \quad . \tag{2.13}$$

First, we will obtain the probability that exactly one reaction of kind of reaction channel R_j will occur in the infinitesimal time interval $[t, t+\delta t)$. According to equation 2.12, each of the $h_j(\boldsymbol{x})$ combinations of the reactant molecules of the reaction channel R_j has a probability of reacting in the time window of $c_j\,\delta t$. Since the reactions occur independently, due to the system being well stirred, we can write out the probability as a product of the probabilities of single reaction occurring and all the other not occurring in the time window, summed over the number of the combinations:

$$\sum_{i=1}^{h_j(\boldsymbol{x})} c_j\,\delta t\,(1-c_j\,\delta t)^{h_j(\boldsymbol{x})-1} = c_j\,h_j(\boldsymbol{x})\,\delta t + o(\delta t) \quad , \tag{2.14}$$

where $o(\delta t)$ represents a term that goes to zero faster than δt in the limit of $\delta t \to 0$.

An interesting fact can now be observed. In the light of equations 2.6 and 2.14, the relationship between the deterministic rate constant k_j and the stochastic rate constant c_j is clear. Recall that in the deterministic formulation we used concentrations $\lim_{|V|\to\infty} x_i\,|V|^{-1} = [S_i]$ in the infinite volume limit. We can now take the

limit of the probability for the reaction to occur per unit volume:

$$\lim_{|V|\to\infty} c_j\, h_j(\boldsymbol{x})\, |V|^{-1} = \\ \lim_{|V|\to\infty} c_j\, |V|^{k-1} \left(\prod_{j=1}^{n} \nu_{j,i}^{-}(\nu_{j,i}^{-} - 1) \cdots 1 \right)^{-1} \left(x_1\, |V|^{-1}\right)^{\nu_{j,1}^{-}} \cdots \left(x_n\, |V|^{-1}\right)^{\nu_{j,n}^{-}} \quad , \tag{2.15}$$

and the right hand side of the equation looks exactly like that of equation 2.6. It appears that $c_j = \left(\prod_{j=1}^{n} \nu_{j,i}^{-}(\nu_{j,i}^{-} - 1) \cdots 1\right) k_j\, |V|^{-k+1}$, where k represents the total reaction order.

Next, using the same arguments that were presented above, we write the probability that no reaction (of any reaction channel) occurs in the infinitesimal time interval $[t, t + \delta t)$:

$$\prod_{j=1}^{m} (1 - c_j\, \delta t)^{h_j(\boldsymbol{x})} = 1 - \sum_{j=1}^{m} c_j\, h_j(\boldsymbol{x})\, \delta t + o(\delta t) \quad , \tag{2.16}$$

and finally, due to equations 2.14 and 2.16, we note that the probability that more than one reaction occurs in the system during the time interval $[t, t + \delta t)$ appears to be trivially $o(\delta t)$.

The three previously introduced statements statements allow us to establish a recurrence relation between the probabilities that the system is in state $\boldsymbol{x}$ at time moment t, given that the state of the system at time moment t_0 was $\boldsymbol{x}(t_0) = \boldsymbol{x}_0$:

$$\pi_{\boldsymbol{x},t}(\boldsymbol{x}, t + \delta t; \boldsymbol{x}_0, t_0) = \sum_{j=1}^{m} \pi_{\boldsymbol{x},t}(\boldsymbol{x} - \boldsymbol{\nu}_j, t; \boldsymbol{x}_0, t_0) \left(a_j(\boldsymbol{x} - \boldsymbol{\nu}_j)\, \delta t + o(\delta t) \right) + \\ \pi_{\boldsymbol{x},t}(\boldsymbol{x}, t; \boldsymbol{x}_0, t_0) \left(1 - \sum_{j=1}^{m} a_j(\boldsymbol{x})\, \delta t + o(\delta t) \right) + o(\delta t) \quad , \tag{2.17}$$

where $a_j(\boldsymbol{x}) = c_j\, h_j(\boldsymbol{x})$ is the propensity function. The first term of the equation 2.17 is the contribution from the fact that a single reaction occurred during the time interval, the second is the contribution from if no reaction occurred, and the third term is the contribution from other number of reactions occurring.

Now, subtracting $\pi_{\boldsymbol{x},t}(\boldsymbol{x}, t; \boldsymbol{x}_0, t_0)$ from both sides, dividing by δt, and taking the limit $\delta t \to 0$ will yield:

$$\frac{\partial}{\partial t}\,\pi_{\boldsymbol{x},t}(\boldsymbol{x}, t; \boldsymbol{x}_0, t_0) = \sum_{j=1}^{m} a_j(\boldsymbol{x} - \boldsymbol{\nu}_j)\,\pi_{\boldsymbol{x},t}(\boldsymbol{x} - \boldsymbol{\nu}_j, t; \boldsymbol{x}_0, t_0) - \sum_{j=1}^{m} a_j(\boldsymbol{x})\,\pi_{\boldsymbol{x},t}(\boldsymbol{x}, t; \boldsymbol{x}_0, t_0) \quad , \tag{2.18}$$

which is what is commonly known as the chemical master equation (CME). As it appears from equation 2.18, the CME is a set of first-order ODEs describing the time evolution of the probability of the state space of the system.

It was also mathematically proved that the deterministic formulation can be obtained as the limiting case of the stochastic formulation, where the number of molecules and the reaction volume approach infinity, the concentrations converging to some finite values [46]. By multiplying both sides of equation 2.18 by $\boldsymbol{x}$, and summing over all values of $\boldsymbol{x}$, it follows that:

$$\frac{\partial}{\partial t}\,\mathbb{E}[\,\boldsymbol{x}(t)\,|\,\boldsymbol{x}_0, t_0\,] = \sum_{j=1}^{m} \mathbb{E}[\,a_j(\boldsymbol{x}(t))\,|\,\boldsymbol{x}_0, t_0\,]\,\boldsymbol{\nu}_j \quad , \tag{2.19}$$

where $\mathbb{E}[\cdot]$ denotes the expected value. By further dividing by $|V|$ and taking the limit $|V| \to \infty$ it can be asserted that the expected value of $\boldsymbol{y}(t)$ is indeed that of the right hand side of equation 2.7. A more complicated proof, involving either Chebyshev inequality or central limit theorem [46, 47], can be then used at this limit to show that the fluctuations and correlations in the molecular numbers will vanish, from which it can be concluded that the distribution becomes degenerate and hence the process deterministic, and in fact is exactly that described in equation 2.7.

2.1.4 Stochastic simulation algorithm

Since obtaining an analytical solution to the CME is sometimes intractable, numerical methods have been proposed to address this problem. One such method is what is commonly called stochastic simulation algorithm (SSA) [42, 44].

The SSA builds on the stochastic formulation of the chemical kinetics, that is, it numerically simulates the underlying Markov process that the CME describes. The approach used to circumvent the intractability of the CME is random sampling. A single simulation of SSA will execute explicitly a single possible sequence of reactions, yielding a single trajectory in the possible state space of the system with the appropriate probability density. The resulting algorithm is simple, and it is rather inexpensive to calculate a single trajectory on a digital computer. However, as it

is usually of interest to estimate the probability density described by the CME, or some other related density, one often needs to run large number of simulations to obtain the sampled distribution of this density, which can become expensive.

The key for generating the trajectories according to the CME is not the probability density of the time evolution of the number of molecules in the system (equation 2.17). Instead, it is yet another density, namely, the density that the next reaction in the system will occur in a infinitesimal time interval $[t+\tau, t+\tau+\delta\tau)$, and will be a reaction R_μ. This density can be obtained by considering the time interval $[t, t+\tau)$ divided into a set of time windows of size $\varepsilon = \tau k^{-1}$. It must be that no reaction occurs in each of these time windows, and finally the reaction R_μ occurs in the time window $[t+\tau, t+\tau+\delta\tau)$. With the knowledge provided by equation 2.16, we can now write:

$$\pi_{\tau,\mu}(\tau, \mu; \boldsymbol{x}, t)\, \delta\tau = (1 - a(\boldsymbol{x})\varepsilon + o(\varepsilon))^k \, (a_\mu(\boldsymbol{x})\, \delta\tau + o(\delta\tau)) \quad , \tag{2.20}$$

where $a(\boldsymbol{x}) = \sum_{j=1}^{m} a_j(\boldsymbol{x})$. Dividing by $\delta\tau$, and taking the limit $\delta\tau \to 0$ we will obtain:

$$\begin{aligned} \pi_{\tau,\mu}(\tau, \mu; \boldsymbol{x}, t) &= (1 - a(\boldsymbol{x})\varepsilon + o(\varepsilon))^k \, a_\mu(\boldsymbol{x}) && (2.21) \\ &= \left(1 - k^{-1}\left(a(\boldsymbol{x})\tau + o(\varepsilon)\varepsilon^{-1}\tau\right)\right)^k a_\mu(\boldsymbol{x}) \quad , && (2.22) \end{aligned}$$

for which we take the limit $k \to \infty$, yielding:

$$\begin{aligned} \pi_{\tau,\mu}(\tau, \mu; \boldsymbol{x}, t) &= a_\mu(\boldsymbol{x}) \exp(-a(\boldsymbol{x})\tau) && (2.23) \\ &= a'_\mu(\boldsymbol{x})\, a(\boldsymbol{x}) \exp(-a(\boldsymbol{x})\tau) \quad , && (2.24) \end{aligned}$$

where $a'_\mu(\boldsymbol{x}) = a_\mu(\boldsymbol{x})\, a(\boldsymbol{x})^{-1}$ is the normalized propensity of reaction μ, and the rest can be recognized to be the probability density of an exponential distribution, with a rate parameter of $a(\boldsymbol{x})$.

Now, based on equation 2.24, the observations fundamental to the SSA can be made. The next reaction is characterized by the pair (τ, μ), where the time when the next reaction occurs $\tau \sim \mathcal{E}(a(\boldsymbol{x}))$ and the choice of the next reaction $\mu \sim \mathcal{M}(a'_1(\boldsymbol{x}), \cdots, a'_m(\boldsymbol{x}))$ are independent random variables with exponential and multinomial densities, respectively. To generate the trajectories, all we now need is to generate pairs of random numbers according to these densities. This can be done using so-called inverse transform sampling, that is, for a pair of continuous uniform random numbers $r_1, r_2 \sim \mathcal{U}[0, 1)$ in the semi-open unit interval, we perform

the following transformations:

$$\tau = -a(\boldsymbol{x})^{-1} \ln(1 - r_1) \tag{2.25}$$

$$\mu = \mu' \quad \text{such that} \quad \sum_{j=1}^{\mu'-1} a_j(\boldsymbol{x}) \leq r_2\, a(\boldsymbol{x}) < \sum_{j=1}^{\mu'} a_j(\boldsymbol{x}) \quad , \tag{2.26}$$

which are simply the inverse functions of the respective cumulative distribution functions of the required distributions.

Now we are ready to outline the full algorithm. Recall that we are provided with the initial moment of time t_0, the initial number of molecules of each species $\boldsymbol{x}_0$, and the characteristics of each of the m reaction channels in terms of the propensities $a_j(\boldsymbol{x})$ and the updates in the number molecules $\boldsymbol{\nu}_j$. The algorithm is executed by following the steps:

1. initialize the time $t \leftarrow t_0$ and the system state $\boldsymbol{x} \leftarrow \boldsymbol{x}_0$
2. evaluate each $a_j(\boldsymbol{x})$ and their sum $a(\boldsymbol{x})$, which depend on the system state $\boldsymbol{x}$
3. generate τ and μ according to equations 2.25 and 2.26, respectively
4. perform the reaction R_μ by letting $t \leftarrow t + \tau$ and $\boldsymbol{x} \leftarrow \boldsymbol{x} + \boldsymbol{\nu}_\mu$
5. go back to step 2

The above scheme will yield a time series $\boldsymbol{x}(t)$ in the state space of the system, where $\boldsymbol{x}(t)$ changes at discrete points, which can be recorded right after performing step 4. As discussed, the trajectory obtained is an exact realization of the Markov process described by the CME, which can be seen since it was derived using the same principles with no additional approximations. In particular, one should note that the time step τ is not an approximation parameter which is typically found in ODE solvers, but a realization of a single time interval with the distribution that was shown to be appropriate.

2.1.5 Delayed stochastic simulation algorithm

One major difficulty with the simulations using SSA is that we must describe the system using elementary reactions, or at least using reactions that appear to behave like the elementary ones to a certain degree.

Especially in biological context many processes, such as transcription, translation, and degradation of their products, are compound multi-step processed that, for example, involve sequential assembly of long molecules. Since these processes are inherently slow, their effects cannot be ignored. Moreover, due to central limit theorem, we would expect such multistage processes to exhibit Gaussian statistics

instead of the exponential, which were found to apply for the elementary reactions used in the SSA. However, such a process could be modeled as a set of sequential reactions, but the explicit modeling would require knowledge about the details of each elementary step, and the complexity and number of free parameters in the model would explode.

To address this problem, several modifications to the SSA have been proposed to account for this kind of semi-Markovian dynamics. The approach presented here was first proposed by Bratsun et. al. [48], and generalized for multiple delayed products by Roussel and Zhu [49]. Let us consider a sequence of elementary reactions of the form:

$$A \xrightarrow{c_0} I_1 \xrightarrow{c_1} \cdots \xrightarrow{c_{n-1}} I_n \xrightarrow{c_n} B \quad , \tag{2.27}$$

where A is transformed almost surely into a B through n intermediates $I_1, \cdots, I_n$, which supposedly do not play any other role in the system. This could be readily represented by the non-elementary reaction $A \to B$, but we must ascertain that the dynamics are preserved. From 2.25 we know that the time intervals $\tau_1, \cdots, \tau_n$ are each independent and follow the exponential distributions $\mathcal{E}(c_1), \cdots, \mathcal{E}(c_n)$, respectively. This can be converted into a reaction of the form:

$$A \xrightarrow{c_0} B(\tau_B) \quad , \tag{2.28}$$

where the parenthesized τ_B denotes the time delay, that it takes after the reaction has occurred, that the product B is introduced into the system.

Now we need to determine the distribution of τ_B, whose probability density can be obtained by convolving the probability densities of the individual distributions. The exact result is rather intricate, but good approximations exists, which can significantly speed up the simulation and reduce the dimensionality of the model. For example, due to central limit theorem, when n is sufficiently large, the time delay can be approximated by the normal distribution $\mathcal{N}(\sum_{i=1}^{n} {c_i}^{-1}, \sum_{i=1}^{n} {c_i}^{-2})$ with mean of $\sum_{i=1}^{n} {c_i}^{-1}$ and variance of $\sum_{i=1}^{n} {c_i}^{-2}$. Note that this approximation is just a convenience and nothing prevents us to obtain the exact distribution for τ_B, or to determine the distribution completely by experimental measures.

The delayed stochastic simulation algorithm (DSSA), which is an extension to the SSA presented earlier, can be outlined as follows. The steps that are equivalent to the original algorithm are represented in cursive:

1. *initialize the time* $t \leftarrow t_0$ *and the system state* $\boldsymbol{x} \leftarrow \boldsymbol{x}_0$
2. *evaluate each* $a_j(\boldsymbol{x})$ *and their sum* $a(\boldsymbol{x})$*, which depend on the system state* $\boldsymbol{x}$
3. *generate* τ *and* μ *according to equations 2.25 and 2.26, respectively*

4. if there are delayed products to be released in the time interval $[t, t+\tau]$
   (a) release the delayed product S_i with least t' by letting $t \leftarrow t'$ and $x_i \leftarrow x_i + 1$
   (b) go back to step 4
5. perform the reaction R_μ by letting $t \leftarrow t + \tau$ and $\boldsymbol{x} \leftarrow \boldsymbol{x} + \boldsymbol{\nu}'_\mu$, where $\boldsymbol{\nu}'_\mu$ is change in number of molecules without the delayed products
6. for each delayed product S_i, delay the release until $t' = t + \tau$, where τ is the time delay
7. *go back to step 2*

Note that in step 6 the time delay τ can be drawn from arbitrary distribution, as necessary. This algorithm also allows different products to have different time delays. Moreover, by comparing the DSSA algorithm to the algorithm of the original SSA, it can be verified that when no delayed products are present, the DSSA algorithm is exactly equivalent to that of the original SSA.

2.2 Modeling genetic circuits

2.2.1 Modeling transcription and translation

The control of features such as timing, location, and total rate of gene expression of the fundamental genes is crucial to the survival of the organisms. The key processes determining the dynamics of gene expression are the processes of transcription and translation. In transcription, the RNA polymerase (RNAP) reads the DNA, assembling a messenger RNA (mRNA), which is a single-stranded copy of the gene. This mRNA is used as a template, which in turn can be read by ribosomes to assemble the genetic products [7]. The details of these processes play a role in determining not only the rate [50, 51] at with the gene is be expressed, but the diversity and the fluctuations [52], and other dynamical features such as burstiness [20, 51, 53–55]. Due to this, the modeling of gene expression often focuses on modeling the details of these two processes.

Since the genetic products often exist in low copy numbers [56–58], the effect of fluctuations and correlation their levels cannot be neglected [20]. Moreover, these processes are inherently complex multi-step processes [59], involving steps such as binding and unbinding of various regulatory molecules, assembly of complexes, diffusion of the assembling molecules through a nucleotide chains of various lengths, and maturation and folding of the produced polymers to their appropriate

three-dimensional structure [7]. This means that they are not only the most time-consuming subprocesses of the process of gene expression, thus limiting the total rate, but they also determine the variability and correlations in the gene expression that are resulted in the overall process.

The models of stochastic chemical kinetics have been found to successfully capture the features that are present in the cellular processes [60]. Arguably the simplest stochastic model for gene expression is the model of zeroth-order creation of the gene expression products:

$$\emptyset \xrightarrow{c_p} p \quad , \tag{2.29}$$

where p represents the gene expression product, typically a protein, and c_p is the rate of production. This has been shown to well describe the measurements of single-molecule dynamics in live cells under certain conditions, such as the production of mRNA in bacterial genes with slow rates [61].

The lifetime of these gene expression products is often limited, and has been shown to be described by a first-order degradation process [57]:

$$p \xrightarrow{d_p} \emptyset \quad , \tag{2.30}$$

where d_p is the degradation rate, analogous to the nuclear decay rate. The gene expression products are often quite short-lived [57], which is a factor that contributes to the small mean levels observed.

If the reactions 2.29 and 2.30 are coupled, the CME can be readily solved to find that, for the limit $t \to \infty$, the protein numbers $p \sim \mathcal{P}(c_p\, {d_p}^{-1})$ follow a Poisson distribution, with rate parameter $c_p\, {d_p}^{-1}$. The Poisson distribution is able to capture one fundamental feature of gene expression, namely, the low copy number or Poisson noise. It is characteristic to the Poisson distribution that the noise, that is the relative uncertainty, decreases as a function of the mean, resulting in strong fluctuations for small mean levels.

A more detailed model of this process is a single step transcription-translation model with delayed products [62] that is represented by the reaction:

$$P + R \xrightarrow{c_p} P(\tau_P) + R(\tau_R) + b_p\, p(\tau_p) \quad , \tag{2.31}$$

where $[P] \in \mathbb{Z}_2$ represents the promoter being occupied ($[P] = 0$) or free ($[P] = 1$), R is an RNA polymerase, and p is the resulting protein. The value of b_p determines how many proteins are produced from a single mRNA, and can be used to tune the burstiness of gene expression. The reaction rate c_p is determined based on the binding affinity of the RNAPs to the promoter region, as well as their diffusion

along the nucleotide strand, searching for the transcription start site. The promoter delay τ_P represents the time it takes to clear the promoter region after starting the transcription, which is necessary before another transcription event can start, whereas the delay on the RNAP additionally includes the time intervals such as that of transcriptional elongation, after which the RNAP is released and ready for another transcription event. The time delay τ_p on the protein includes all the events, starting from the transcription initiation, including translation, to the protein folding and maturation, which are necessary for the protein to become functional. Due to their physical meaning, it is expected that $\tau_P < \tau_R < \tau_p$.

The natural extension of this model is to separate the steps of transcription and translation. This is represented by the following model [63], where the first equation models the transcription, and the second the translation:

$$P + R \xrightarrow{c_m} P(\tau_P) + R(\tau_R) + m(\tau_m) \tag{2.32}$$

$$m + r \xrightarrow{c_p} m(\tau'_m) + r(\tau_r) + p(\tau_p) \quad , \tag{2.33}$$

where P, R, and p denote the promoter region, RNAP, and protein, as in equation 2.31. Moreover, m is used to represent the messenger RNA, to which the ribosome r binds to, initiating translation. Here, no burstiness parameter b_p is required, since in the production of multiple proteins from a single mRNA is inherent, and the burstiness can be tuned using the other parameters of the model.

In the model represented by reaction equations 2.32 and 2.33, the parameter c_m acts in the role of the c_p of the previous model, and the transcriptional parameters τ_P and τ_R have the respective roles of τ_P and τ_R of the previous model. However, the value of τ_m now represents time it takes form a piece of mRNA where the ribosomes r can bind, after the transcription has initiated. In prokaryotic gene expression this time is equivalent to the time of forming a piece of mRNA which contains the ribosome binding site, whereas in eukaryotes the messenger RNA needs to be fully produced and transported into the nucleus for translation to initiate. The translational parameters c_p, τ'_m, τ_r, and τ_p act similarly to their transcriptional counterparts: c_p involves the binding of the ribosomes and their diffusion to find the translation initiation site, τ'_m is the time delay after which the ribosome binding site of the mRNA is available for another initiation of translation event, τ_r the delay that the corresponding ribosome is available for another translation event, and τ_p includes each of the translational steps, including translational elongation, with the addition of post-translational steps such as protein folding, maturation, and possible transportation to its active site.

Also, even more detailed models have been presented [49, 64, 65], allowing the study of more fine grained details of the steps involved in the processes of transcrip-

tion and translation. This has been motivated by the single-molecule measurements in live cells [54, 61]. These models account for the smallest known details, such as the individual steps of formation of the complexes in transcription and translation initiation [59], premature termination [66] of the transcription and tranlation process, stepwise elongation nucleotide-by-nucleotide, or codon-by-codon, with features such as arrests [67], pauses [68, 69], and editing and backtracking [67].

2.2.2 Gene regulatory networks

In real world, genes form complex circuits. The products of gene expression can bind to the promoter regions of other genes, or to that of the same gene, causing regulation of the expression of the downstream genes. This interactive behavior gives rise to pathways of genetic networks possibly consisting of thousands of elements with intricate control structures including multiple feedback and feedforward loops.

It is often the case that several molecules bind to a promoter region of a gene, acting, for example, as a transcription factor or a cofactor. Typically, the proteins acting as transcription factors are produced by other genes in the same cell, and the cofactors are non-protein compounds that act in cooperation with proteins catalyzing the expression of the gene. The sites that these molecules bind to are called operator sites. Recall that if the molecules binding to the promoter region are present in low copy numbers, their effects cannot be conflated to the reaction rate, but we must explicitly consider them in our models.

The earliest models of genetic networks were boolean networks, proposed by Kauffman [10]. In these models, the boolean variables of the nodes represent genes being on or off, depending on the value of the variable. Each of the nodes is then updated according to some boolean function that depends on the states of a subset of these nodes. Despite their relative simplicity and abstractness, Boolean networks have been shown to exhibit rich dynamics with features typical to non-linear complex systems, such as oscillations, attractors, and hysteresis [70–72].

Quite often, many of the regulatory pathways are not known in full detail, or the details are so intricate that their modeling and simulation is not possible. In these cases the models are often reduced to a certain degree of detail. For these reasons, the study of boolean networks in the context of genetic networks concentrated on studying ensembles of random networks, that is, networks whose topology and boolean networks were generated randomly.

The boolean functions that are involved in the boolean networks can be used in an equivalent manner in stochastic networks [62], where the effects of low-copy number of the regulatory molecules can be appropriately accounted for. Let us consider a network consisting of k genes. We use $P_{i,(s_1,\cdots,s_k)}$ to denote the state of a promoter region of the gene i, where $s_j = 1$ if and only if a regulatory molecule produced by

gene j is bound to it, and $s_j = 0$ otherwise. It is evident that there is 2^k promoter states that we need to consider. By using these different states of promoters, we can model the state transitions as appropriate, and consider different set of parameters for the gene expression model depending on the state.

The expression of each gene is modeled by their appropriate reactions. In addition to the gene expression we should account for the degradation of its product(s), as is usually the case in biological systems. For example, based on the model in reaction equation 2.31, we have:

$$P_{i,\boldsymbol{s}} + R_i \xrightarrow{f_i(\boldsymbol{s})\, c_{p_{i,\boldsymbol{s}}}} P_i(\tau_{P_{i,\boldsymbol{s}}}) + R_i(\tau_{R_{i,\boldsymbol{s}}}) + p_i(\tau_{p_{i,\boldsymbol{s}}}) \tag{2.34}$$

$$p_i \xrightarrow{d_{p_i}} \emptyset \quad , \tag{2.35}$$

where that P, R, and p of the original model is replaced by $P_{i,\boldsymbol{s}}$, R_i, and $p_{i,\boldsymbol{s}}$, respectively, to make a distinction between the different genes and the different states of their promoters. Moreover, $f_i(\boldsymbol{s}) : {\mathbb{Z}_2}^k \mapsto \mathbb{Z}_2$ is the boolean function of the gene, specifying the genes, whose products are required to bind or not to bind to the promoter region of the gene i to make it active, and $\boldsymbol{s} = (s_1, \cdots, s_k) \in {\mathbb{Z}_2}^k$ is just an abstract representation of the 2^k different promoter states.

Note that the kinetic parameters of each gene can be selected to be different. The RNAPs in the vicinity of the ith gene can be modeled separately for each gene as in reaction equation 2.34. Alternatively, the RNAPs can be either shared by the genes that are closely located, or the limit $[R] \to \infty$ can be taken, in case the RNAPs are abundant, in which case the values of $c_{p_{i,\boldsymbol{s}}}$ and $\tau_{R_{i,\boldsymbol{s}}}$ become irrelevant.

The coupling between the different genes is represented by a set of reactions of the form:

$$P_{i,(s_1,\cdots,s_{j-1},0,s_{j+1},\cdots,s_k)} + p_j \underset{a_{j,i}\, c_{u_{i,\boldsymbol{s}}}}{\overset{a_{j,i}\, c_{b_{i,\boldsymbol{s}}}}{\rightleftharpoons}} P_{i,(s_1,\cdots,s_{j-1},1,s_{j+1},\cdots,s_k)} \tag{2.36}$$

where the product of gene j bind to the promoter of the gene i. The parameters $c_{b_{i,\boldsymbol{s}}}$ and $c_{u_{i,\boldsymbol{s}}}$ control the rate of the binding and unbinding of this molecule in that promoter region. Moreover, $a_{j,i} = [\boldsymbol{A}]_{j,i}$ is the element of the adjacency matrix $\boldsymbol{A} \in {\mathbb{Z}_2}^{k \times k}$ of the genetic network, that is, $a_{j,i} = 1$ if and only if the products of gene j regulate the expression of gene i.

The scheme presented above will lead to the exponential explosion number of promoter state transitions. However, we can omit the reactions which have a zero reaction rate, since consequently they can never occur. Typically, either the number of regulatory molecules of a given promoter is taken to be small, and/or most of the state transitions are not possible. For example, this can be due to some of the elements being cofactors, binding to elements that are already bound, or due to

conformational changes in the promoter region, which may require that some the regulatory molecules have a specific order of binding.

The modeling of stochastic genetic networks can be exemplified by considering a k-gene repressilator, that is, an genetic circuit with ring topology. Such circuits are found in cells and they are used in building clock-like subcircuits [9, 73], that regulate other parts of genetic networks. In this circuit, products of gene $i-1 \,(\mathrm{mod}\, k)$ are used to repress the expression of gene i. By using the scheme presented above, this can be represented by a set of following reactions for each gene:

$$P_i + R \xrightarrow{c_{p_i}} P_i(\tau_{P_i}) + R_i(\tau_{R_i}) + p_i(\tau_{p_i}) \tag{2.37}$$

$$p_i \xrightarrow{d_{p_i}} \emptyset \tag{2.38}$$

$$P_i + p_{i-1 \,(\mathrm{mod}\, k)} \underset{c_{u_i}}{\overset{c_{b_i}}{\rightleftharpoons}} P_i' \xrightarrow{d_{p_i}} P_i \quad , \tag{2.39}$$

where P_i represents the unrepressed state of the promoter of gene i and P_i the repressed. The reaction equations 2.37 correspond to those of 2.34, the first part of 2.39 to that of 2.36, and 2.38 and the latter part of 2.39 to those of 2.35.

It is visible that this scheme of building stochastic networks is an extension built on the methods used in Boolean network models. The adjacency matrix $\boldsymbol{A}$ is used to describe the network topology, and the boolean function $\boldsymbol{f}(\boldsymbol{s}) = (f_1(\boldsymbol{s}), \cdots, f_k(\boldsymbol{s}))$ controls the logic that drives the network dynamics, just as they do in the context of boolean networks. If the networks are generated in a random fashion, one can study the network in terms of ensembles of random stochastic networks with given properties. Such network generators are already available [74].

In the presented modeling strategy, higher-order regulatory elements such as multimerization of the protein products were not considered. However, instead of the protein products directly binding to the operator sites, an additional set of reactions can be introduced to form the multimers that are used to control the expression of the genes, in cases where such control is appropriate.

2.2.3 Evolving dynamics

The Darwinian theory of origin of species is based on heritable variability and natural selection [1]. The variability enables some of the individuals to be more suited to the environmental conditions under which they are living. This score of having evolutionary advantageous set of traits is called fitness. Natural selection, for example, operates on the variable population by affecting the fecundity and survival of the individuals based on their fitness, providing selective advantage to the individuals with higher fitness.

The fitness of an individual is determined by its phenotype through the interac-

tions with its environment. The phenotype is largely determined by the genotype [26], which is the set of heritable traits, that is transferred from previous generations. Some traits might be advantageous in a certain kind of environment, but a disadvantageous in another. Additionally, most organisms do not live in a static environment, but are subject to multiple different environmental conditions, either caused by the fluctuating and transient environmental conditions, or by the population growing and migrating to places that are subject to other kinds of environments. A population of individuals, evolved to be optimal in certain environmental conditions, might go extinct, when a perturbation in the environmental conditions is triggered.

Different organisms are known to possess different ways of coping with environmental changes. Unpredictable environmental changes can be countered by having high population diversity, which allows survival of sufficient subset of the population from the attacks of various kinds of threats. Consequently, the variability in the genotypes and phenotypes of a population of organisms is a crucial factor determining the adaptability and the robustness of the survival of the population.

The fitness is rarely considered to be proportional to the lifetime of an individual, but instead, it is associated to their capability to reproduce. For example, an equally fit individual with shorter lifetime than its competitors, but being able to produce larger offspring, has better chance of spreading its gene pool down to the successive generation of individuals. This emphasizes the fact that not even the best traits are evolutionary preserved if they cannot be inherited.

In addition to natural selection, there are mechanisms that can change the frequencies of the traits in a population. Genetic drift is the process where the frequency of certain trait changes due to the effects of random sampling. In populations with low number of individuals, it is not that unlikely that the population adopts a trait that is not the most advantageous one just by chance. It is also possible that a trait can disappear by chance, even though it would have been advantageous, if it was carried only by a few individuals who got eliminated, for example, by accident. This effect is only present in sufficiently small populations, the effects becoming less drastic as the number of the individuals increases.

There are also a number of ways by which heritable variability can be generated in a population. The most common source of this variability are mutations, which are spontaneous changes in the genetic sequence of an organism. Mutations are inherently random, but they might be sometimes biased towards certain effects [4]. Some of the mutations may be deleterious, that is, they lead to the death of the individual. On the other hand, some of them provide the individuals with advantageous traits, and introduce unforeseen combinations of traits in the population. There are also neutral mutations that do not affect the phenotype of the individual.

Since harmful mutations are common, there are mechanisms to limit the rates

of mutations [75]. Such mechanisms include error detection and repairing of the DNA, detection of transcriptional and translational errors, as the nucleotide strand is being copied, and elimination of erroneous products. There are estimates that the fraction of mutations that are harmful might be as high as 70% [76]. One way of controlling the effective rates of mutations is the regulation of such mechanisms. The optimal mutation rates depend on the environmental conditions [32], and such control is likely to provided evolutionary advantage if the environmental conditions are changing.

Another source of genetic variation is the process of genetic recombination. In this process, the two or more pieces of DNA are split to smaller runs, and these runs are used to recreate a piece of DNA, which is a combination of the smaller runs from multiple sources. Coupled with natural selection that is used to select the evolutionary advantageous combination of traits, this is effectively a mechanism for generating the best of both worlds.

Some species are inherently clonal and some are not [77]. Species where different sexes exist, the introduction of variability at reproduction is enhanced compared to that in the asexual species. However, organisms that reproduce by division might have other mechanisms of generating mixtures of their gene pool. For example, some prokaryotes are known to exchange genetic material with their peers [78], as opposed to the gene transfer of genetic recombination, where the genetic material is transferred only from the ancestors.

Most of the kinetic parameters are determined by the nucleotide sequence, encoded by the DNA, and consequently the changes in DNA can introduce changes in the kinetic parameters. Moreover, it is the dynamical features that the individuals exhibit that give rise to their phenotype in a given environment, and therefore, the values of these kinetic parameters that define the dynamical features of gene expression are under evolutionary pressure. One example are the transcriptional and translational elongation rates that depend on sequence of bases and codons that is encoded by the DNA sequence. Even more importantly, the mutations in the regulatory regions can affect other kinetic parameters with small changes causing drastic effects in the expression levels, such as affecting the binding affinity of the regulatory molecules, or causing conformational changes in the macromolecules involved. On the other hand, some of the regulatory molecules are transported to the cell from the environment external to the cell, the environmental conditions playing a major role in determining the dynamics of the expression of the genes.

The process of simulating evolution to solve a computational problem is called genetic programming. This is especially well suited in problems that are NP-hard, and the dimensionality and/or state space are huge. The evolution is mimicked by applying the genetic operators the on a population of individuals. The fitness

of each individual is evaluated, and selection is applied based on the fitness of the individuals. Finally, some means of genetic recombination is used to create the next generation of potential candidates.

To study the evolution of the gene expression, similar techniques can be used. The models provided in the previous sections can be used to construct models of gene expression in cells and in cellular networks, and their parameters can be mutated. The fitness can be measured either by based on abundances of important molecules or arbitrarily generated fitness units, or based on games between the individuals. In models that work explicitly on the nucleotide sequence, the modeling of the mutations can be made to directly affect the genotype. From these changes in the genotype, the effects can be propagated to the dynamics. For example, if different rates of elongation in transcription and translation are known for each type of nucleotide, they can be introduced in the model, instead of concentrating to evolve abstract features such as the overall rate.

However, it is typical that the effects of the mutations are not that well known in full detail. Even single nucleotide mutations might introduce changes in less well known features such as the three-dimensional structure of the molecules, which can in turn affect the dynamical features. Therefore, it is hard to predict what are the exact effects of a mutation in a single nucleotide of the DNA. For this reason, the modeling of these processes often concentrates in varying the parameters, which are known to be evolvable, in a random fashion, rather than explicitly modeling the mutations, and natural selection is let to work out its way to the optimal set of parameters in terms of the constraints imposed.

Similarly to how evolution in the real world is thought to work, genetic programming does not necessary lead to solutions that are optimal. At any given time, only solutions that are sufficiently close to the ones that the previous generation of the population possessed are considered. Nevertheless, if enough time is given, novel combinations of traits will emerge, providing evolutionary advantage to the individuals. Moreover, if a sufficiently large sample is used, the set of results obtained will provide valuable insight about the general behavior of the processes.

2.2.4 Evolution of genetic networks

It is not only the behavior of the individual elements that determine the dynamics of a complex system, such as a genetic network, where the individual elements are intricately connected with interactions between them. As we saw in section 2.2.2, two additional factors can be identified, and must be considered when modeling a complex biological network.

The first factor that controls the dynamics of a network is the nature of the interactions. In section 2.2.2 this was represented by a combination of a boolean

function and a set of chemical reactions used for coupling the proteins with the promoter of the target gene, which were used to determine if a set of input states have promoting or repressing function to expression of the target gene. If the type of interactions were to be altered, the dynamics would likely to be different. For example, if the repressing function of the proteins in the case of the repressilator were all to be replaced with an activating function, the circuit would not feature multistable periodic behavior. Another factor that deserves consideration is the network topology, that in the previous section was defined in terms of an adjacency matrix. The adjacency matrix completely determines the network topology, that is, which elements can interact and with whom. Even similar nodes, put in a network with slightly different topology are likely to have different pattern of behavior as a whole.

First studies of biological networks often utilized regular lattices or random graphs. This was pioneered by Kauffman [10] in random graphs [79], where the nodes were boolean variables representing genes being expressing or not. The sole purpose of this approach was convenience. First of all, it was not usually well known what kind of structures and interactions these network exhibited in the real world. Second, this kind of structures were more tractable for mathematical analysis, and consequently studies about their properties started to emerge.

Later studies pointed out that the networks found in real world do not appear to exhibit topologies of regular or random graphs. For this reason, a family of networks, called small-world networks was proposed. Whereas the regular graphs are characterized by high clustering with the cost of high shortest path length, and the random graphs are characterized by low clustering and relatively low shortest path length, the small-world networks appear to possess best features of both regular and random graphs. The algorithm proposed by Watts and Stroganz [80] allows a parameter to be used to tune steplessly between regular and random graphs.

Small-world networks are characterized by low average shortest path length, that is the minimum distance between two nodes of a network, and high clustering coefficient, which measures the degree to which the nodes tend to cluster together in the network. Due to high clustering coefficient, small-world networks tend to contain cliques, which are subgraphs where the nodes are highly connected. Moreover, these cliques appear to be connected using few hub nodes, which are highly connected, leading to a low average shortest path length. The hubs are both the strength and the weakness of this kind of networks. If nodes were to be deleted in a random fashion, it would be very unlikely that the hubs were deleted, but the deleted nodes would be the ones with low number of connections. On the other hand, the network is vulnerable to deletions which are targeted to the hub nodes, which may cause parts of the network to become disconnected. Nevertheless, even if one of the hub

nodes happened to be deleted by chance, the structure of the remaining smaller parts of the network would stay unchanged, since the leaf nodes are highly clustered, and potentially continue their functioning as a separate subnetworks.

One possibility is that small-world networks are preferred due to their robustness to perturbations over other network architectures [81]. If this was the case, networks with small-world structure could provide advantage to biological systems that are subject to damage by mutations or unexpected disturbances such as a viral infection, and it could be the reason why such networks would be preferred in real life.

It is possible, and likely, to be the case that also the structure of complex networks present in biological systems evolve over time. Networks that are observed in real life are thought to be generated this way. That is, instead of being formed by a procedure of precise design, they are formed by based on continuous expansion and evolutionary pressure on their nodes.

Based on the continuous expansion and preferential attachment, a method for generating networks was proposed [81, 82]. The process of continuous expansion is present on evolving networks that are growing or otherwise transient. This feature preserves already existing structures of the network, allowing it to evolve in a manner where the changes are introduced incrementally in small quantities. The other key feature of the design process is that the newly introduced nodes are attached to the network in a preferential manner. Specifically, when a new node is inserted, it is connected to the existing nodes with a probability proportional to the number of connections in they already possess, making highly connected nodes to become even more connected.

The scheme of generating networks based on continuous expansion and preferential attachment will yield a network where the vertex connectivities of the nodes will follow a power-law distribution. Networks with such power-law distributions are commonly known as scale-free network. Scale-free networks exhibit properties similar to the small-world networks, but with additional characteristics, such as ultra-small path lengths [83]. More recently, there has been evidence in favor that the networks found in real world would be scale-free [81, 84, 85]. This kind of networks were first recognized by de Solla Price in 1965 in networks of scientific publications [84]. The currently known examples range from genetic networks and social networks of human interactions [84] to technological networks such as the world wide web [81].

3. SIMULATING EVOLVING CELL POPULATIONS

3.1 Introduction

One of the objectives of this thesis was to develop a tool, which allows the simulation of evolving stochastic networks. In addition to the dynamics of the simulations being based on stochastic molecular kinetics, the following criteria was selected.

One of the facts is that the networks of cells or genes, which are being modeled, will at times need synchronization and other forms of interaction, despite most of the time working as a cohesive whole, independent of the neighboring individuals. For example, the regulatory networks in a single cell functions somewhat independently of those of the other cells. However, at some points it is necessary to transmit information between these systems. Such communication between the networks is known to occur in biological systems. It is also required that the interactions between the individual elements that are being modeled are not limited interactions that are stationary processes over time. This feature allows studies of evolution of network structures, in addition to evolving the kinetic parameters.

Second major feature was the modeling of environmental conditions. It is the environment along with the genetic material of an individual, which gives rise to a phenotype, and consequently determines the fitness of an individual. Rarely are the environmental conditions static, or even stationary in nature, but rather vary over time. The ability to model different kinds of environments, for example, periodic or oscillatory environments, as a stochastic process was also one criterion in the design. The environmental conditions can be implemented using the dynamics permitted by the DSSA. Modeling of cellular networks in such environments has applications in studying events such as day-night cycles and viral infections.

Moreover, the environmental conditions for a population of individuals are often not homogeneous, but the conditions vary between the individuals, for example, as a function of the spatial location. A single individual which has evolved to be fit to certain local environmental conditions, might be less fit in other regimes. This can promote the generation of complex distribution of phenotypes with different optima for different environmental conditions. To study the effects of non-uniform distribution toxins and nutrients, it is of interest to have this kind of variations in

the environmental conditions to which a single or a small group of individuals are subject to.

Another interesting class of features, whose studies the simulator should enable, are the effects of the spatial structures in cellular populations. The modeling of spatial structure of the environment and the locations the individuals living in this environmental space allows us to study the evolutionary effects as a function of the spatial location. Moreover, local phenotypic distributions can be compared to the global distributions, and the evolution of these distributions can be studied in cases where the speed and function of spreading of mutations, diseases, and genetic material can be constrained.

Furthermore, it is of interest to study the evolving populations without any artificial constraint on the size of the population. For example, an assumption that the population size would be fixed should not be imposed, but the size of the population should be limited by other means, such as making overpopulation of the environment to lead to starvation. Moreover, the individuals should be created and destroyed over time in an asynchronous fashion. These features allow studies of the effects of population size to the survival of the population, such as determining the optimal population sizes in certain kind of evolutionary constraints. For example, we might be interested to study the effects of some limited resource to cell populations that would normally grow exponentially in size. Moreover, this feature allows us to study the evolution of populations that are growing.

3.2 Overview of the simulation

Based on the criteria that was set up for the simulator, the following design was opted. The simulation is based on a set of entities that are being modeled, which is let to vary dynamically both in its size and its content. Each of these entities is an independent DSSA simulation, whose time evolution is simulated according to the algorithm described in section 2.1.5. The final decision of what does an entity represent is left to the user. However, the entities are the representation of the objects on which the genetic operators, such as selection, are applied on.

When simulating the evolution of a genetic network, where the major component under selection might be an individual gene, each entity can be selected to be a single gene. On the other hand, in cases where there are a lot of gene-to-gene interactions, and the performance of different kind of genetic networks are compared, an entity can be taken to represent a cell. The entities in a simulation are not constrained to be equivalent in their behavior. The set of reactions, the set of molecular species, and the kinetic parameters can be set independently for each entity.

The environmental conditions can be modeled either independently for each of the entities or externally as a separate entities. In either case, the environmental

conditions can be coupled with arbitrary entities, independently of them being elements of selection or different environmental conditions. This is done using the entity-to-entity interactions.

The simulation generally consist of simulating the entities and manipulating them over time. The entities are created dynamically. This means that at any point in time, an entity can be created based on a stochastic chemical kinetic model. The entity is formed with a number of reaction channels that transform molecules of different molecular species to those of another species, based on the initial molecular abundances. The creation of an entity is an operator that allows modeling the birth of the individuals.

Another fundamental operation that acts on entities as a whole is their destruction. Using this operator in tandem with the creation allows us a rudimentary modeling of life, a birth-death process. Note that, since at the creation of an entity the molecular concentrations can be arbitrarily specified or altered, using these two operators it is possible implement to more complex schemes of birth that are present in different organisms. An example of this is cell division, which is a process where a mother cell duplicates its genetic information and other macromolecules that vital the survival of the organism, and splits to two. Consequently, the molecules of the mother cell are split to the newly created daughter cells, causing the mother cell to cease to exist. It is also possible to model more complex schemes of reproduction, such as that of the sexual reproduction. In such a scheme, two individuals can be selected to mate, resulting in a birth of a child that inherits the traits from the parents, possibly in a probabilistic manner.

The operator that generates phenotypic variability is the mutation operator. In the simulator, point mutations can be performed at arbitrary points in time in arbitrary entities. The mutations can be applied to any of the kinetic parameters that are involved in their stochastic models. Moreover, the molecular concentrations may be mutated, in case some of the genetic parameters are represented in those terms. Again, it is up to the user to define not only when, but how the mutations are applied and what is the actual effect of the mutation, that is, how much and to which direction these parameter are changed. When applying the mutations, facts such as the fitness of the cell or the environmental conditions can be considered, allowing the mutations to be performed in a non-homogeneous manner.

In a way similar to the other operators, the point mutation operator can be used to form more complex genetic operators. These include genetic crossover, which is the process where some of the parameters are swapped between units such as two genes, possibly in a stochastic manner. The mutation operator can be also applied to other schemes of recombination, such as that of the exchange of genetic material between the peers.

3.3 Implementation of DSSA

The DSSA simulation engine that governs the dynamics of each individual entity is based on that of SGN Sim [62]. The original idea was that SGN Sim would have been invoked as an external program. This would have allowed the users to replace the simulator by a simulator of their choice. Consequently, by replacing SGN Sim by, for example, an ODE solver could have been used to turn the stochastic dynamics into deterministic ones, if such a choice was applicable and advantageous.

However, by using an external simulator for the entities possessed two fundamental problems related to the simulation performance. The first problem was the overhead of processes spawning in the current operating systems. An attempt to circumvent this problem was made by pre-spawning a pool of SGN Sim processes, but it turned out that this was not the only factor hindering the performance. The other problem was that most of the time was spent parsing the reaction files, since the mechanism of continuing simulation was based on dumping the system state as a reaction file that was then reread.

Since the idea of using an external simulator engine turned out to be infeasible, the DSSA simulation engine of SGN Sim was embedded to the main application of the evolution simulator. The authors of SGN Sim, Andre Ribeiro and Jason Lloyd-Price, generously provided the parts of code that were necessary for this process, and few modifications were made to this code that were vital for the evolution of the entities. These modifications include the duplication of reaction systems that are being simulated, and the mutation of the parameter values. Moreover, the input syntax was augmented to mark the mutated parameters in order to reduce the lookup times for systems that feature massive numbers of parameters.

The delayed stochastic simulation algorithm is rather straightforward to implement on a digital computer. When the algorithm outlined in section 2.1.5 is to be implemented, there is only few choices that can be made to improve the performance over a naive implementation. The algorithm that is implemented here is a variation of what is called the logarithmic direct method [86], which under typical conditions, provides asymptotically better runtime compared to the naive approach.

The system state $\boldsymbol{x}$ is implemented in the most obvious way, that is, by using a plain array of the molecular counts. If there are total of n molecular species, this array occupies $\mathcal{O}(n)$ space, and can be created in $\mathcal{O}(n)$ time, even if the size of the array is not known in prior. Also, it is required to store the information about the m reaction channels, each of which might involve information about $\mathcal{O}(n)$ of the substances. This makes the initialization of the simulation algorithm $\mathcal{O}(mn)$ in the worst case, in both space and time. However, it is often the case that not all reactions involve all of the molecular species in the system. If each of the species is

involved in $\mathcal{O}(1)$ reactions, both of the complexities relax to $\mathcal{O}(m+n)$.

The key of the logarithmic direct method is to represent the values of $a_j(\boldsymbol{x})$ as a tree of partial sums [86]. This means that the search for μ, in step 2 of the DSSA algorithm specified in section 2.1.5 is modified from the naive approach. Instead of forming the sums of equation 2.26 for each iteration, an array of partial sums are maintained. This array is $\mathcal{O}(m)$ in space, and permits that the reaction to be fired can be found using binary search in $\mathcal{O}(\log m)$ time. It is to note that after updating $\boldsymbol{x}$, we need to update each affected $a_j(\boldsymbol{x})$, and the partial sums in which they are involved. The consequent time complexity is $\mathcal{O}(mn + m\log m)$, but again $\mathcal{O}(mn)$-part vanishes if all substances are involved in $\mathcal{O}(1)$ reactions.

The uniform random numbers that are required in step 3 of the algorithm, are generated using Mersenne Twister MT19937 random number generator [87]. It is a 32-bit uniform pseudorandom number generator with a period of $2^{19937}-1$ that generates random numbers that are k-distributed in 624 dimensions to the 32-bit accuracy. The sequences of numbers generated pass numerous tests of statistical randomness. Unlike the naive pseudorandom number generators, it is generally considered to be adequate for numerical simulations. The generation of the numbers requires constant time and space.

The waiting list of the DSSA is implemented as a binary heap. A binary heap occupies linear space, that is, $\mathcal{O}(w)$ space if w is the number of elements in the heap. Moreover, it permits lookup to the largest element in $\mathcal{O}(1)$ time, and the extraction of the largest element and insertion in $\mathcal{O}(\log w)$ time. The size of the waiting list is limited by $\mathcal{O}(tn)$, where t is the number of reactions executed. As a consequence, the operations on the waiting list per reactions executed will be $\mathcal{O}(n)$ amortized time and space.

Using the details provided in above, the following applies for our implementation of the DSSA. Step 1 is limited by the initialization of the molecular counts. The time t is a scalar so it is $\mathcal{O}(1)$, the molecular counts take $\mathcal{O}(n)$, and an empty waiting list is $\mathcal{O}(1)$, each of them in both time and space. The second step takes $\mathcal{O}(mn)$ space and time to initialize the $\mathcal{O}(m)$ partial sums of $a_j(\boldsymbol{x})$s, whereas subsequent executions of this step takes $\mathcal{O}(mn + m\log m)$ time. In step 3, generating τ is $\mathcal{O}(1)$ and finding μ is $\mathcal{O}(\log m)$ time. In the next step, the comparison is $\mathcal{O}(1)$ time. The release of an item from the waiting list is $\mathcal{O}(w)$ time followed by the mutation of a single element, which can be done in constant time. Step 5 is $\mathcal{O}(n)$ time, whereas step 6 is $\mathcal{O}(n\log w)$ time, since $\mathcal{O}(n)$ molecular species can be involved.

To summarize the complexity of the DSSA implementation, the worst case space complexity is $\mathcal{O}(mn)$ with a time complexity of $\mathcal{O}(mn)$ for initialization and $\mathcal{O}(mn)$ per reactions executed. The worst case occurs when there is a substrate that is involved in $\Theta(m)$ reactions. In the case where each of the m molecular species are

involved in only $\mathcal{O}(1)$ reactions, the respective numbers are relaxed to $\mathcal{O}(m+n)$, $\mathcal{O}(m+n)$, and $\mathcal{O}(m \log m)$.

3.4 Implementation of the simulator

The simulator was built to utilize event-driven architecture coupled with a concept of time. The events are scheduled in a queue with arbitrary waiting times. This is similar to the waiting list of the DSSA. However, the event queues were implemented using stable queues, that is, for events with equal time, the event that was scheduled first gets executed first, such that the order of the events is predictable.

To promote parallelization, events are separated to those that do not require interactions between the entities, and to those that do. The former kind of events can be executed in worker threads, in parallel, while for the latter, synchronization is required. The events can be either a combination of the built-in events, or arbitrary functions written in Lua [37]. The simulation is performed by manipulating the event queues as follows:

1. the events are pushed to their appropriate event queues
2. the first event E_g is popped from the global event queue
3. each of the entities are simulated up to time moment t_g; this step can be done in parallel
   (a) if for the time t_ℓ of the first event E_ℓ in the local event queue $t_\ell < t_g$ holds
      i. perform the DSSA simulation up to time moment t_ℓ
      ii. perform event E_ℓ
      iii. go to step 3a
   (b) perform the DSSA simulation up to time moment t_g
4. the event E_g is performed
5. go to step 2

In the above algorithm, step 3 can be performed in parallel for each entity, in a separate unit of processing. The current implementation uses multiple threads, which are individual entities of execution in the operating system scheduler, sharing the same memory address space. The memory address space can be used for synchronization and inter-thread communication. The number of worker threads should be chosen to be the number of logical processing units in the system, if all resources of the computer is to be optimally utilized.

The built-in events include creation, duplication, and destruction of entities. The latter two can be done on individual entities without triggering the synchronization. The space and time complexity of each of these events $\mathcal{O}(1)$ with respect to the number of entities in the simulation, and that of a single DSSA system with respect to the number of molecular species and reactions involved, as discussed in the previous section. Similarly, querying a value of a kinetic parameter or a number of molecules of a certain species of an entity is $\mathcal{O}(1)$ time, and making a mutation is $\mathcal{O}(n + \log m)$ time, where n is the number of the molecular species and m is the number of the reactions in the entity. The details and derivation of these bounds are discussed in section 3.3.

Moreover, the events that manipulate the sets of entities have their expected complexities. These events require synchronization, and are performed in the main thread. For k representing the number of entities involved in the simulation at the same time, the time complexity for selecting and shuffling is $\mathcal{O}(k)$, $\mathcal{O}(k \log k)$ for sorting, and head and tail operations are $\mathcal{O}(k)$ time, which can be lower ($\mathcal{O}(1)$ for example) depending on the parameters. Each of these operations is asymptotically optimal, and is performed in-place, such they can be easily chained and require no additional space for intermediate storage.

The operations requiring random numbers, such as the shuffling, and the random number generation functions exposed to the user, utilize a Mersenne Twister MT19937 pseudorandom number generator [87]. There is one generator for the purposes of whole simulation operated by the main thread, from which a generator for each entity is seeded. The user can seed the global generator in case the exactly same path of simulation is to be reproduced.

3.5 Describing the DSSA system

As a consequence of the simulators inheritance to SGN Sim, the specification of the DSSA system to be simulated is given the in format of SGN Sim. Full details can be found in the SGN Sim manual [88], but a short explanation is given in this section to give an idea what kind of actions can be easily implemented.

The parameters are specified using identifier-data pairs. These directives are used to provide all the required information to the simulator. The directives should be written into a text file, which is then loaded at the creation of an entity. It is also possible to dynamically create the configuration for the DSSA simulator, in the fashion of the command line parameters of SGN Sim. Note that since the entities can be duplicated, it is not necessary to reload the file at the creation of every duplicate. Moreover, after the creation of the DSSA system, it is possible to mutate any of the parameters prior to starting the simulations.

In the configuration file, the identifier-data pairs can be provided in any of the

following formats. These pairs can be mixed with C++ style comments [36].

1. *identifier data* ;
2. *identifier* { *data-1* ; ... *data-N* ; }
3. *identifier* !{ *data* }!

The first two forms are useful for specifying data that is well behaved, that is, does not contain the control characters that include the braces and the semicolon. The second form is syntactic sugar for:

identifier data-1 ;
⋮
identifier data-N ;

and the third form is used to specify data that is not well behaved, such as sections of arbitrary Lua code [37].

The comments have the rules of C++, that is, two types of comments are recognized. The first type is the multiline comment `/* ... */`, and the second is the single line comment `// ...` , where the part `...` is the comment, and can contain arbitrary sequence of characters not including the comment terminator. In the former case, the comment can span multiple lines, whereas in the latter, the comment is terminated on the newline.

The following identifiers are recognized, and they have their equivalent behavior of SGN Sim: `include`, `lua`, `molecule_readout`, `population`, `queue`, `reaction`, `readout_interval`. Additionally, the following set of identifiers are recognized and parsed appropriately, but ignored for various reasons: `fourier_file`, `output_file`, `output_file_header`, `performance`, `progress`, `save_file`, `save_index`, `save_interval`, `save_now`, `seed`, `stop_time`, `time`, `warn`. Most of these ignored parameters are provided by other means, and some of them were excluded for their complexity.

The directive `include` is used for inclusion of subsequent configuration files. The argument *`data`* specifies the file name that is to be read, and the directive is effectively replaced by the contents of that file. The syntax of the filename is platform-specific, and not discussed here.

The `molecule_readout` directive is used to control the printing of the molecule counts to the output file. If the argument *`data`* is `show`, any subsequently added molecular species will be printed in the output file. The argument value `hide` can be used to make the counts of the subsequently added molecular species not to be printed in the output file.

The `population` directive specifies the initial number of molecules. The following forms of the directive are recognized:

```
population species = count;
population species += count; // Add
population species -= count; // Subtract
population species; // Equivalent to species = 0
```

If a species appears for which the molecular count is not specified, it is taken to be zero, and any possible action, such as addition or subtraction, is performed after that operation. Similarly, if the species was not previously introduced but appears in a reaction, its concentration is taken to be zero.

Additionally, molecules can be placed on the waiting list prior to the beginning of the simulation, for the purpose of introducing them to the system at a later point in time. This is achieved with the `queue` directive, and the following specifies its syntax:

```
queue [count]species(release-time);
```

which causes *count* molecules of the molecular species *species* to be released at time point *release-time*. If the number of molecules to be released is one, *count* can be omitted along with the square brackets. Moreover, the square brackets are not required if *count* is a number.

The reaction channels are specified using the `reaction` directive. The following format is used:

```
substrate-list --[ rate-constant ]--> product-list
```

where *substrate-list* is a sequence of the molecular species that act as the substrates of the reaction, separated by `+`, specified in the following format:

```
[count]species(rate-function:param-1,...,param-N)
```

and *product-list* is a list of the same type for the reaction products in the following form:

```
[count]species(delay-distribution:param-1,...,param-N)
```

Again, when *count* is one it can be omitted, and the same rules as above apply for the square brackets. Moreover, if the rate function 2.13 is to be used or no time delay is to be present, the parenthesized expressions specifying them can be omitted. Additionally species can be prefixed with a `*` to prevent the consumption of a substrate. For the different values of *rate-function*, *delay-distribution*, and their parameters, please refer to the SGN Sim manual [88].

An extension to the syntax presented above is that the parameters that are to be mutated are written in the form of `mutable(`*param*`)` instead of the *param*. This applies to parameters such as the molecular counts, reaction rates, and the parameters of rate functions and delay distributions.

Finally, the sampling interval for output printing is specified using the directive `readout_interval`. These causes a line to be printed in the output file with intervals specified by the arguments *data*, containing the molecular counts of each species for which the printing was not suppressed. Note that the sampling interval is not related to the simulated precision, unlike in some ODE solvers, and does not affect the dynamics of the simulation.

3.6 Manipulating evolving cell populations

The simulator utilizes the parser of the built-in Lua interpreter [37]. The files that describe the simulated system are Lua scripts, and arbitrary Lua code can be included in them. However, in addition to the built-in functions of Lua, a set of functions that is used to control the simulation is exposed.

The simulator is invoked in the command line. Currently, there is only a command line interface available. The syntax on the command line is the following:

```
cellsel [--function [argument] | script] [...]
```

where, *function* is the name of a function to be evaluated, its argument *argument* is optional, and if provided, it is evaluated as a Lua expression and the values are passed to the corresponding function. If any *script*s are provided, they are interpreted as filenames of Lua scripts that are to be run.

The simulation consists of manipulating entities and simulating their time evolution. The entities are automatically simulated, that is, the user is left to specify the actions how the entities are to be manipulated. The simulation is outlined using the following functions:

```
each(interval, action-1, ..., action-N)
once(interval, action-1, ..., action-N)
run(stop_time)
time = time()
```

The functions `each()` and `once()` are used to schedule events. In both cases the expression *action-1*`(...(`*action-N*`()))` is evaluated when the simulation time reaches `time() + interval()`. Moreover, in the case of the former, the expression is evaluated indefinite times with intervals obtained by successive invocations of `interval()`.

The function `run()` makes the simulator to simulate the currently specified system up to time moment *stop_time*, whereas the function `time()` may be used to query the current state of the simulation, for example, inside the events.

Entities, which represent individual simulations of DSSA, are manipulated using the entity objects. The entity objects behave like a Lua tables. You can use the

regular means to query and/or manipulate an entity. Note that the indexing is zero-based (Lua typically uses one-based indexing):

```
#entity
entity[0]
entity['A']
entity.X = 5
```

where the first form returns the number of parameters in the entity, that is, the number of different molecular species plus the number of mutable parameters. The last three forms exemplify the different forms querying and manipulation of the parameters of an entity. As shown in the examples, either raw indices or parameter names can be used.

Each of the entities has an unique identifier, that is automatically generated from a pattern. The pattern can be specified by user using the following function. It is not necessary that the idenfiers of the entities follow the same pattern.

```
output_pattern(pattern)
```

where *pattern* is a Lua string, in which the substring '%%' appears one or more times. When an identifier is generated, this substring is to be replaced by an undefined number such that the pattern becomes unique.

A range object is used to represent a sequence of entities and it behaves like a sequential Lua table with the exception that it is immutable. To obtain the entity objects, they must be extracted them from a range object. Regular means of Lua can be used to query a range:

```
#range
first = range[0]
last = range[#range - 1]
```

where the first expression returns the number of entities in a range, whereas the second and the third expressions return the first and the last entity of the range, respectively.

Moreover, the entity and range objects can be iterated in the fashion that is typical to Lua:

```
for _, entity in range do
  for index, key, value in entity do
    ...
  end
end
```

In addition, the following set of functions is made available for manipulation of the range objects.

```
range = all()
range = create(arg-1, ..., arg-N)
range = dup(range)
range = head(count, range)
range = kill(range)
range = select(predicate, range)
range = shuffle(range)
range = sort(predicate, range)
range = tail(count, range)
```

The function `all()` returns a range that spans over all entities. A new entity can be created using the function `create()`, where the arguments *arg-1*, ..., *arg-N* are passed to SGN Sim, whereas the function `dup()` duplicates each of the entities contained in the range provided as an argument. Both of the functions return a range spanning over the newly created entities.

Entities are destroyed using the function `kill()`, which returns the entities that were left to the system. The functions `head()` and `tail()` can be used to limit the number of entries in a range. The former includes *count* first entities, and the latter *count* last entities. If provided *count* is smaller than the range length *#range*, the range will be unmodified.

The functions `select()`, `sort()`, and `shuffle()` manipulate their argument *range* accordingly. In the context of `select()`, the argument *predicate()* is a function that maps an entity object to a boolean, `true` denoting that the entity should be included in the range. Alternatively, the parameter *predicate* can be a string denoting that a presence of substance of that name is required, or a string prefixed with `'!'`, inverting the condition. For `sort()`, the argument *predicate*`()` is a function that returns `true` if and only if the entity provided in the first parameter is to be sorted prior to the entity in the second parameter. Again, *predicate* can alternatively be a string, optionally prefixed with `'+'` or `'-'` denoting ascending or descending sorting order. The default is ascending. Finally, shuffling uses a modified Fisher-Yates shuffle [89], to shuffle the elements that are included in the range provided as an argument. This means that a permutation of the elements is chosen with each of the permutations having an equal probability.

Also, a corresponding generator is available for each of the range manipulation functions. The generators can be used as events in the argument list of scheduling functions `each()` and `once()`.

```
callable = All(...)
callable = Create(...)
callable = Dup(...)
callable = Head(...)
callable = Select(...)
callable = Shuffle(...)
callable = Sort(...)
callable = Tail(...)
```

The facilities for pseudorandom number generation is provided using both functions and generators. The generator counterparts are useful as the first argument for the scheduling functions `each()` and `once()`, whereas the functions can be called in user-defined events.

By default, the random number generator is initialized based on the system time and the process id. This seed should guarantee that the results are different between separate runs. However, it is sometimes desirable to be able to reproduce the trace of a single simulation exactly. In this case, the random number generator can be initialized with a fixed seed:

```
seed(seed_id)
```

The following functions can be used to generate random numbers with various distributions:

```
number = betarnd(alpha, beta)
number = chi2rnd(nu)
number = exprnd(lambda)
number = gamrnd(alpha, beta)
number = geornd(p)
number = normrnd(mu, sigma)
number = rand()
number = unidrnd(a, b)
number = unifrnd(a, b)
```

where the function `rand()` is the raw interface to the underlying uniform pseudorandom number generator, and the other functions generate variates with beta, chi-squared, exponential, gamma, geometric, normal, discrete uniform, and continuous uniform distribution, in that respective order. Beta distribution is parametrized using two shape-parameters *alpha* and *beta*, chi-square by degrees of freedom *nu*, exponential by rate *lambda*, gamma distribution by shape parameter *alpha* and rate parameter *beta*, geometric by the probability *p*, and normal by mean *mu* and standard deviation *sigma*. The function `rand()` returns raw unscaled uniform random number from the underlying generator. In contrast, the function `unidrnd()`

returns uniform random integers and the function `unifrnd()` uniform random real numbers from the continuous uniform distribution, both scaled to the semi-open interval $[a, b)$.

Each of the functions that generate random variates also has a corresponding generator. Some generators have multiple aliases. Again, these generators are useful in combination with the scheduling functions `each()` and `once()`.

```
callable = Beta(...)
callable = Chi2(...)
callable = Exp(...), Exponential(...)
callable = Gam(...), Gamma(...)
callable = Geo(...), Geometric(...)
callable = Gaus(...), Gaussian(...), Norm(...), Normal(...)
callable = Unid(...), UniformDiscrete(...)
callable = Unif(...), Uniform(...)
```

Moreover, the following generator is available for convenience:

```
callable = Const(value-1, ..., value-N)
```

which is a generator that returns the arguments *value-1*, ..., *value-N* that were provided on its creation. This generator is useful for generating intervals from degenerate distributions, where the intervals are deterministic.

Finally, there are some utility functions that are useful for composing the desired set of events from the built-in ones:

```
callable = bind(fun, arg1, ..., argN)
callable = compose(fun-1, ..., fun-N)
loop(count, fun-1, ..., fun-N)
```

Here, the function `bind()` returns a callable object that binds the arguments *arg-1*, ..., *arg-N* to the function *fun*. Consequently, the expressions `bind(`*fun*`, `*arg-1*`, ..., `*arg-N*`)(`*arg-N+1*`, ..., `*arg-M*`)` and *fun*`(`*arg-1*`, ..., `*arg-N*`, `*arg-N+1*`, ..., `*arg-M*`)` are thus equivalent. The function `compose()` returns a callable object that is the function composition of its arguments *fun-1*, ..., *fun-N*, consequently making the expression `compose(`*fun-1*`, ..., `*fun-N*`)(`*arg-1*`, ..., `*arg-M*`)` equivalent to the expression *fun-1*`(...(`*fun-N*`(`*arg-1*`, ..., `*arg-M*`)))`. Furthermore, the function `loop()` that can be used to invoke a function, such as `create()`, several times. Invoking `loop()` invokes *count* times the expression *fun-1*`(...(`*fun-N*`()))`.

Using the set of functions presented above, evolutionary simulations can be easily composed. For example,

```
loop(100, Create('foo.g'));
each(Exp(10), dup, kill, Head(.5 * #all()), Sort('A'));
run(10000);
```

will simulate a population of 100 entities, killing the worst 50 entities determined by their fitness measured in the number of molecules of the species `A`, and duplicating the rest. The time intervals for the selection are drawn from exponential distribution with a mean of 10, and the simulation is performed until time moment 10000.

4. EVOLUTIONARY DYNAMICS OF A POPULATION OF CELLS WITH A TOXIN SUPPRESSOR GENE

4.1 Introduction

A study was performed, using a stochastic model, to investigate the evolution of the dynamics of a population of cells that are subject to a toxin [90]. The toxin is introduced in the cell in a stochastic manner with a rate depending on the environmental conditions. To cope with this threat, the cells regulate the expression of a gene that is used to degrade the toxin.

A self-repressing gene used to degrade a toxin has been characterized in bioluminescent *Escherichia coli* K-12 cells [91]. This gene is responsible for producing TetR proteins that are used to degrade tetracycline, which is a substance toxic to the cell. In the absence of tetracycline, the produced TetR proteins bind to the promoter region of the gene producing them causing repression of the gene, effectively leading to a scheme of inhibitory self-regulation.

In the model used, the environmental conditions are not only stochastic, but also transient. The evolution of cells was investigated in environmental conditions where the introduction of toxin is either on or off. Environmental conditions with both predictable and unpredictable period were considered.

This model was used to quantify the effects of different environmental conditions to the genetypic and consequent phenotypic diversity of the cell population. Furthermore, it was quantified how sudden environmental changes affect these diversities. Finally, the optimal mutation rates for the parameters regulating the system were quantified as a function of the parameters in the environmental conditions.

4.2 Model

In the study, a population of k cells was simulated. Each of the cells contains a self-repressing gene responsible for degrading the toxin. The gene expression is a two-step delayed stochastic model of gene expression, accounting for the various steps in transcription and translation. The gene expression is represented by the

following set of reactions:

$$P + R \xrightarrow{k_m} P(\tau_P) + R(\tau_R) + m(\tau_m) \tag{4.1}$$

$$m + r \xrightarrow{k_p} m(\tau_{m'}) + r(\tau_r) + p(\tau_p) \tag{4.2}$$

$$m \xrightarrow{d_m} \emptyset \tag{4.3}$$

$$p \xrightarrow{d_p} \emptyset \tag{4.4}$$

$$P + p \underset{k_u}{\overset{k_r}{\rightleftharpoons}} P' \tag{4.5}$$

$$P' \xrightarrow{d_p} P \quad , \tag{4.6}$$

where P represents the promoter region in active state, R an RNA polymerase, m a messenger RNA, r a ribosome, and p a protein. This model is similiar to those represented by reaction equations 2.32 and 2.33. Furthermore, P' represents the promoter region that is bound by protein p, causing the gene to be in the repressed state where its expression is inhibited. The stochastic rate constants k_m, k_p, d_m, d_p, k_r, and k_u represent the transcription initiation rate, translation initiation rate, mRNA degradation, protein degradation, repressor protein binding rate, and repressor protein unbinding rate, respectively. Moreover, the time delays τ_P, τ_R, τ_m represent the time it takes after an initiation of transcription for the promoter clearance, RNAP becoming available for subsequent transcription, and forming the ribosome binding site in the mRNA, respectively. Similarly, their translational counterparts $\tau_{m'}$, τ_r, τ_p represent the time taken after initiation of translation for ribosome binding site clearance, ribosome becoming available for next translation, and forming a functional protein, respectively. This part of the model is based on the models of gene expression discussed in sections 2.2.1 and 2.2.2.

The toxin is produced in the environment, and their cellular actions are modeled using the following reactions:

$$\emptyset \underset{d_X}{\overset{k_X}{\rightleftharpoons}} X \quad , \tag{4.7}$$

where X represents the toxin, k_X is the rate at which the toxin is transported to the cell, and d_X is the degradation rate of the toxin. The interactions between the toxin molecules and the proteins produced by the cells are represented by the reactions:

$$X + p \xrightarrow{k_a} X' \tag{4.8}$$

$$X + P' \xrightarrow{k_a} X' + P \quad , \tag{4.9}$$

where k_a is the association rate of the protein-toxin binding. The fitness of a cells

is measured in terms of the fitness units, that are created in a stochastic manner according to the reaction:

$$\emptyset \xrightarrow{k_f} f \quad \text{where} \tag{4.10}$$

$$k_f = \alpha \, (1 + [X']) \, (1 + [p])^{-1} \, (1 + [X])^{-1} \quad , \tag{4.11}$$

where f represents a fitness unit, and k_f is the rate at which the fitness units are being produced. The fitness function is based on the fact that not only the toxin X is assumed to be harmful to the cell, but also the excess of proteins p. The excess of these proteins will lead to cell death due to loss of membrane potential [92], and should be thus avoided.

The size of the cell population k is kept constant. Each of the cells are simulated for a fixed lifetime ℓ. After the lifetime of a generation is past, q-quantile of the cells are selected for reproduction. The reproduction is based on the fitness units f present in the cells at the time of the division. The cells with highest fitness will be selected to produce the largest offspring, while the cells with the lowest fitness produce no offspring. For simplicity, only the value of $q \leftarrow 0.5$ was considered, and the surviving cells were let to produce an offspring of two cells. At the time of reproduction, the daughter cells inherit all features of their mothers by becoming duplicates of the mother cell. Only the fitness counter is set to zero, such that the fitness of the newly created cells is not dominated by that of their ancestors.

Furthermore, the parameters k_m, k_r, and k_u are let to mutate. These parameter are known to be dependent on factors such as the genetic sequence of the promoter, which is a genotypic feature that is known to be mutable. For each of these parameters, the following mutation scheme is used:

$$k = k^* \, \left(1 + \delta \, [n_k^+]\right) \left(1 + \delta \, [n_k^-]\right)^{-1} \quad \text{with} \tag{4.12}$$

$$\emptyset \xrightarrow{k_\delta} n_k^+ \tag{4.13}$$

$$\emptyset \xrightarrow{k_\delta} n_k^- \quad , \tag{4.14}$$

where k is the effective parameter value, k^* is the initial value of the parameter, δ is the mutation step size, and n_k^+ and n_k^- are the mutation counts, to up and down, respectively. The rate constant k_δ is the mutation rate. Note that the mutations are performed in a stochastic manner.

4.3 Results

A population of $k = 100$ cells was simulated, for various number of generations and for different environmental conditions. The cell division time was set to $\ell \leftarrow 1800$,

which is the average division time in *E. coli*.

The following values of rates were used throughout the study: $k_p \leftarrow 0.0005$, $d_m \leftarrow 0.005$, and $d_p \leftarrow 0.0004$. Also, the initial values for the mutable parameters were set to $k_m^* \leftarrow 0.0025$, $k_r^* \leftarrow 0.0001$, and $k_u^* \leftarrow 0.1$. Finally, the time delays were set to $\tau_P \leftarrow 2$, $\tau_R \leftarrow 40$, $\tau_m \leftarrow 2$, $\tau_m' \leftarrow 2$, $\tau_r \leftarrow 20$, and $\tau_p \leftarrow 50$. These parameters were selected based on the justifications presented in a previous study by Zhu et. al. [63], and in the citations therein.

The expected amount of toxin was controlled by tuning k_X, while the toxin degradation rate was kept fixed at $d_x \leftarrow 0.01$ [93]. Additionally, the toxin-protein binding rate was set to $k_a \leftarrow X$, which is within realistic ranges according to [93]. Moreover, fitness scaling of $\alpha \leftarrow 1$ was used throughout the simulations.

The standard mutation rates in *E. coli* are known to be in the order of 10^{-7} per cell division, but are also known to vary in orders of magnitudes, depending on internal and external factors, such as environmental conditions [94]. The mutational parameters k_δ and δ were let to be varying parameters in this study.

First, the effects of varying δ was studied while keeping the mutation rate fixed at $k_\delta \leftarrow 0.0001$. The toxin is introduced to the cells with a rate of $k_X \leftarrow 0.1$ for ten generations, after which, k_X is set to zero for another ten generations, repeating this procedure to produce an environment where the toxin concentration is a stochastic function with a periodic, transient distribution. The mutation step size δ was only found to affect the time it takes for the population to reach the maximal value of fitness permitted by the set of imposed parameters.

The second step was to vary k_δ in the range $[10^{-7}, 1]$. For this purpose, it was let $\delta \leftarrow 10$, such that the mutation effects were able to propagate fast enough in the simulation timescale. It was found that outside of the range $k_\delta \in [10^{-6}, 0.1]$ the mutation rate was either too low or too high. In the former case the mutations could not introduce effects that would be significantly advantageous compared to the inherent stochasticity between the individuals, while in the latter case selection was not able to eliminate the effects of accumulation of harmful mutations.

Next, by letting $k_\delta \leftarrow 10^{-4}$ and $\delta \leftarrow 10$, the distribution of phenotypes was studied in a periodic environment. For the first 100 generations, no toxin was set to be present (k_X set to zero), and for the second 100 generations, the rates were set to $k_X \leftarrow 0.1$ and $d_X \leftarrow 0.001$. Again, by repeating this procedure an environment with a period of 100 generations was created. The phenotypic diversity was quantified using squared coefficient of variation ${c_v}^2$, that is, the variance over the square of the mean, from the mutable parameters.

It was found that after a change in the environmental conditions the phenotypic diversity was significantly increased, after which it settled down to a value near zero. This verifies that changes in the environment are followed by transient increases in

phenotypic diversity of the cell population, during which the cell population adapts to the new environmental conditions. Afterwards, this diversity is reduced, since the optimal solution has been adopted by a majority of the individuals. This effect is visible in figure 4.1, where the transient period appears to last up to 40 generations after each environmental change.

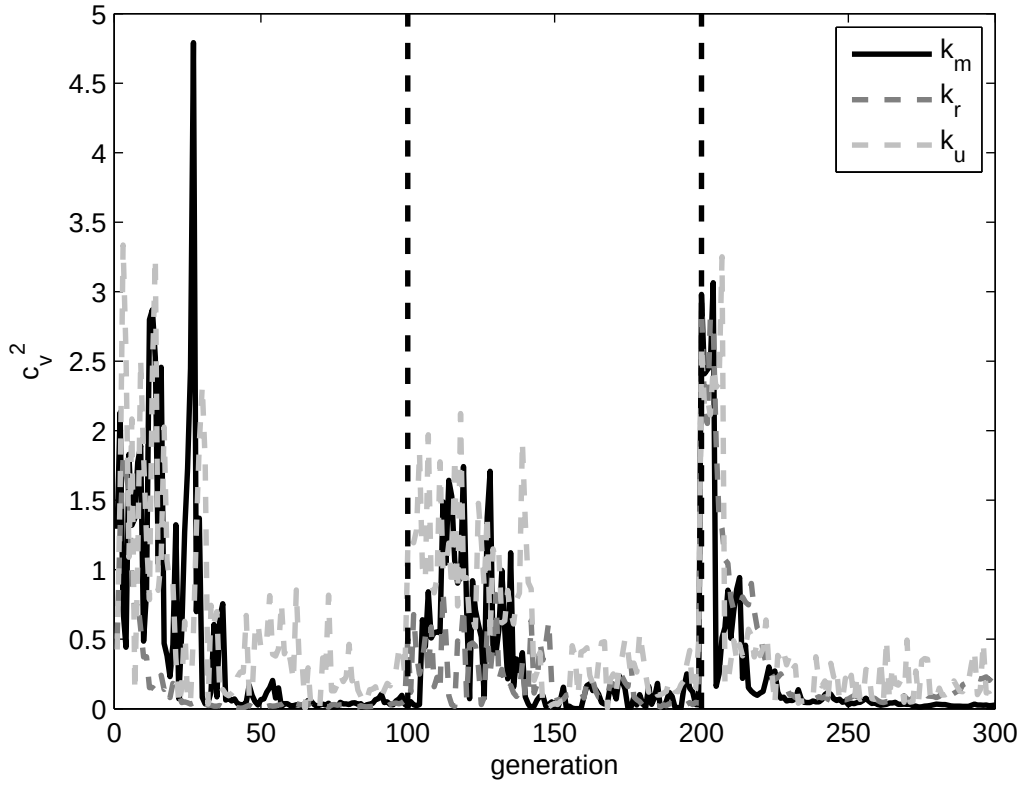

Figure 4.1: Time evolution of the phenotypic diversity of the mutable rate parameters k_m, k_r, and k_u in a periodic environment with period of 100 generations. The dashed black lines represent changes in the environment; toxin is introduced to the cells during the generations 100 through 199.

It appears that even for fixed mutation rates, as was the case in the simulations, the phenotypic diversity is affected by the changes in the environment. The environmental changes trigger sudden increases in the diversity of the population. Moreover, in stationary conditions, a constantly mutating population maintains a certain degree of diversity even after arriving near to the optimal phenotype, due to continuous introduction of mutations that are harmful or neutral in nature.

While the harmful mutations get quickly eliminated, the neutral mutations give rise to complex distributions of phenotype. Figure 4.2 depicts the phenotypic distribution of the parameter k_r, which is one of the rates controlling the repressor binding affinity. With the set of parameters discussed above, some of the cells appeared to opt for tuning the value of k_r while the others opted for tuning k_u. The

resulting distribution of phenotypes is a complex multimodal distribution. Both of these paths of evolution provide the same expected promoter availability and might be equivalent in terms of fitness.

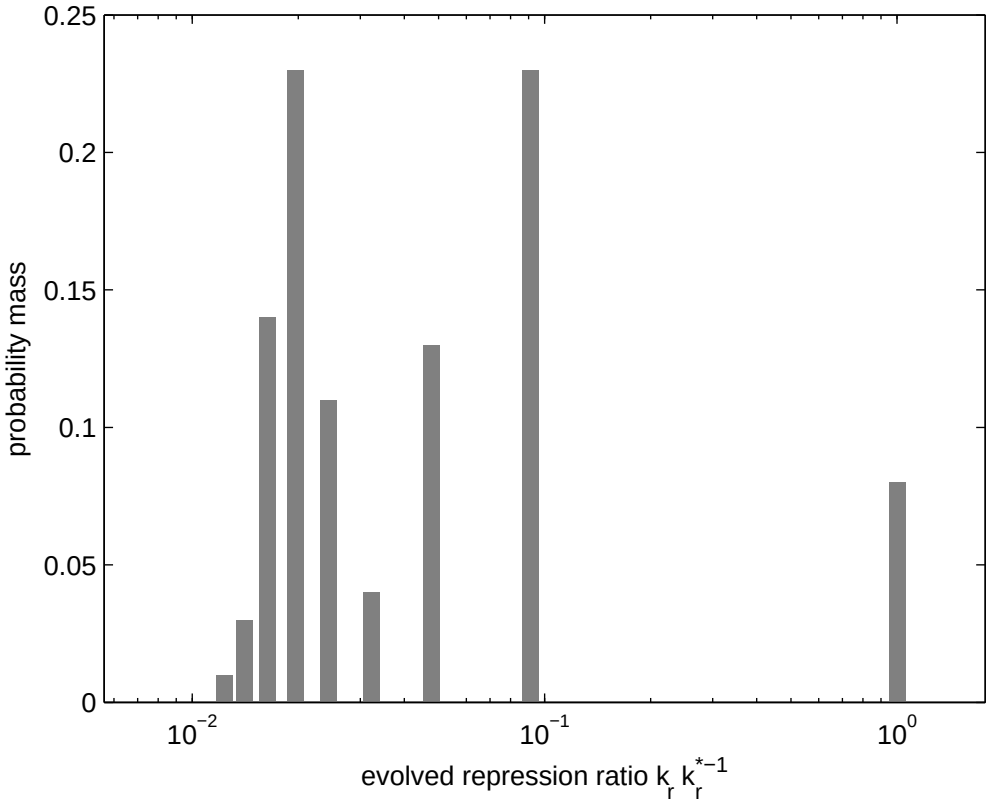

Figure 4.2: Distribution of the evolved value of $k_r\, k_r^{*-1}$, that is, the evolved ratio of the repression rate to its initial value, in a population of 100 cells. The cells are from 100th generation, that is, just before the change in the environmental conditions such that the distribution is well-evolved.

Finally, the optimal mutation rates were quantified in environments with unpredictable period. The environment was let to change state with intervals drawn from an exponential distribution with a mean of $(10\,\ell)^{-1}$. Environments with different rates of toxin production were tested, namely, $k_X \leftarrow 0.001$, $k_X \leftarrow 0.1$, and $k_X \leftarrow 1$. Meanwhile, the degradation was kept constant at $d_X \leftarrow 0.01$. The average fitness of the cell population as a function of the mutation rate in these three environments is shown in figure 4.3.

It was found that the optimal mutation rates depend on the environmental conditions, that is, on the expected amount of toxin. The reason for this is that when an environmental change is triggered, the higher mutation rates provide evolutionary advantage, since the cells are likely to be of suboptimal phenotype and larger measures are required to quickly change the phenotype. However, in an environment where the environmental changes are infrequent, even lower mutation rates allow

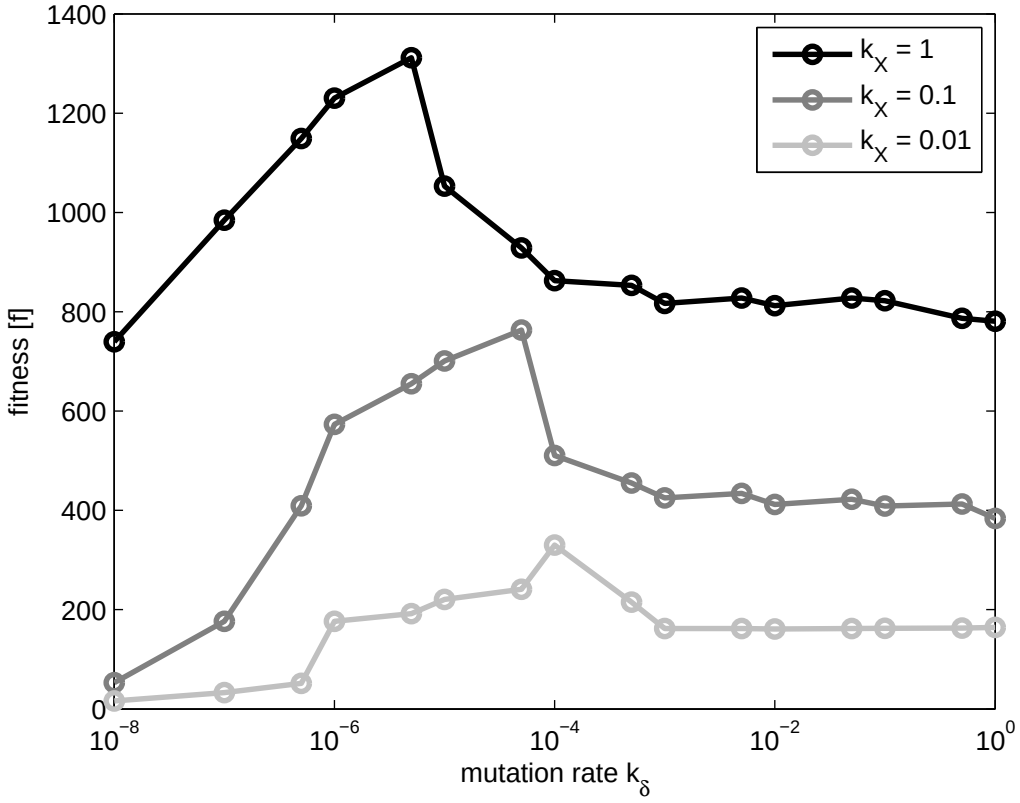

Figure 4.3: Average fitness $[f]$ in a population of 100 cells as a function of the mutation rate k_δ, for cell populations subject various concentrations of toxin, controlled by the rate k_X. The cells are subject to an environment with unpredictable period and the fitness is measured at the end of the simulation, from the 100th generation.

the cells to adapt the environmental conditions while possessing evolutionary advantage over the higher rates by allowing more fine-grained tuning of the phenotype during the stationary-like periods. This suggests that it is advantageous for the cells to tune their mutation rates, to obtain better adaptability for a wider range of environments. Such tuning of mutation rates has been observed in real cells, for example, in a population of bacteria living in mouse gut the individuals are able to control their mechanism of DNA repair, and consequently can vary the effective rate of mutations [32].

5. EVOLVING THE KINETICS OF SINGLE GENE EXPRESSION

5.1 Introduction

Another example, in which the simulator was utilized, was a study of evolving kinetics of a single gene expression in an stochastic environment where the food is scarce and the individuals of the cell population compete for survival. This allowed the quantification of the optimal set of parameters for both the mean expression level and the fluctuations involved in protein levels, as a function of the environmental conditions. Also, it was studied if there are optima with different characteristics, triggered by different regimes of the environmental parameters.

It is known that in biological systems, different genes have evolved to express with various rates, depending on their function. The optimal rate of gene expression is controlled by several factors. In addition to the energetic costs, such as wasted energy and raw materials, the gene expression is often limited by other constraints such as some genetic products being harmful in excess concentrations [92].

Furthermore, different genes have evolved to have certain degree of fluctuations in the levels of their products. Under some conditions it is favorable to have more precise intervals between the production of the resulting polymers, leading to smaller fluctuations in levels of abundance. On the other hand, under different conditions bursty expression and/or on-off periods of activity of the gene might be preferred. Numerous examples of the differences between the expression of distinct genes have be found [95].

As discussed in section 2.2.1, the distribution of protein levels is a result of a series of complex processes that are related, for example, to the regulation of gene expression, the assembly of the polymers, and finally degradation. In this study, instead of using a model with high level of detail in these processes, a more abstract model was opted for. The dynamics are let to be controlled by a sufficiently flexible family of physically feasible distributions, whose parameters can be evolved. For example, the duration of the intermediate steps in transcription initiation [96] and the number of pause-prone sites [97, 98] in the process of elongation are determined by the genetic sequence of the promoter and gene regions, respectively. These details of these processes are known to regulate both mean and variability of time intervals

between production of consecutive proteins [96–98], resulting in differences in the distribution of temporal protein numbers [99, 100].

5.2 Model

In the study, a population of k cells was modeled, in an environment, which provides food for consumption in the cells. The population size k is let to freely vary over time. By consuming food that is available to a cell, the cell then produces a protein, and by the accumulation of these proteins, the cell can become competent, triggering reproduction by division.

In the model, a stochastic environment is featured. In the environment, the food is produced with a constant rate. To prevent the accumulation of the food and to promote its consumption, the food is let to degrade. The food can be taken represent any resource, which is stably produced in the environment and then transported to the cells. The creation of the food in the environment is modeled according to the reaction equation:

$$\emptyset \xrightarrow{\lambda_f} f_r \quad , \tag{5.1}$$

where r is an uniform random number in $[1, k]$, and f_i represents the food available to the ith cell. Obviously, the Markovian dynamics guarantee that this is equivalent to having the following reaction equation for each cell i:

$$\emptyset \xrightarrow{\lambda_f k^{-1}} f_i \quad , \tag{5.2}$$

which makes us to notice that since the food is produced in a constant rate, the expected food available per cell is inversely proportional to the population size k, making explosion in the number of cells harmful to the population by leading to starvation.

Next, cellular activities are modeled as follows. Each cell will transform an item of food to a genetic product, via transcription and translation. This transformation also occurs in a stochastic manner. Moreover, the products of the gene expression are also let to degrade. For these purposes we have the following reactions:

$$P_i + f_i \xrightarrow{\infty} P_i(\tau_{P_i}) + p_i \tag{5.3}$$

$$p_i \xrightarrow{d_p} \emptyset \tag{5.4}$$

$$f_i \xrightarrow{d_f} \emptyset \quad , \tag{5.5}$$

where P_i denotes the occupancy of the promoter region, f_i the available food, and p_i the transcription product in the ith cell. Note that the reaction rate in equation 5.3

is let to be infinite, such that the time between production of the protein molecules is independent of the amount of food f_i present in the system, determined only by the promoter delay τ_{P_i}.

Additionally the promoter delay $\tau_{P_i} \sim \Gamma(\alpha_i, \beta_i)$ is let to be gamma distributed, with shape parameter α_i and rate parameter β_i. It is the parameters α_i and β_i, which the cells are allowed to mutate. The gamma distribution was chosen due to its flexibility to represent protein production dynamics ranging from sub- and super-Poissonian. The Poisson process in the abundance of genetic products, that is the result from the Markovian dynamics, can be obtained by letting the parameter $\alpha_i = 1$, while the parameter β_i determines the rate.

The mutations in the shape α_i and the rate β_i of the production interval distribution of the cell is performed in linear scale with exponentially distributed time intervals with rate of λ_m, with uniform probability to up or down. The mutation uses a step size of Δ_m, by which the parameter value is varied, that is, to mutate parameter x_i we set $x_i \leftarrow x_i + (-1)^d \Delta_m$, for $d \in \{0, 1\}$. Again, a consequence of the Markovian dynamics is that this is equivalent to performing the mutations independently up and down, each with one half the rate. Additionally, a constraint $0 < \alpha_i, \beta_i$ is applied to keep the system well-behaved. This only has an effect if a value of α_i or β_i becomes small.

The cell cycle is modeled using a fixed cell lifetime ℓ. During its lifetime the cell tries to reach the competence, or it will decease without offspring. Additionally, cell death can be triggered by starvation, which is caused by the cell running out of available food. On the other hand, when a cell manages to produce enough proteins it becomes competent and it divides. That is, it forms two new daughter cells, with parameters α_i and β_i inherited from the mother cell, and the proteins p_i and food f_i are split to the two daughter cells. The state of competence is determined based on a threshold T in the number of proteins present in the cell.

5.3 Results

The simulations were performed with an initial generation of population of size $k \leftarrow 10$. The food was let to be created with rate $\lambda_f \leftarrow 20\,T$, and degraded with rate $d_f \leftarrow 0.1$. The protein degradation rate was set to $d_p \leftarrow 1$, and the initial values of the parameters of gene expression were set to $\alpha_i \leftarrow 1$ and $\beta_i \leftarrow 5$, for each cell. This correspond to an exponential distribution with rate β_i, and will result in Poissonian dynamics. If there is no evolutionary advantage for non-Poissonian dynamics, the evolutionary programming should retain Poissonian dynamics.

The initial concentrations of the molecules were set to $[P_i] \leftarrow 1$, $[f_i] \leftarrow 0$, and $[p_i] \leftarrow \lfloor \beta_i \, (\alpha_i \, d_p)^{-1} \rfloor$, last of which is the expected protein level if there was infinite amount of food available, rounded towards zero. The cellular lifetime was set to

$\ell \leftarrow 100$, the mutation rate was set to $\lambda_m \leftarrow 1$, and the mutation step was set to $\Delta_m \leftarrow 0.02$.

Finally, the model was simulated with two values of threshold T for cell competence, namely $T \leftarrow 1$ and $T \leftarrow 10$, which were called the low and the high threshold case, respectively. Each of the models was simulated up to time point 500000, after which it was clear that the solution had arrived near the steady state that could be reached with current set of parameters. Moreover, 10 separate runs were done to guarantee that the obtained result was not a single stochastic pathway, but indeed the result is repeatable.

It was found out that by varying the threshold for cell competence various kinds of optima can be found. Figures 5.1a and 5.1b depict the time evolution of the distribution of the parameters α_i in the cell population, for the low and high threshold case, respectively. Recall that the shape parameter α can be used to tune the noise in the process of gene expression, whereas tuning the rate parameter β can be used to adjust the mean levels independently, maintaining the variance-to-mean-ratio of production constant.

There are interesting observations to be made from these figures. As depicted by figure 5.1a, it turned out that in an environment where the threshold for the competence is low, the cells appear to favor more deterministic control of gene expression than would be expected. It is evident that the distribution of values of α evolve to values greater than unity, making the process of gene expression sub-Poissonian. After a short period of time, the individuals who favored smaller shape parameter get eliminated, guaranteeing that there is not a single individual, whose value of α is smaller or equal to unity, as the time moves further on.

On the other hand, for high values of threshold, this evolutionary model appears to favor more noisy gene expression. In contrast to the low threshold condition, in figure 5.1b we see the shape parameter α to evolve to values strictly less than unity. This means that in an environment where the threshold for the competence is relatively high, the cells opt for having highly unpredictable timings between the productions of proteins, making the process of gene expression super-Poissonian.

The resulting protein distributions are presented in figures 5.2a and 5.2b, along with the expected distribution if gene expression was purely Poissonian, that is, for $\alpha = 1$. It should be noted that while these distributions do not appear to exhibit drastic differences to the Poisson distribution, the evolution of the shape parameter α in both cases (see figures 5.1a and 5.1b) shows that the non-Poissonian distribution definitely exhibits evolutionary advantage in comparison to the Poissonian, since the cells with the former have completely displaced the cells with the latter.

In both cases of threshold, more probability mass of the distribution is placed in the region that passes the threshold than would have been expected. This provides

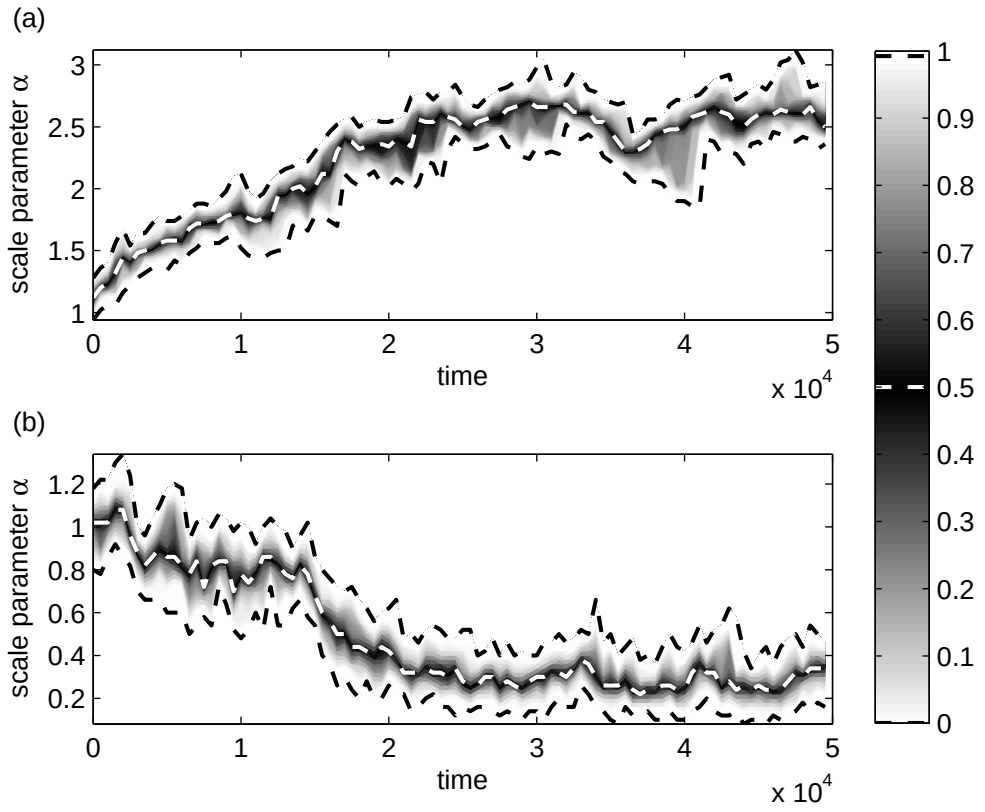

Figure 5.1: Time evolution of the distribution of the scale parameter α, in (a) an environment with low threshold for competence ($T \leftarrow 1$), and in (b) an environment with high threshold for competence ($T \leftarrow 10$). Different levels of gray represent quantiles of the population, with median denoted by the white dashed line, and maximum and minimum denoted by the black dashed lines.

the cells with means of having greater probability of crossing the threshold for competence, consequently allowing them to reproduce, and to pass on their evolved features. Additionally, in both cases, the bulk of the distribution lies below the threshold. Recall that the rate of gene expression is proportional to the consumption of food, so it is therefore favorable to decrease the rate of the protein production as much as possible, so as to conserve the available food and prevent starvation, as long as the competence can be reached with a probability high enough.

In the case of low threshold, the probability mass have been shifted to the bar corresponding concentration of a single protein in a cell. The rest of the tail is reduced, since producing extra proteins does not provide any advantage. Consequently, the overall variance of this distribution is reduced in comparison with the Poisson distribution. Note that due to the constraints on the shape of the distribution of the intervals between protein production in this specific model, the shape of the tail can have a restricted form, preventing complete elimination of the tail.

These effects are also visible in the case of the high threshold. However, since the

threshold is shifted, the distribution has evolved having a different kind of shape. Again, the bulk of the mass is placed below the threshold, making the expected consumption of food small, preventing the starvation of the cells. This is accompanied with a fat right tail, which provides the fluctuations by which the cell can reach the competence.

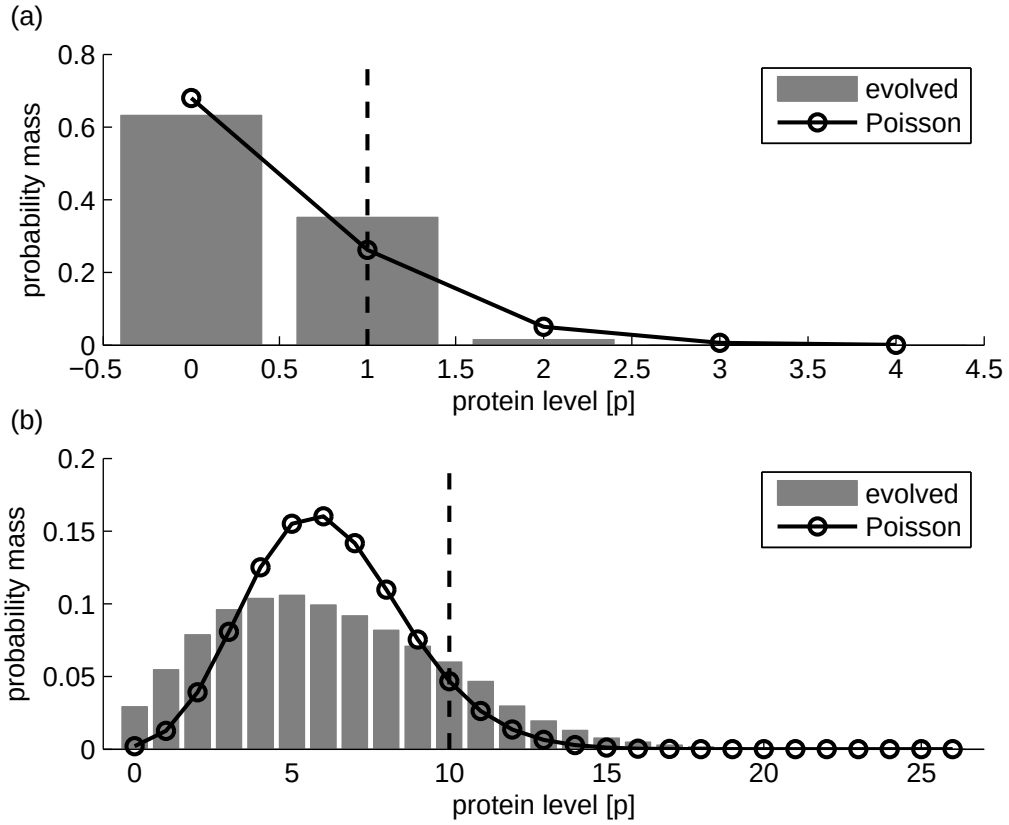

Figure 5.2: Distribution of protein concentration in the cells, at time moment 50000, in (a) an environment with low threshold for competence ($T \leftarrow 1$), and in (b) an environment with high threshold for competence ($T \leftarrow 10$). Solid lines with round markers represent Poisson distributions with equivalent production rate and the vertical dashed lines represent the threshold.

To summarize, in the case where the threshold is low, the competence can be easily reached. Consequently, it is favorable to save the food to prevent the starvation of the cell, and finally reach the competence with more precisely timed production of proteins during the lifetime of the cell. In the case of the high threshold, due to the extremely noisy dynamics of gene expression, the mean expression of food can be kept low. Again, this allows a large fraction of the cells to escape starvation, but since the fluctuations have been evolved to be large, there is still a large probability for producing a burst of proteins, guaranteeing that the cell to be able to reach the state of competence during its lifetime.

There are few additional things to be noted. First, in the study, the cells were

placed in a homogeneous environment. This is likely not to be the case in real world, but different cells are subject to different environmental conditions, for example, the expected amount of food available to a cell is likely to depend on several factors. For example, it is likely that there is a cost of transport in the environment, such that more distant locations of cells are subject to less available food. Another example is the density of the cells. If there is an abundance of food in specific locations, the cell density is likely to increase, either due to cells migrating from other areas to that specific area, or just by survival and reproduction of the cells that were localized there in the first place. Third, it might be that there is a scheme of competition of the food, such that the amount of the food that a cell is able to gather is proportional to some property of the cell.

Next, in figures 5.1a and 5.1b it appears that the values of the shape parameter α has not yet been well converged. This stochasticity in the shape parameters is inherent to the model, since the mutation rate is fixed. In cases where the stability of the parameters is of importance, the cellular systems might be able to tune the rate of mutations. This can be done, for example, by regulating facilities that provide error correction in the processes of replication of genetic material [32].

Finally, it is important to note that the initial value might play a role [35]. Since we wanted to compare the dynamics of the gene expression with the Poissonian model of gene expression, it was a sufficiently safe assumption to start with a homogeneous population of cells with $\alpha = 1$. Also the evolved number of parameters in this model is rather small, and cost function for the cells is expected to be sufficiently smooth, at least in terms of α and β. It is less straightforward to predict if the effects of the variable population size k provide means of arriving to a multiple minima.

6. DISCUSSION

A simulator was introduced in this work, which allows the evolution of cellular processes and genetic circuits to be simulated in a probabilistic manner. In addition to the simulator, strategies for modeling such systems were presented, allowing the construction of studies that investigate evolutionary phenomena in cellular systems. Such a computational tool has not been previously available, either preventing such studies, or requiring the construction of an ad-hoc tool for these purposes. As an example of the applicability of this tool, two biologically relevant example studies were presented, in which the different features of the simulator were exploited.

The presented tool aims to be applicable for studying trajectories of complex systems that are shaped by evolutionary processes. Due to the dimensionality of these problems, it is not feasible nor intended that the tool would allow one to predict where the evolution will lead, but rather to generate these trajectories that can be used to obtain insight of the available evolutionary paths. Moreover, the tool provides means for obtaining information on the behavior more restricted systems, such as the evolution of bacterial populations in controlled environments. This might be useful, for example, in such studies prior to wet lab experiments, which tend to be laborious and expensive, unlike computer simulations. Alternatively, the tool might prove itself useful as a tool aiding the engineering of genetic networks, to reveal the circuits that not only behave well under the specific conditions, but also can cope with environmental changes.

In addition to the modeling and simulation of genetic networks, the simulator can be used as part of a stochastic and/or evolutionary optimization process. For problems with large dimensionality, many of the parameters of the systems can be let freely mutate, and a carefully specified fitness function can be used to drive the optimization process to reveal solutions of parameters that are a good candidates for solving the problem.

Finally, the presented tool can be used to generate realistic data that can be used as an input for building tools that convert the measurement data of real world experiments to some higher form. For example, the simulated data can be used to train and evaluate performance of tools that attempt to infer the time-varying structures of networks that are under evolutionary pressure.

In the study that acted as the first example, the effects of environmental changes

to the phenotypic diversity and mutation rates were studied. It was found that environmental changes promote the introduction of additional population diversity, and that in highly transient and unpredictable environments higher rates of generation of this variability is favored, for example, by controlling the mutation rates. Moreover, it was shown that neutral mutations can lead to complex distribution of phenotypes in cellular populations, by allowing multiple solutions to be optimal.

The second example focused on the evolution of gene expression dynamics in stationary environmental conditions. It was found that small changes in the evolutionary constraints can drive a population to favor different levels of stochasticity in their cellular processes. Moreover, it was shown that even small changes in the details of these processes will lead to generation of phenotypes that provide significant evolutionary advantage to trigger specific paths of evolution.

For future, it is planned that the number of features in the simulator will be increased, such that its applicability of quickly building and simulating stochastic kinetic models with evolutionary programming becomes even more easier. Moreover, it is of interest to make the tool more friendly to end-users that are unfamiliar with the intricate details that are involved in the evolutionary processes of stochastic networks, without the loss of the simulators applicability.

Additionally, it is of interest to use the simulator to perform studies of evolution in simple genetic circuits in complex environments. Among other genetic motifs fundamental to life, the circuits to start with could be genetic toggle switches and three-gene repressilators. Currently, the availability of such studies is limited, and while the dynamical features of these processes have been studied previously, the understanding of their behavior under evolutionary pressure is rather poor. Another interesting line of work that the simulator allows to study involves investigating the effects of evolution to isolated subpopulations. While such isolation might be unlikely to occur in realistic networks by chance, intentional damage might cause transient isolation of parts of the networks.

Furthermore, it would be interesting to see if the phenomena observed in the examples can be reproduced using live cells. For example, live bacteria can be used in such experiments, due to the fact that their genotype more limited in size and their rate of reproduction is fast. Additionally current measurement techniques allows the observations in single mRNA and protein level. For example, one could test if different controlled environmental conditions lead to the evolution of populations with different variability in their gene expression.

BIBLIOGRAPHY

[1] C. Darwin, *On the origin of species by means of natural selection, or the preservation of favoured races in the struggle for life.* London, UK: John Murray, 1859.

[2] G. Mendel, "Versuche über pflanzen-hybriden," in *Verhandlungen des naturforschenden Vereines in Brünn*, vol. IV, pp. 3–47, Brno, CZ: Verlage des Vereines, 1866.

[3] E. Mayr, "Foreword," in *Variation: A central concept in biology* (B. Halgrimsson and B. Hall, eds.), p. xvii, Amsterdam, NL: Elsevier Academic Press, 2005.

[4] R. Hershberg and D. A. Petrov, "Evidence that mutation is universally biased towards at in bacteria," *PLoS Genet.*, vol. 6, no. 9, p. e1001115, 2010.

[5] W. Johanssen, "The genotype conception of heredity," *Am. Nat.*, vol. 45, no. 531, pp. 129–159, 1911.

[6] E. Mayr, "The objects of selection," *Proc. Natl. Acad. Sci. U.S.A.*, vol. 99, no. 6, pp. 2091–2094, 1997.

[7] B. Alberts, A. Johnson, J. Lewis, M. Raff, K. Roberts, and P. Walters, *Molecular Biology of the Cell.* New York, NY: Garland Science, 2002.

[8] G. M. Suel, J. Garcia-Ojalvo, L. M. Liberman, and M. B. Elowitz, "An excitable gene regulatory circuit induces transient cellular differentiation," *Nature*, vol. 440, no. 7083, pp. 545–550, 2006.

[9] M. B. Elowitz and S. Leibler, "A synthetic oscillatory network of transcriptional regulators," *Nature*, vol. 403, no. 6767, pp. 335–338, 2000.

[10] S. A. Kauffman, "Metabolic stability and epigenesis in randomly constructed genetic nets," *J. Theor. Biol.*, vol. 22, no. 3, 1969.

[11] G. Yagil and E. Yagil, "On the realtion between effector concentration and the rate of induced enzyme synthesis," *Biophys. J.*, vol. 11, no. 1, pp. 11–27, 1971.

[12] T. Mestl, E. Plahte, and S. W. Omholt, "A mathematical framework for describing and analysing gene regulatory networks," *J. Theor. Biol.*, vol. 176, no. 2, pp. 291–300, 1995.

[13] N. Friedman, M. Linial, I. Nachman, and D. Pe'er, "Using bayesian networks to analyze expression data," *J. Comput. Biol.*, vol. 7, no. 3–4, pp. 601–620, 2000.

[14] D. C. Weaver, C. T. Workman, and G. D. Stormo, "Modeling regulatory networks with weight matrices," *Pac. Symp. Biocomput.*, vol. 4, no. 1, pp. 112–123, 1999.

[15] R. N. Stuart and E. W. Branscomb, "Quantitative theory of in vivo lac regulation: significance of repressor packaging : I. equilibrium considerations," *J. Theor. Biol*, vol. 31, no. 2, pp. 313–329, 1971.

[16] M. B. Elowitz, A. J. Levine, E. D. Siggia, and P. S. Swain, "Stochastic gene expression in a single cell," *Science*, vol. 16, no. 5584, pp. 1183–1186, 2002.

[17] E. Ozbudak, M. Thattai, I. Kurtser, A. D. Grossman, and van Oudenaarden Alexander, "Regulation of noise in the expression of a single gene," *Nat. Genet.*, vol. 31, no. 1, pp. 69–73, 2002.

[18] W. J. Blake, M. Kaern, C. R. Cantor, and J. J. Collins, "Noise in eukaryotic gene expression," *Nature*, vol. 10, no. 6932, pp. 633–637, 2003.

[19] H. Maamar, A. Raj, and D. Dubnau, "Noise in gene expression determines cell fate in *Bacillus subtilis*," *Science*, vol. 317, no. 5837, pp. 526–529, 2007.

[20] H. H. McAdams and A. Arkin, "Stochastic mechanisms in gene expression," *Proc. Natl. Acad. Sci. U.S.A.*, vol. 94, no. 3, pp. 814–819, 1997.

[21] A. Arkin, J. Ross, and H. H. McAdams, "Stochastic kinetic analysis of developmental pathway bifurcation in phage λ-infected *Escherichia coli* cells," *Genetics*, vol. 149, no. 4, pp. 1633–1648, 1998.

[22] J. Hasty, D. McMillen, F. Isaacs, and J. J. Collins, "Computational studies of gene regulatory networks: *in numero* molecular biology," *Nat. Rev. Genet.*, vol. 2, no. 4, pp. 268–279, 2001.

[23] M. Thattai and A. van Oudenaarden, "Intrinsic noise in gene regulatory networks," *Proc. Natl. Acad. Sci. U.S.A.*, vol. 98, no. 15, pp. 8614–8619, 2001.

[24] L.-h. So, A. Ghosh, C. Zong, L. A. Sepulveda, R. Segev, and I. Golding, "General properties of transcriptional time series in *Escherichia coli*," *Nat. Genet.*, vol. 43, no. 6, pp. 554–560, 2011.

[25] M. Bon, S. J. McGowan, and P. R. Cook, "Many expressed genes in bacteria and yeast are transcribed only once per cell cycle," *FASEB J.*, vol. 20, no. 10, pp. 1721–1723, 2006.

[26] R. Wu and M. Lin, "Functional mapping – how to map and study the genetic architecture of dynamic complex traits," *Nat. Rev. Gen.*, vol. 7, no. 3, pp. 229–237, 2006.

[27] M. S. Samoilov, G. Price, and A. P. Arkin, "From fluctuations to phenotypes: The physiology of noise," *Sci. STKE*, vol. 2006, no. 366, p. re17, 2006.

[28] R. Kellermayer, "Physiologic noise obscures genotype-phenotype correlations," *Am. J. Med. Genet.*, vol. 143A, no. 12, pp. 1306–1307, 2007.

[29] J. Maynard Smith, *The evolution of sex.* Cambridge, UK: Cambridge University Press, 1978.

[30] M. Kaern, T. C. Elston, W. J. Blake, and J. J. Collins, "Stochasticity in gene expression: From theories to phenotypes," *Nat. Rev. Genet.*, vol. 6, no. 6, pp. 451–464, 2005.

[31] C. Pal, M. D. Macia, A. Oliver, and A. Schachar, Iraand Buckling, "Coevolution with viruses drives the evolution of bacterial mutation rates," *Nature*, vol. 450, no. 7172, pp. 1079–1081, 2007.

[32] A. Giraud, I. Matic, O. Tenaillon, A. Clara, M. Radman, M. Fons, and F. Taddei, "Costs and benefits of high mutation rates: Adaptive evolution of bacteria in the mouse gut," *Science*, vol. 291, no. 5513, pp. 2606–2608, 2001.

[33] J. Gerhart and M. Kirschner, *Cells, embryos, and evolution.* Malden, MA: Blackwell, 1997.

[34] H. J. Bremermann, "Optimization through evoltion and recombination," in *Self-organizing systems* (M. C. Yovits, G. T. Jacobi, and G. D. Goldstein, eds.), pp. 93–106, Washington, DC: Spartan Books, 1962.

[35] T. Lu, T. Shen, M. R. Bennett, P. G. Wolynes, and J. Hasty, "Phenotypic variability of growing cellular populations," *Proc. Natl. Acad. Sci. U.S.A.*, vol. 104, no. 48, pp. 18982–18897, 2007.

[36] ISO/IEC, "ISO/IEC 14882:1998(E): Programming languages C++." International standard, 1998.

[37] R. Ierusalimschy, L. H. de Figueiredo, and W. Celes, *Lua 5.1 reference manual.* Lua.org, 2006.

[38] M. W. Hirsch and S. Smale, *Differential equations, dynamical systems, and linear algebra*. San Diego, CA: Academic Press, 1974.

[39] A. D. Polyanin and V. F. Zaitsev, *Handbook of exact solutions for ordinary differential equations*. Boca Raton, FL: CRC Press, 1995.

[40] J. C. Butcher, *Numerical methods for ordinary differential equations*. Chichester, UK: John Wiley and Sons, 2005.

[41] D. A. McQuarrie, "Stochastic approach to chemical kinetics," *J. Appl. Prob.*, vol. 4, no. 3, pp. 413–478, 1967.

[42] D. T. Gillespie, "A general method for numerically simulating the stochastic time evolution of coupled chemical reactions," *J. Comput. Phys.*, vol. 22, no. 4, pp. 403–434, 1976.

[43] D. T. Gillespie, "Concerning the validity of the stochastic approach to chemical kinetics," *J. Stat. Phys.*, vol. 16, no. 3, pp. 311–318, 1977.

[44] D. T. Gillespie, "Exact stochastic simulation of coupled chemical reactions," *J. Phys. Chem.*, vol. 81, no. 25, pp. 2340–2361, 1977.

[45] D. T. Gillespie, "A rigorious derivation of the chemical master equation," *Physica A*, vol. 188, no. 1–3, pp. 404–425, 1992.

[46] T. G. Kurtz, "The relationship between stochastic and deterministic models for chemical reactions," *J. Chem. Phys.*, vol. 57, no. 7, pp. 2976–2978, 1972.

[47] T. G. Kutrz, "Limit theorems for sequences of jump markov processes approximating ordinary differential process," *J. Appl. Prob.*, vol. 8, no. 2, pp. 344–356, 1971.

[48] D. Bratsun, D. Volfson, L. S. Tsimring, and J. Hasty, "Delay-induced stochastic oscillations in gene regulation," *Proc. Natl. Acad. Sci. U.S.A.*, vol. 102, no. 41, pp. 14593–14598, 2005.

[49] M. R. Roussel and R. Zhu, "Validation of an algorithm for delay stochastic simulation of transcription and translation in prokaryotic gene expression title," *Phys. Biol.*, vol. 3, no. 4, pp. 274–284, 2006.

[50] D. Kennel and H. Riezman, "Transcription and translation initiation frequencies of the *Escherichia coli lac* operon," *J. Mol. Biol.*, vol. 114, no. 1, pp. 1–21, 1977.

[51] C. Chapon, "Expression of *malT*, the regulator gene of the maltose region in *Escherichia coli*, is limited both at transcription and translation," *EMBO J*, vol. 1, no. 3, pp. 369–374, 1982.

[52] A. Raj, C. S. Peskin, D. Tranchina, D. Y. Vargas, and S. Tyagi, "Stochastic mrna synthesis in mammalian cells," *PLoS Biol.*, vol. 4, no. 10, p. e309, 2006.

[53] O. Yarchuck, N. Jacques, J. Guillerez, and M. Dreyfus, "Interdependence of translation, transcription and mrna degradation in the lacz gene," *J. Mol. Biol.*, vol. 226, no. 3, pp. 581–596, 1992.

[54] I. Golding, J. Paulsson, S. M. Zawilski, and E. C. Cox, "Real-time kinetics of gene activity in individual bacteria," *Cell*, vol. 123, no. 6, pp. 1025–1036, 2005.

[55] J. R. Chubb, T. Trcek, S. M. Shenoy, and R. H. Singer, "Transcriptional pulsing of a developmental gene," *Curr. Biol.*, vol. 16, no. 10, pp. 1018–1025, 2006.

[56] P. Guptasarma, "Does replication-induced transcription regulate synthesis of the myriad low copy number proteins of *Escherichia coli*?," *Bioessays*, vol. 17, no. 11, pp. 987–997, 1995.

[57] J. A. Bernstein, A. B. Khodursky, P.-H. Lin, S. Lin-Chao, and S. N. Cohen, "Global analysis of mrna decay and abundance in *Escherichia coli* at single-gene resolution using two-color fluorescent dna microarrays," *Proc. Natl. Acad. Sci. U.S.A.*, vol. 99, no. 15, pp. 9697–9702, 2002.

[58] S. Ghaemmaghami, W.-K. Huh, K. Bower, R. W. Howson, A. Belle, N. Dephoure, E. K. O'Shea, and J. S. Weissman, "Global analysis of protein expression in yeast," *Nature*, vol. 425, no. 6959, pp. 737–741, 2003.

[59] W. R. McClure, "Mechanism and control of transcription initiation in prokaryotes," *Annu. Rev. Biochem.*, vol. 54, no. 1, pp. 171–204, 1985.

[60] P. J. Halling, "Do the laws of chemistry apply to living cells?," *Trends Biochem. Sci.*, vol. 14, no. 8, pp. 317–318, 1989.

[61] J. Yu, J. Xiao, X. Run, K. Lao, and X. S. Xie, "Probing gene expression in live cells, one protein molecule at a time," *Science*, vol. 311, no. 5767, pp. 1600–1603, 2006.

[62] A. S. Ribeiro, R. Zhu, and S. A. Kauffman, "A general modeling strategy for gene regulatory networks with stochastic dynamics," *J. Comp. Biol.*, vol. 13, no. 9, pp. 1630–1639, 2006.

[63] R. Zhu, A. S. Ribeiro, D. Salahub, and S. A. Kauffman, "Studying genetic regulatory networks at the molecular level: Delayed reaction stochastic models," *J. Theor. Biol.*, vol. 246, no. 4, pp. 725–745, 2007.

[64] A. S. Ribeiro, O.-P. Smolander, T. Rajala, A. Hakkinen, and O. Yli-Harja, "Delayed stochastic model of transcription at the single nucleotide level," *J. Comp. Biol.*, vol. 16, no. 4, pp. 539–553, 2009.

[65] J. Makela, J. Lloyd-Price, O. Yli-Harja, and A. S. Ribeiro, "Stochastic sequence-level model of coupled transcription and translation in prokaryotes," *BMC Bioinf.*, vol. 12, no. 1, p. 121, 2011.

[66] F. J. Grundy and T. M. Henkin, "From ribosome to riboswitch: Control of gene expression in bacteria by rna structural rearragements," *Crit. Rev. Biochem. Mol. Biol.*, vol. 41, no. 6, pp. 329–338, 2006.

[67] S. J. Greive and P. H. von Hippel, "Thinking quantatively about transcriptional regulation," *Nat. Rev. Mol. Cell. Biol.*, vol. 6, no. 3, pp. 221–232, 2005.

[68] K. M. Herbert, A. La Porta, B. J. Wong, R. A. Mooney, K. C. Neuman, R. Landick, and S. M. Block, "Sequence-resolved detection of pausing by single rna polymerase molecules," *Cell*, vol. 125, no. 6, pp. 1083–1094, 2006.

[69] R. Landick, "The regulatory roles and mechanism of transcriptional pausing," *Biochem. Soc. Trans.*, vol. 34, no. 6, pp. 1062–1066, 2006.

[70] S. Huang, "Gene expression profilling, genetic networks, and cellular states: an integrating concept for tumorigenesis and drug discovery," *J. Mol. Med.*, vol. 77, no. 6, pp. 469–480, 1999.

[71] I. Shmulevich, E. R. Dougherty, S. Kim, and W. Zhang, "Probabilistic boolean networks: a rule-based uncertainty model for gene regulatory networks," *Bioinformatics*, vol. 18, no. 2, pp. 261–274, 2002.

[72] I. Shmulevich, E. R. Dougherty, and W. Zhang, "From boolean to probabilistic boolean networks as models of genetic regulatory networks," *Proc. IEEE*, vol. 90, no. 11, pp. 1778–1792, 2002.

[73] A. Goldbeter, *Biochemical oscillations and cellular rhythms: The molecular bases of periodic and chaotic behaviour*. Cambridge, UK: Cambridge University Press, 1997.

[74] A. S. Ribeiro and J. Lloyd-Price, "Sgn sim, a stochastic genetic networks simulator," *Bioinf.*, vol. 23, no. 6, pp. 777–779, 2007.

[75] M. Kirschner and J. Gerhart, "Evolvability," *Proc. Natl. Acad. Sci. U.S.A.*, vol. 95, no. 15, pp. 8420–8427, 1998.

[76] S. A. Sawyer, J. Parsch, Z. Zhang, and D. L. Hartl, "Prevalence of positive selection among nearly neutral amino acid replacements in *Drosophila*," *Proc. Natl. Acad. Sci. U.S.A.*, vol. 104, no. 16, pp. 6504–6510, 2007.

[77] J. Maynard Smith, N. H. Smith, M. O'Rourke, and B. G. Spratt, "How clonal are bacteria?," *Proc. Natl. Acad. Sci. U.S.A.*, vol. 90, no. 10, pp. 4384–4388, 1993.

[78] R. Jain, M. C. Rivera, and J. A. Lake, "Horizontal gene transfer among genomes: The complexity hypothesis," *Proc. Natl. Acad. Sci. U.S.A.*, vol. 96, no. 7, pp. 3801–3806, 1999.

[79] P. Erdos and A. Renyi, "On random graphs i," *Publ. Math. Debrecen*, vol. 6, no. 1, pp. 290–297, 1959.

[80] D. J. Watts and S. H. Stroganz, "Collective dynamics of 'small-word' networks," *Nature*, vol. 393, no. 6684, pp. 440–442, 1998.

[81] A.-L. Barabasi and R. Albert, "Emergence of scaling in random networks," *Science*, vol. 286, no. 5439, pp. 509–512, 1999.

[82] R. Albert and A.-L. Barabasi, "Statistical mechanism of complex networks," *Rev. Mod. Phys.*, vol. 74, no. 1, pp. 47–97, 2002.

[83] R. Cohen and S. Havlin, "Scale-free networks are ultrasmall," *Phys. Rev. Lett.*, vol. 90, no. 5, p. 058701, 2003.

[84] D. J. de Solla Price, "Networks of scientific papers," *Science*, vol. 149, no. 3683, pp. 510–515, 1965.

[85] A. Clauset, C. Rohilla Shalizi, and M. E. J. Newman, "Power-law distributions in empirical data," *SIAM Rev.*, vol. 51, no. 4, pp. 661–703, 2009.

[86] H. Li and L. Petzold, "Logarithmic direct method for discrete stochastic simulation of chemically reacting systems," tech. rep., Department of Computer Science, University of California, Santa Barbara, 2006.

[87] M. Matsumoto and T. Nishimura, "Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator," *ACM T. Model. Comput. S.*, vol. 8, no. 1, pp. 3–30, 1998.

[88] J. Lloyd-Price and A. S. Ribeiro, "Sgnsim simulator manual." Retrieved on Aug 05 2011 from `http://www.cs.tut.fi/~sanchesr/SGN/manual_sgns.pdf`.

[89] R. A. Fisher and F. Yates, *Statistical tables for biological, agricultural and medical research.* London, UK: Oliver & Boyd, 1963.

[90] A. Hakkinen, F. G. Biddle, O.-P. Smolander, O. Yli-Harja, and A. S. Ribeiro, "Evolutionary dynamics of a population of cells with a toxin suppressor gene," in *Transactions on Computational Systems Biology XIII* (C. Priami, R.-J. Back, I. Petre, and E. de Vink, eds.), vol. 6575 of *Lecture Notes in Computer Science*, pp. 1–12, Heidelberg, DE: Springer Berlin, 2011.

[91] M. T. Korpela, J. S. Kurittu, J. T. Karvinen, and M. T. Karp, "A recombinant *Escherichia coli* sensor strain for the detection of tetracyclines," *Anal. Chem.*, vol. 70, no. 21, pp. 4457–4462, 1998.

[92] B. Eckert and C. F. Beck, "Overproduction of transposon tn10-encoded tetracycline resistance protein results in cell death and loss of membrane potential," *J. Bacteriol.*, vol. 171, no. 6, pp. 3557–3559, 1989.

[93] W. Hillen and C. Berens, "Mechanisms underlying expression of tn10 encoded tetracycline resistance," *Annu. Rev. Microbiol.*, vol. 48, no. 1, pp. 345–369, 1994.

[94] P. L. Foster, "Sorting out mutation rates," *Proc. Natl. Acad. Sci. U.S.A.*, vol. 96, no. 14, pp. 7617–7618, 1999.

[95] Y. Taniguchi, P. J. Choi, G.-W. Li, H. Chen, M. Babu, J. Hearn, A. Emili, and X. S. Xie, "Quantifying *E. coli* proteome and transcriptome with single-molecule sensitivity in single cells," *Science*, vol. 329, no. 5991, pp. 533–538, 2010.

[96] A. S. Ribeiro, A. Hakkinen, H. Mannerstrom, J. Lloyd-Price, and O. Yli-Harja, "Effects of the promoter open complex formation on gene expression dynamics," *Phys. Rev. E*, vol. 81, no. 1, p. 011912, 2010.

[97] T. Rajala, A. Hakkinen, S. Healy, O. Yli-Harja, and A. S. Ribeiro, "Effects of transcriptional pausing on gene expression dynamics," *PLoS Comput. Biol.*, vol. 6, no. 3, p. e1000704, 2010.

[98] A. S. Ribero, A. Hakkinen, S. Healy, and O. Yli-Harja, "Dynamical effects of transcriptional pause-prone sites," *Comp. Biol. Chem.*, vol. 34, no. 3, pp. 143–148, 2010.

[99] J. Paulsson, "Summing up the noise in gene networks," *Nature*, vol. 427, no. 6973, pp. 415–418, 2004.

[100] J. M. Pedraza and J. Paulsson, "Effects of molecular memory and bursting on fluctuations in gene expression," *Science*, vol. 319, no. 5861, pp. 339–343, 2008.